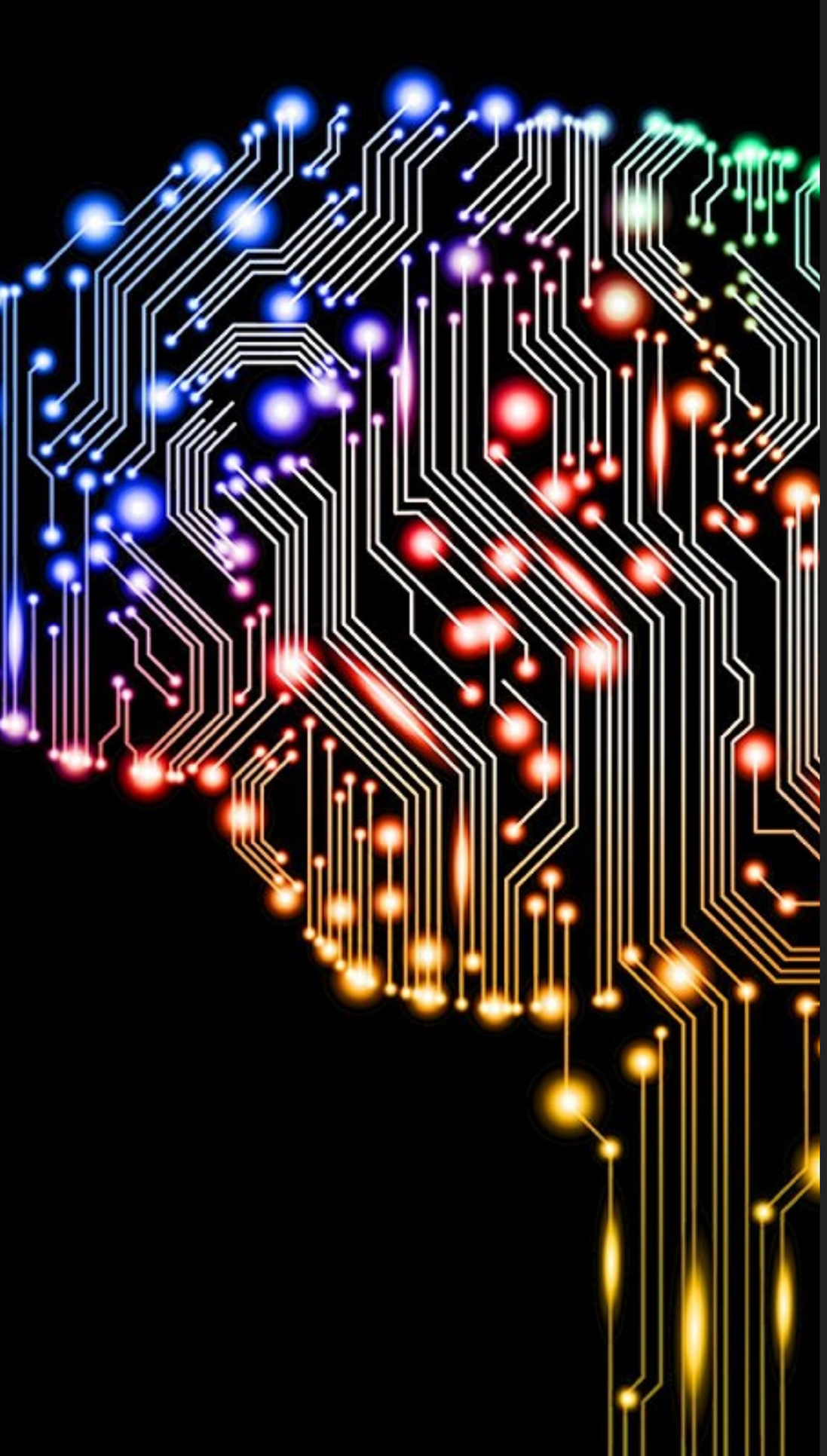
다우기술 S/W 인공지능랩 김정규

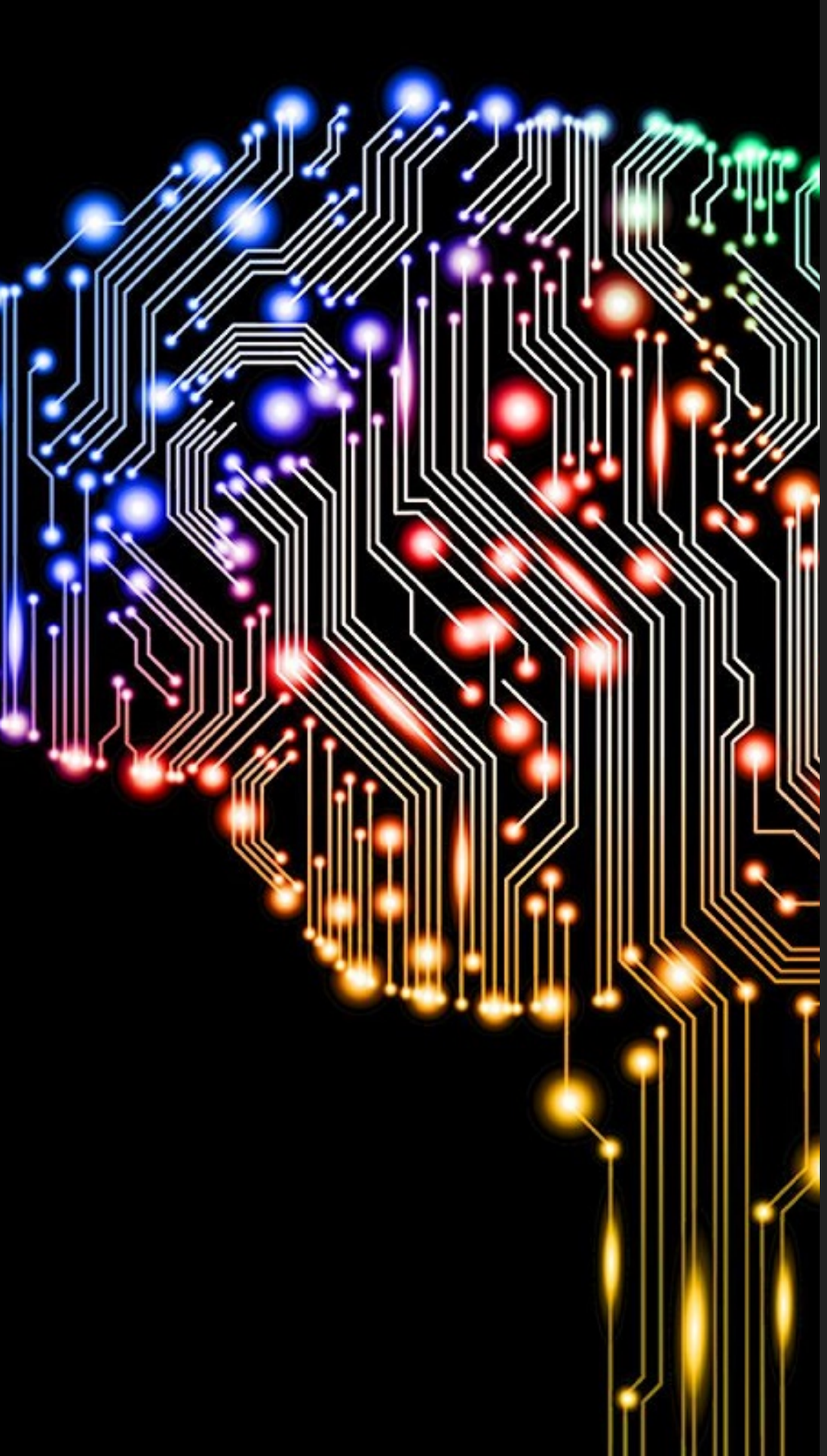
CNN 파헤치기(3탄)



지난 세미나에는

- AlexNet
- VGG
- GoogleNet

이번시간에는...

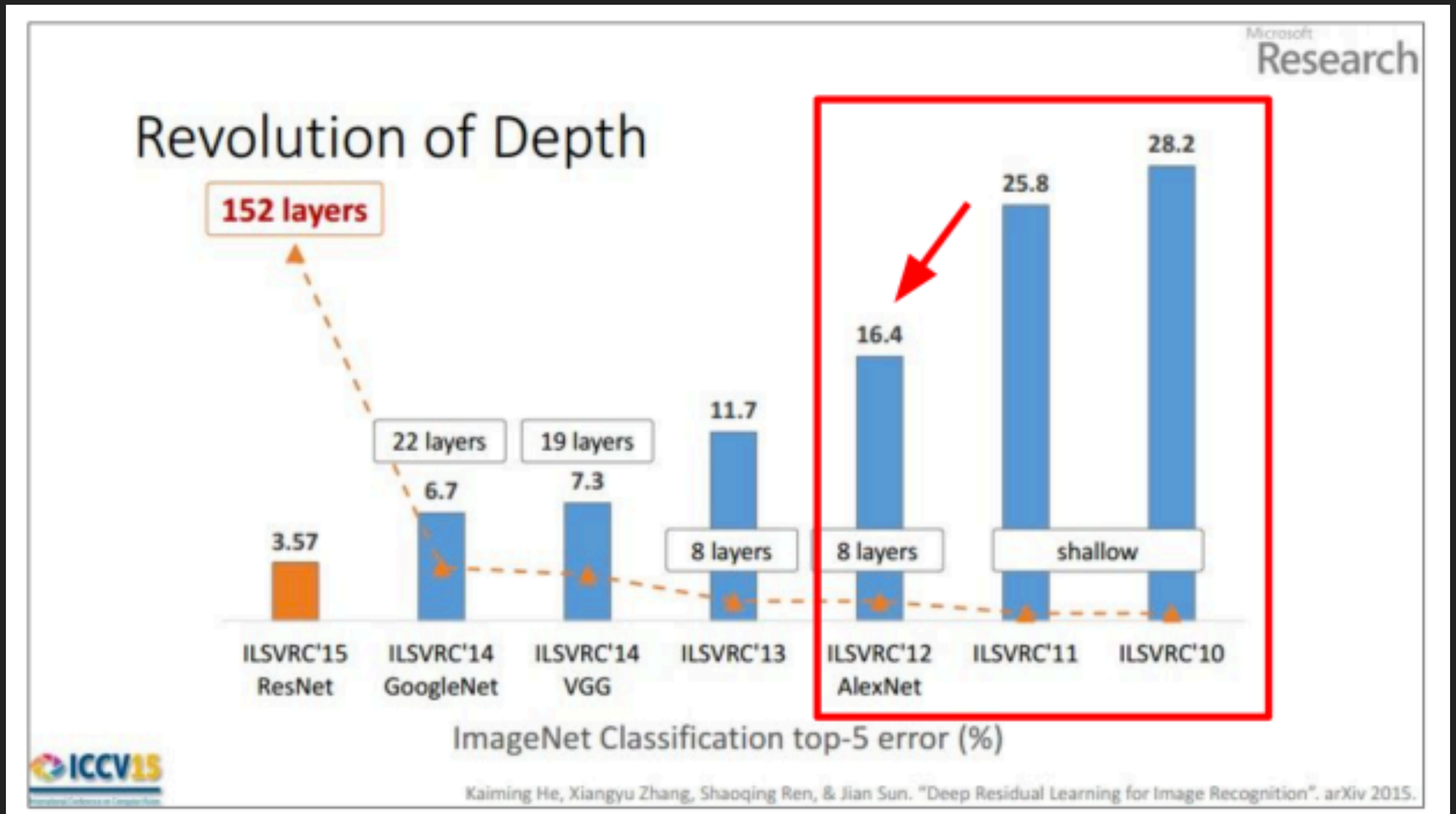


이번 세미나에는

- ResNet
- Overfitting
- Regularization

WHY?

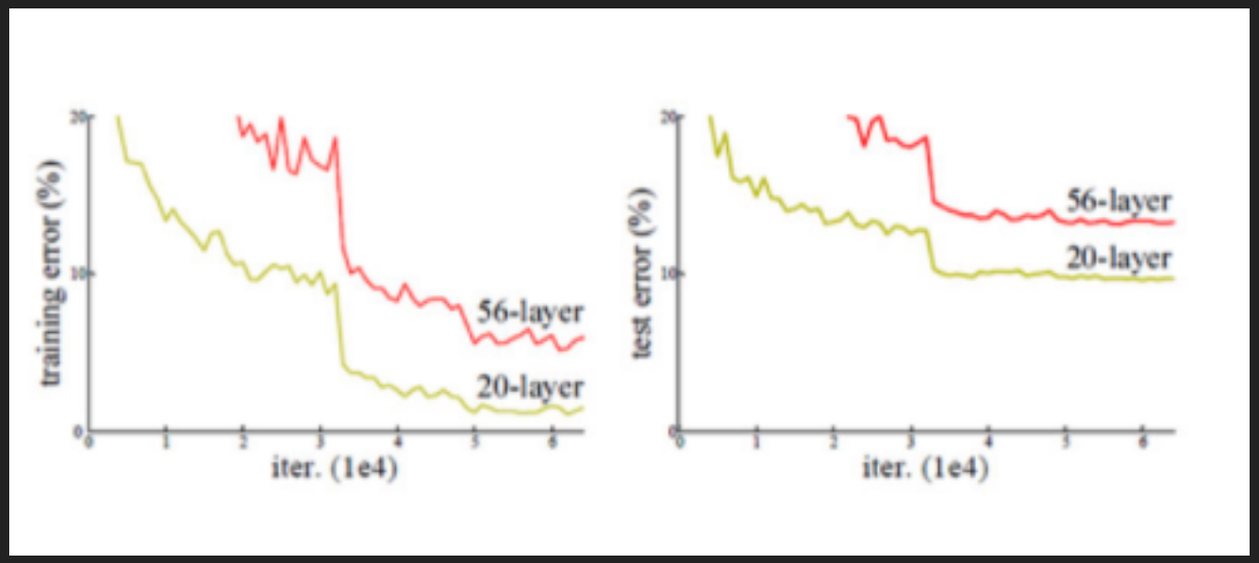
What is Resnet?



(slide from Kaiming He's recent presentation)

What is Resnet?

- 논문은 문제 제기로 시작함.
  "Is learning better networks as easy as stacking more layers?"

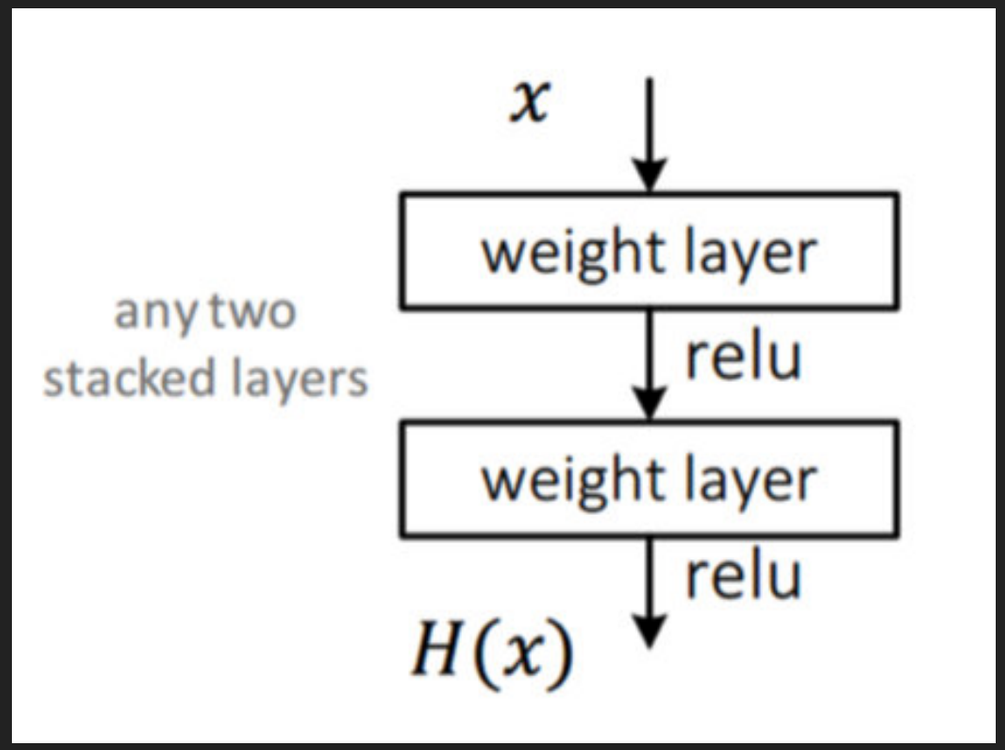


Result

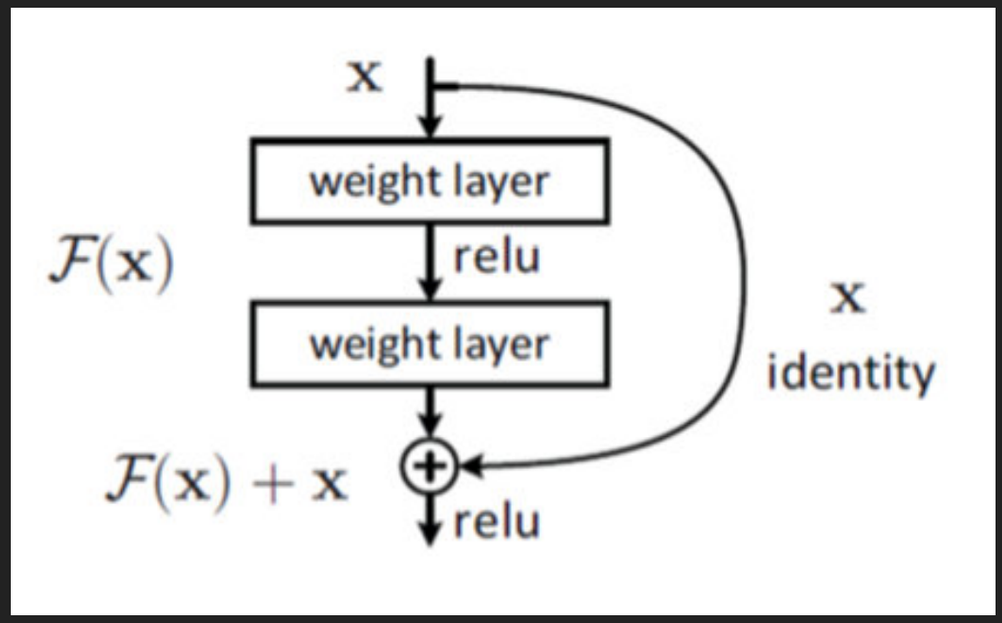
- Vanishing/Exploding Gradient 문제
- 학습망이 깊어질수록 파라미터의 수가 비례적으로 늘어나게 되어 overfitting 의 문제가 아닐지라도 오히려 에러가 커지는 상황이 발생

!!! Residual Learning !!!

- Residual Learning은 단순한 가정에서 비롯됨.



[기존의 방식]

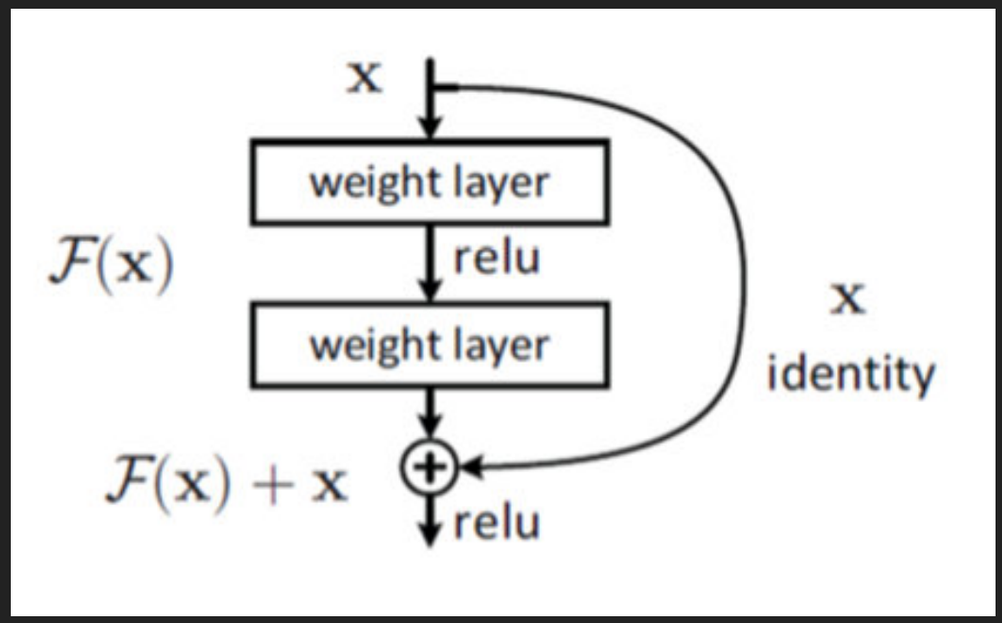


[새로운 방식]

!!! Residual Learning !!!

- Residual Learning은 단순한 가정에서 비롯됨.

- Result
1. 깊은 망도 쉽게 최적화가 가능하다.
2. 늘어난 깊이로 인해 정확도를 개선할 수 있다.

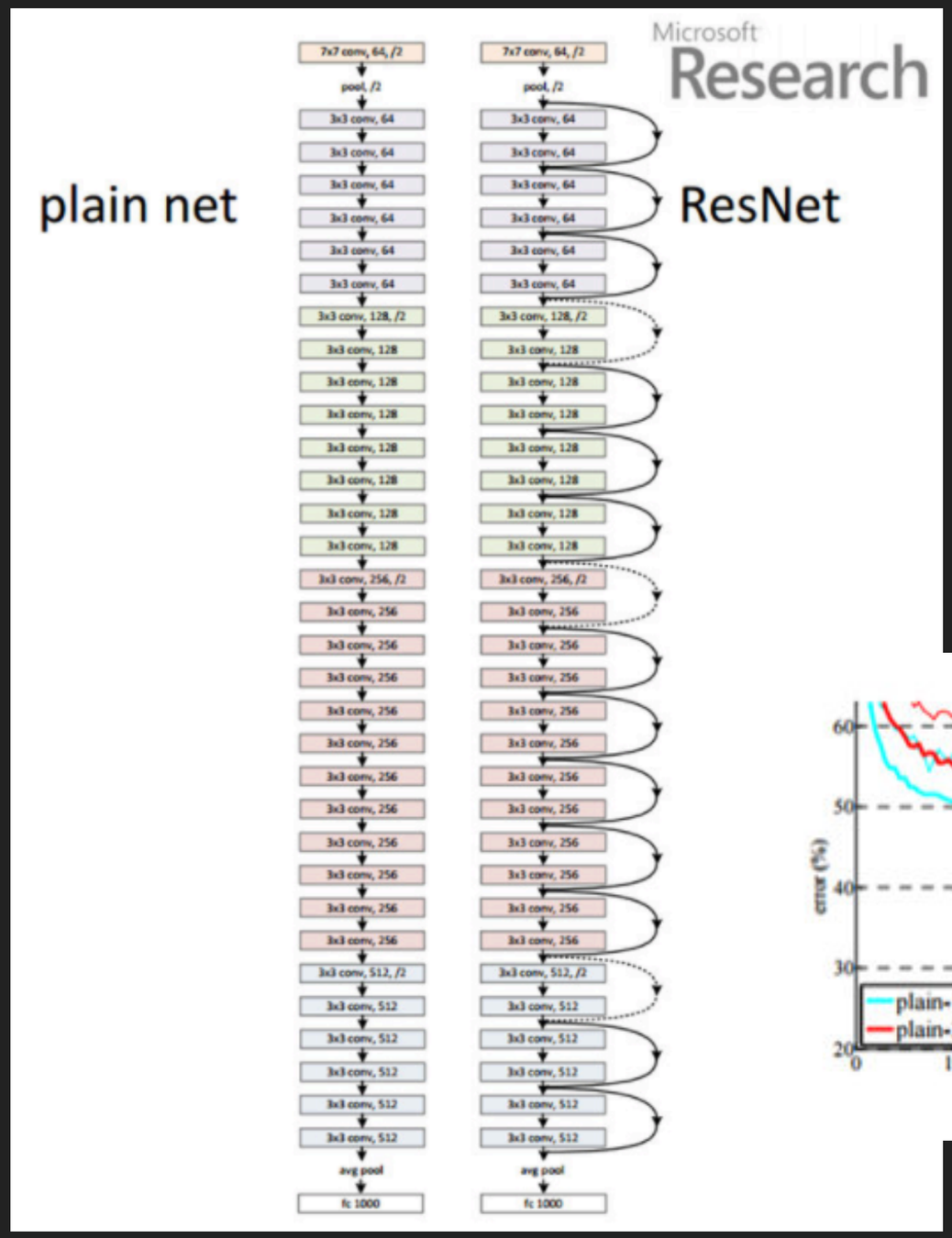


[새로운 방식]

!!! Residual Learning More

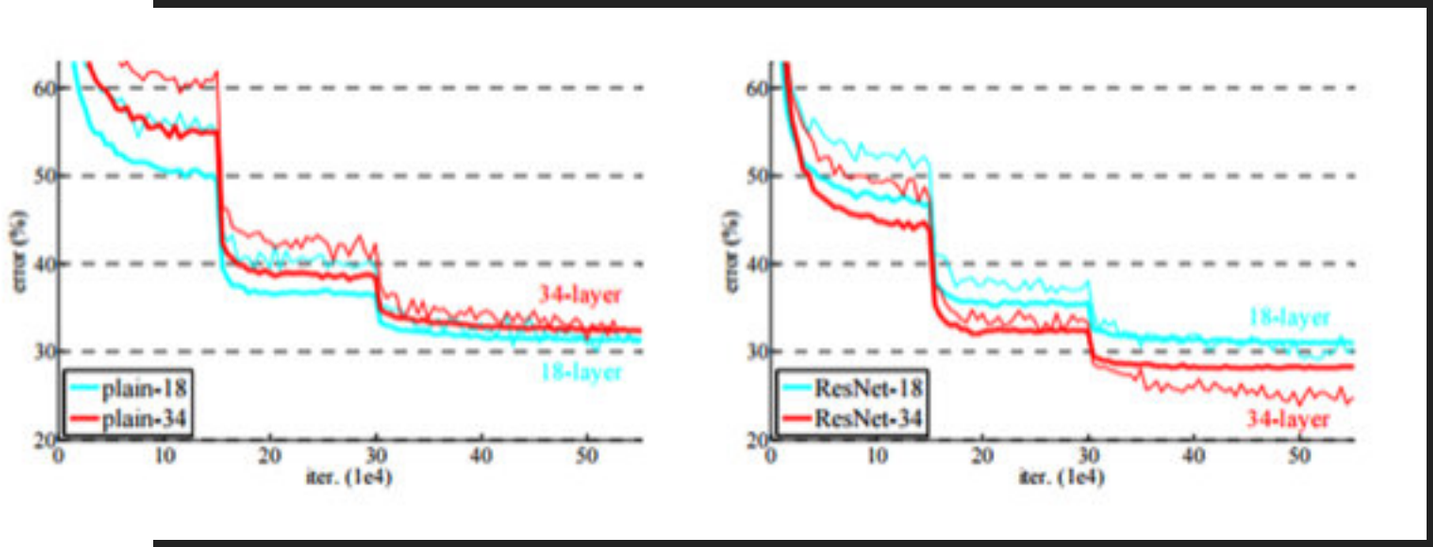
- GoogleNet 처럼 ResNet도 VGG의 설계 철학을 많이 이용했음.
  - Convolutional layer는 3 x 3
  - 그러나, 복잡도를 줄이기 위한 Max-pooling, hidden FC, DropOut은 사용하지 않음
- 출력 Feature-map 크기가 같은 경우, 해당 모든 layer는 모두 동일한 수의 filter를 갖는다.
- Feature-map의 크기가 절반으로 작아지는 경우는 연산량의 균형을 맞추기 위해 필터의 수를 두 배로 늘린다.(cf, Feature-map의 크기를 줄일 때는 Pooling을 사용하는 대신에 Convolution을 수행 할 때, stride의 크기를 "2"로 하는 방식을 취했다.
-

Comparing between Plain Network and Residual Network



- Plain Network의 경우도 VGG의 Max-pooling이나, FC와 같이 복잡도를 높이는 요소를 제거하여 연산량을 20%미만으로 줄였다.(19-layer의 VGGnet vs 34-layer)

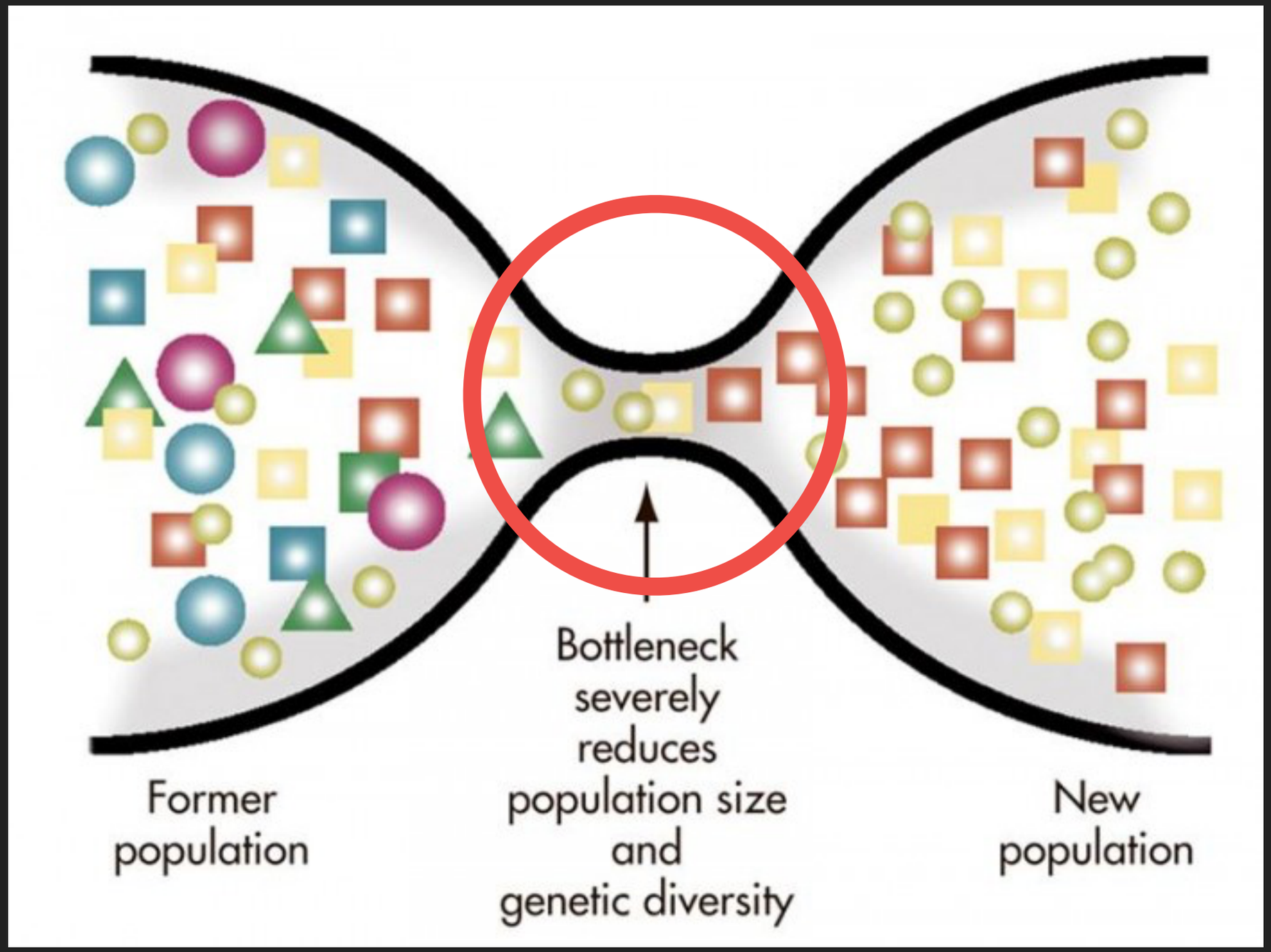
Result



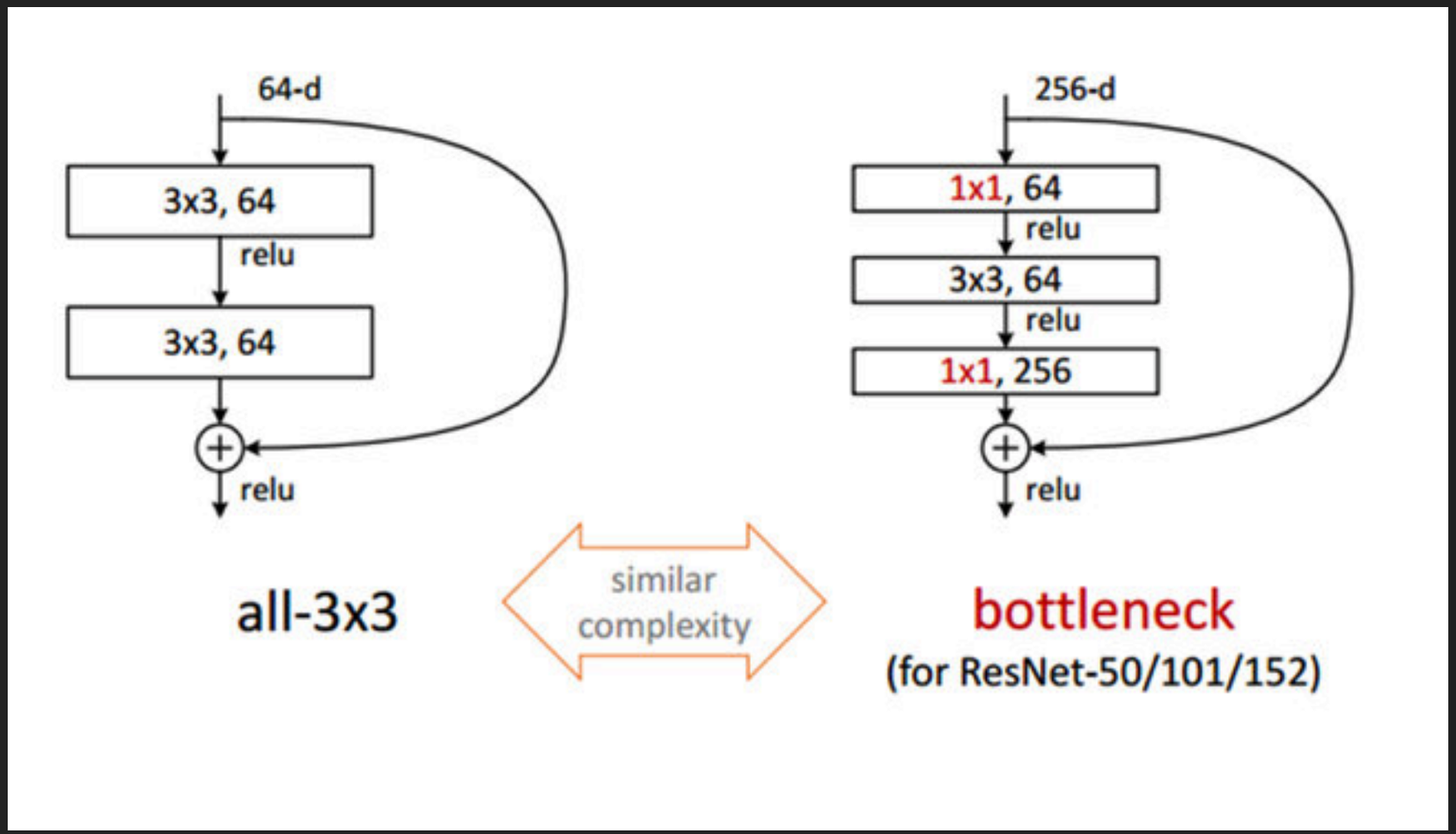
More Practice for Result

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

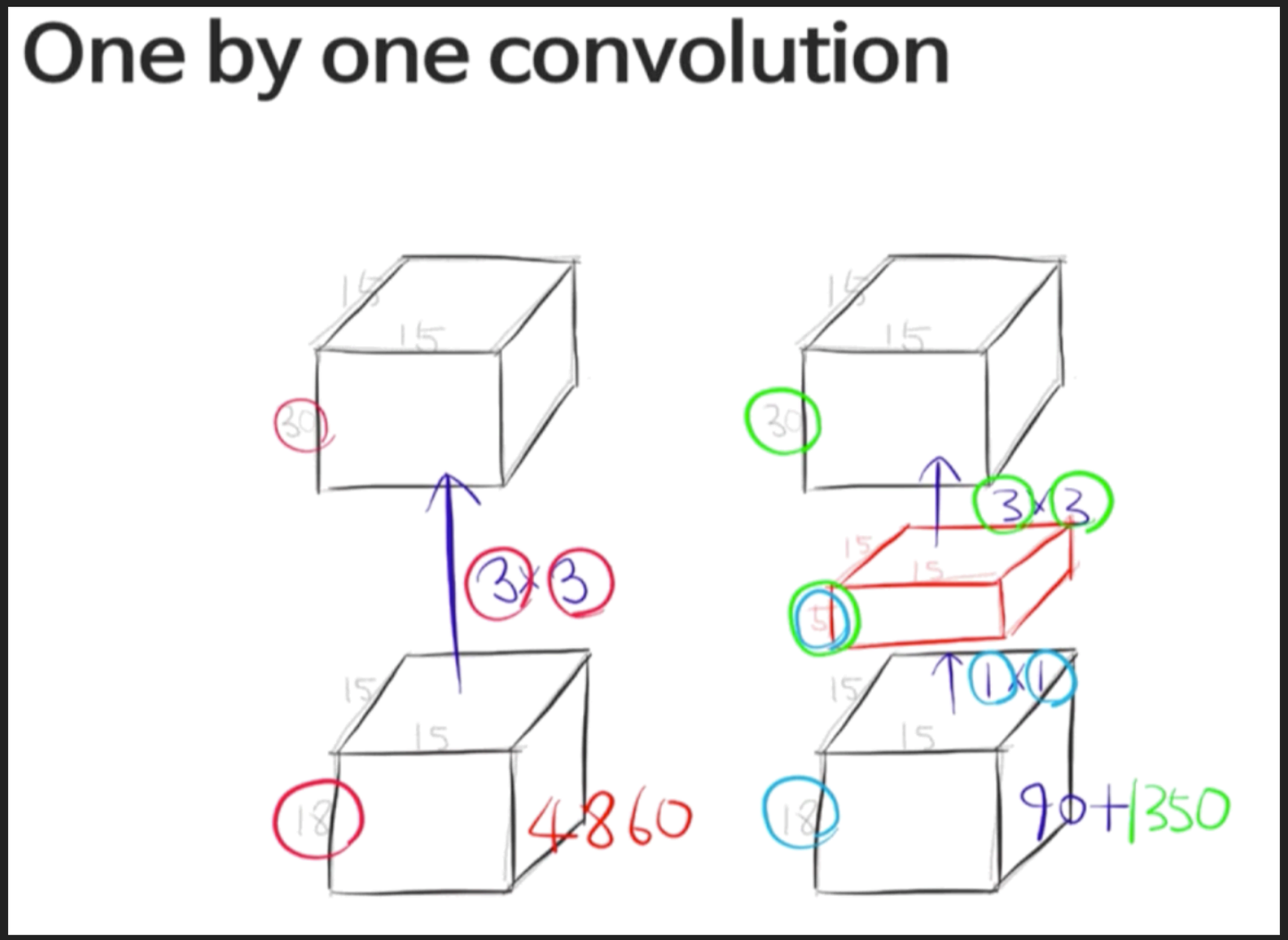
Bottleneck


Former
population
Bottleneck
severely
reduces
population size
and
genetic diversity
New
population

Deeper Bottleneck Architecture



Deeper Bottleneck Architecture

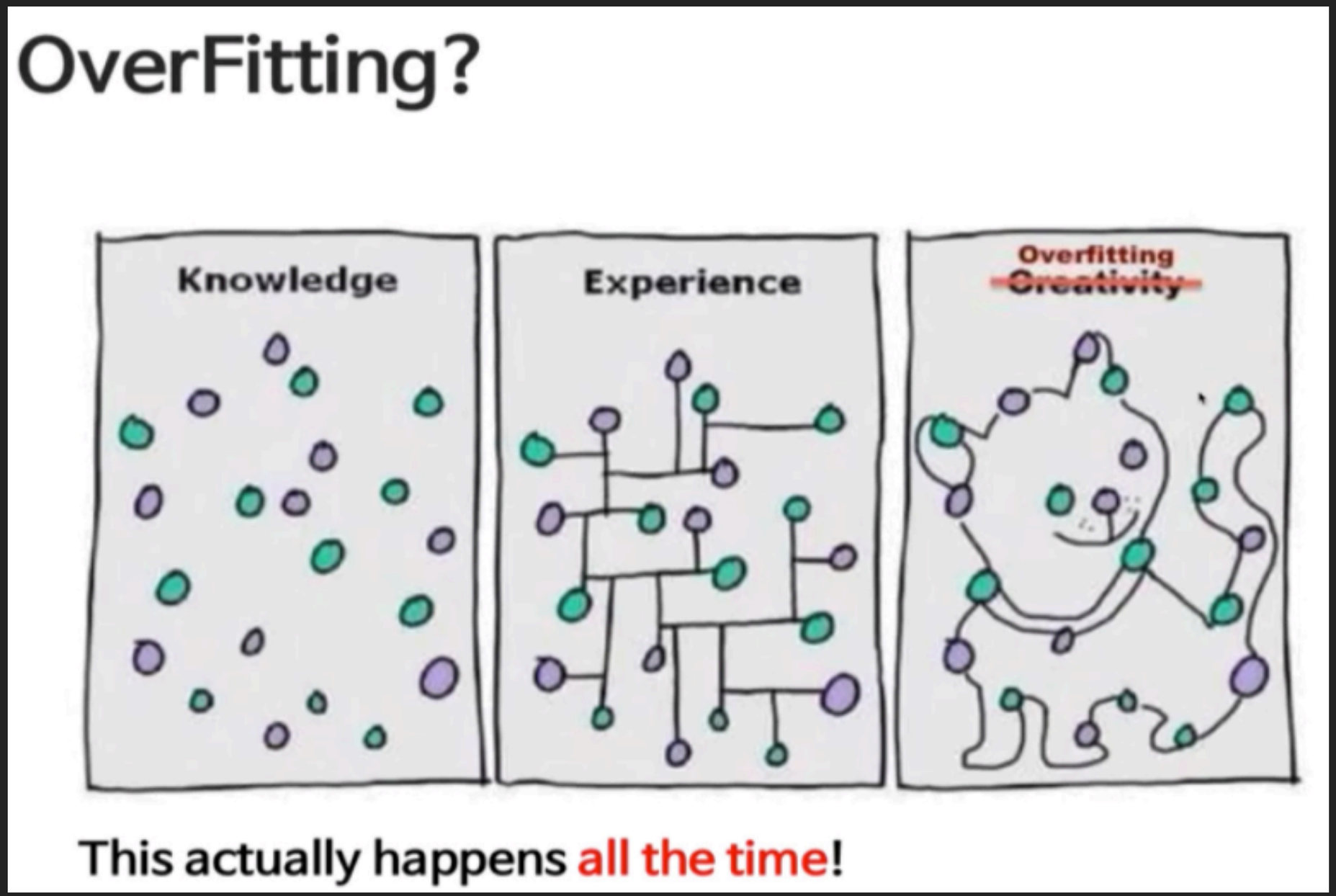


ResNet

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

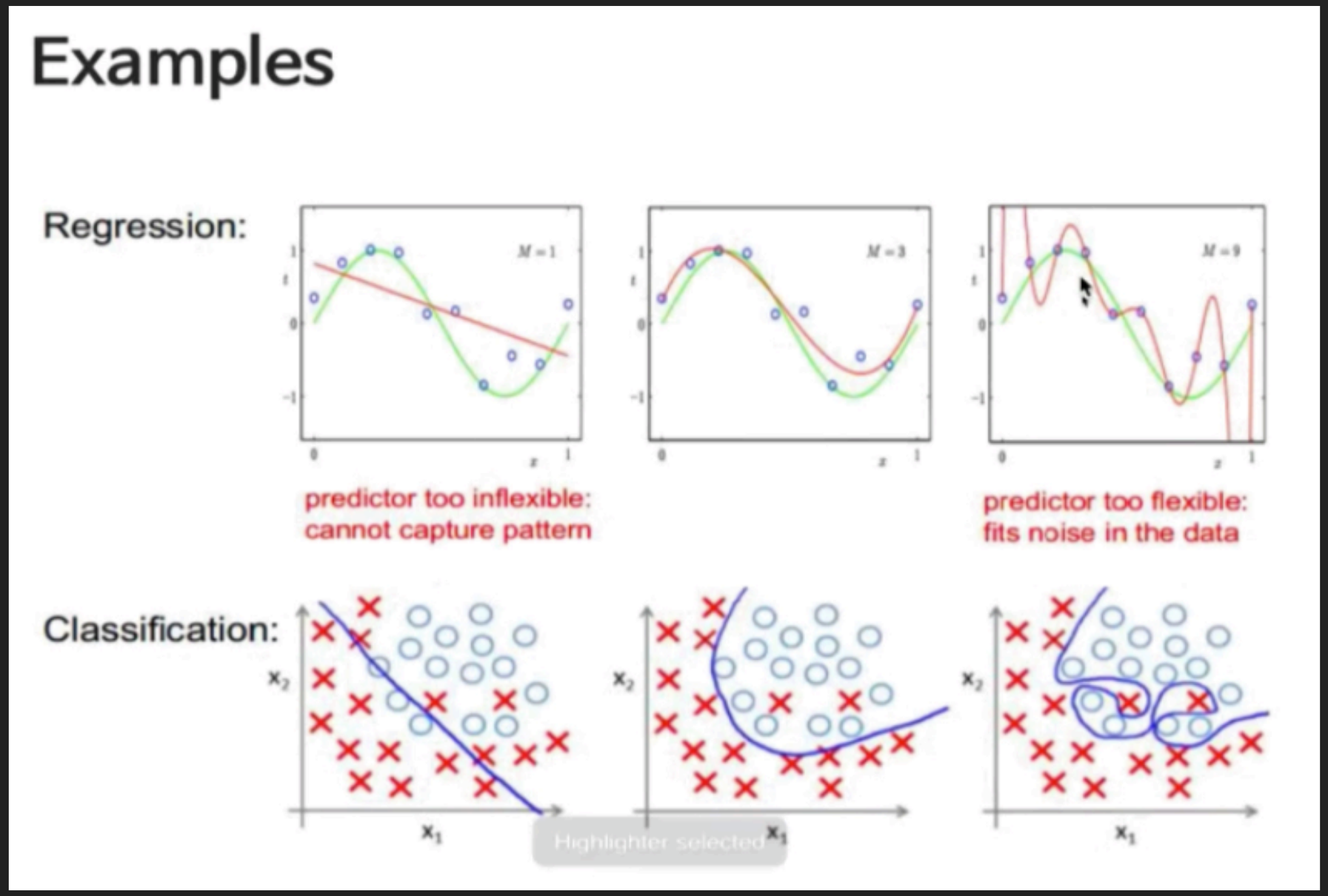
What is Regularization?

Main purpose is to avoid "Overfitting"



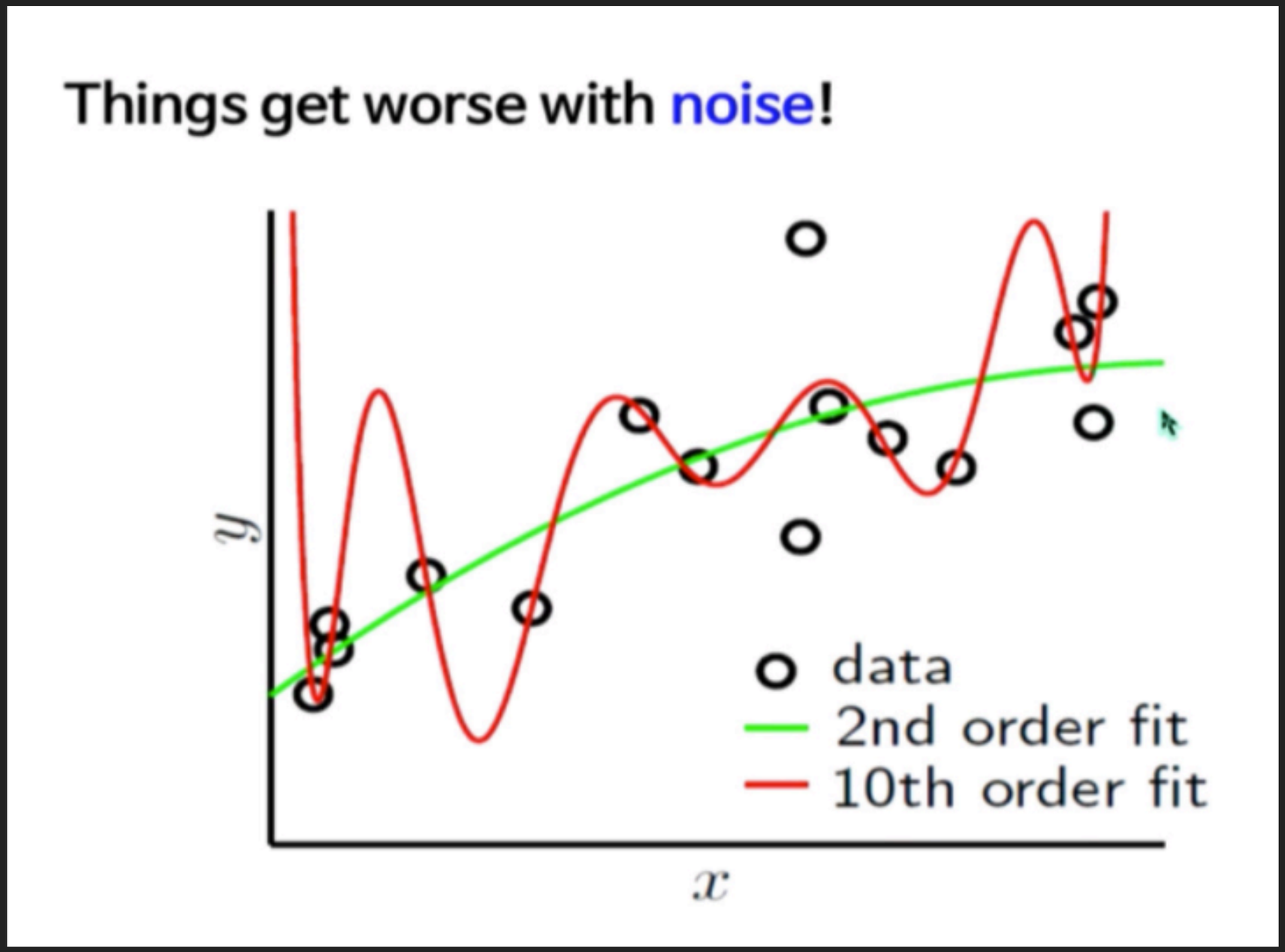
What is Regularization?

Main purpose is to avoid "Overfitting"



Overfitting

Things get worse with **noise**



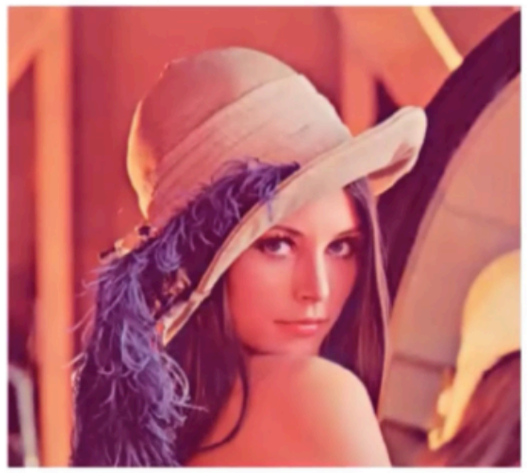
Overfitting

Things get worse with **noise**

<Can not decompose>

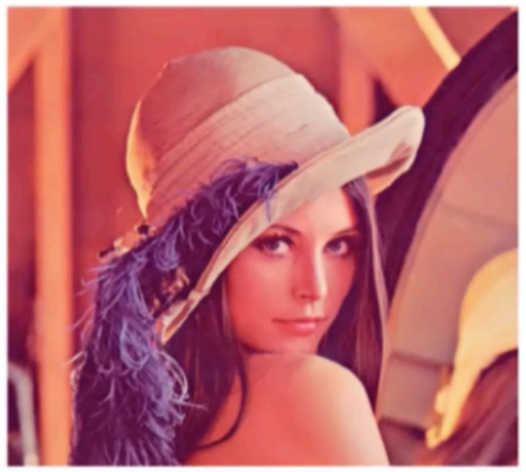
Stochastic Noise

Comes from random measurement error



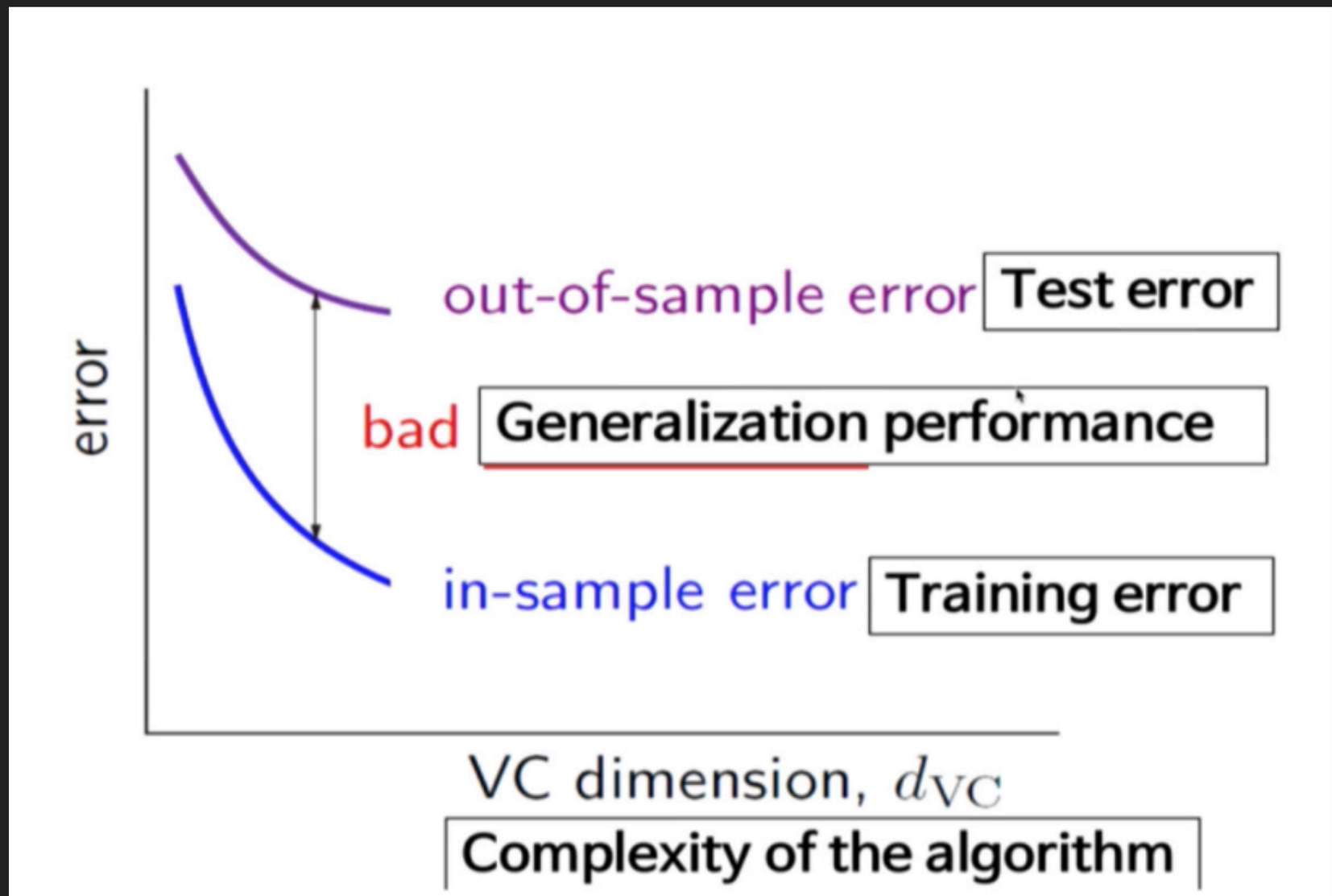
Deterministic Noise

Cannot model this type of error



Overfitting

Things get worse with **noise**



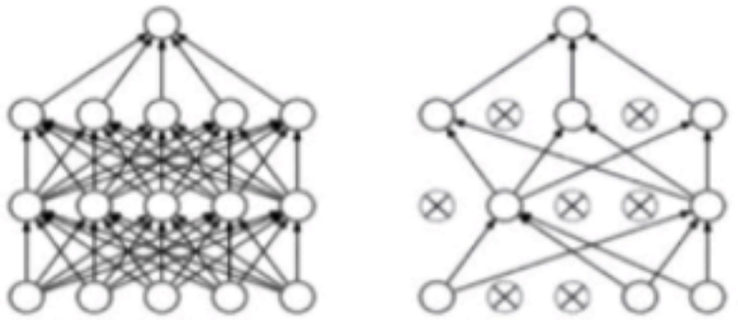
Preventing OverFitting?

Approach 1: Get more data

Approach 2: Use a model with the right capacity

Approach 3: Average many different models (Ensemble)

Approach 4: Use DropOut, DropConnect, or BatchNorm



(a) Standard Neural Net (b) After applying dropout.

Limiting the Capacity

Architecture: Limit the number of hidden layers and units per layer

Early stopping: Stop the learning before it overfits using validation sets

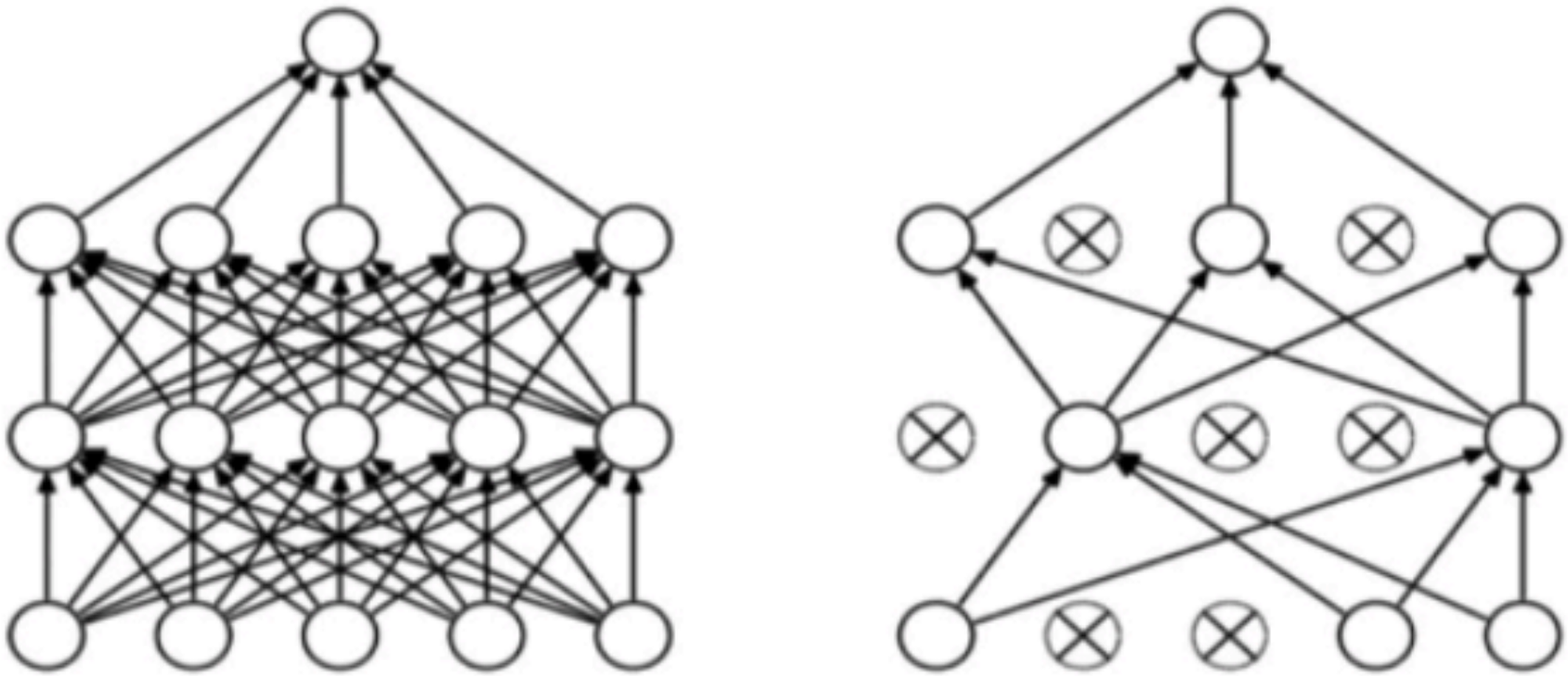
Weight-decay: Penalize large weights using penalties or constraints on their squared values (L2 penalty) or absolute values (L1 penalty)

Limiting the Capacity

Weight Decay

$$E(w) = E_0(w) + \frac{1}{2}\lambda \sum_i w_i^2$$

DropOut



(a) Standard Neural Net

(b) After applying dropout.

DropOut increases the generalization performance of the neural network by restricting the model capacity!

Batch Normalization

배치(Batch) 정규화(Normalization)는 각 Affine layer를 통과한 미니배치 단위 output을 표준정규분포로 정규화하는 작업

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Benefits of BN

1. Increase learning rate
2. Remove Dropout
3. Reduce L2 weight decay
4. Accelerate learning rate decay
5. Remove Local Response Normalization

MORE

Ian Goodfellow http://www.deeplearningbook.org

Book review

1. Parameter Norm Penalties
2. Dataset Augmentation
3. Noise Robustness: to input, weights, and output
4. Semi-Supervised Learning = learning a representation
5. Multitask learning
6. Early Stopping
7. Parameter Tying and Parameter Sharing
8. Sparse representation
9. Bagging and Other Ensemble Methods
10. Dropout
11. Adversarial Training

Ian Goodfellow http://www.deeplearningbook.org

1. Parameter Norm Penalties

Many regularization approaches are based on limiting **the capacity** of models by adding a parameter norm penalty to the objective function

- L2 weight decay(제곱값)
- L1 weight decay(절대값)

2. Dataset Augmentation

- 기계 학습 모델에서 데이터를 많이 모으는 것이 중요
- 데이터가 없다면 Fake data를 만들어서 학습 데이터를 늘리자
- Label-preserving transformation (좌우 상하 반전, 축소 등의)
- 인위적인 Noise를 넣어 학습시키는 방법
- Subsampling을 통해 데이터를 늘림

Ian Goodfellow http://www.deeplearningbook.org

3. Noise Robustness

Noise injection can be much **more powerful** then simply shrinking the parameters, especially when the noise is added to the hidden unit

Another way that noise has been used is by adding it to the **weights**

Most datasets have some amount of mistakes in the y labels, **Label-smoothing** can be used in this regard

4. Semi-Supervised Learning

In the paradigm of semi-supervised learning, both unlabeled examples and labeled examples are used to.

In the context of deep learning, semi-supervised learning usually refers to learning a representation

Ian Goodfellow http://www.deeplearningbook.org

5. Multi-Task Learning

Multi-task learning is a way to improve generalization by pooling the examples arising out of several tasks

From the point of view of deep learning, the underlying prior belief is the following:
"Among the factors that explain the variations observed in the data associated with the different tasks, some are shared across two of more tasks"

6. Parameter Tying and Parameter Sharing

A common type of dependency that we often want to express is that certain parameters should be close to one another

- Parameter Tying
  Consider the following scenario: we have two models performing the same classification task(with the same set of classes) but with **different input distributions**

- Parameter Sharing
  By far the most popular and extensive used of parameter sharing occurs in **convolutional neural networks(CNNs)**

MORE

Ian Goodfellow http://www.deeplearningbook.org

7. Sparse Representations

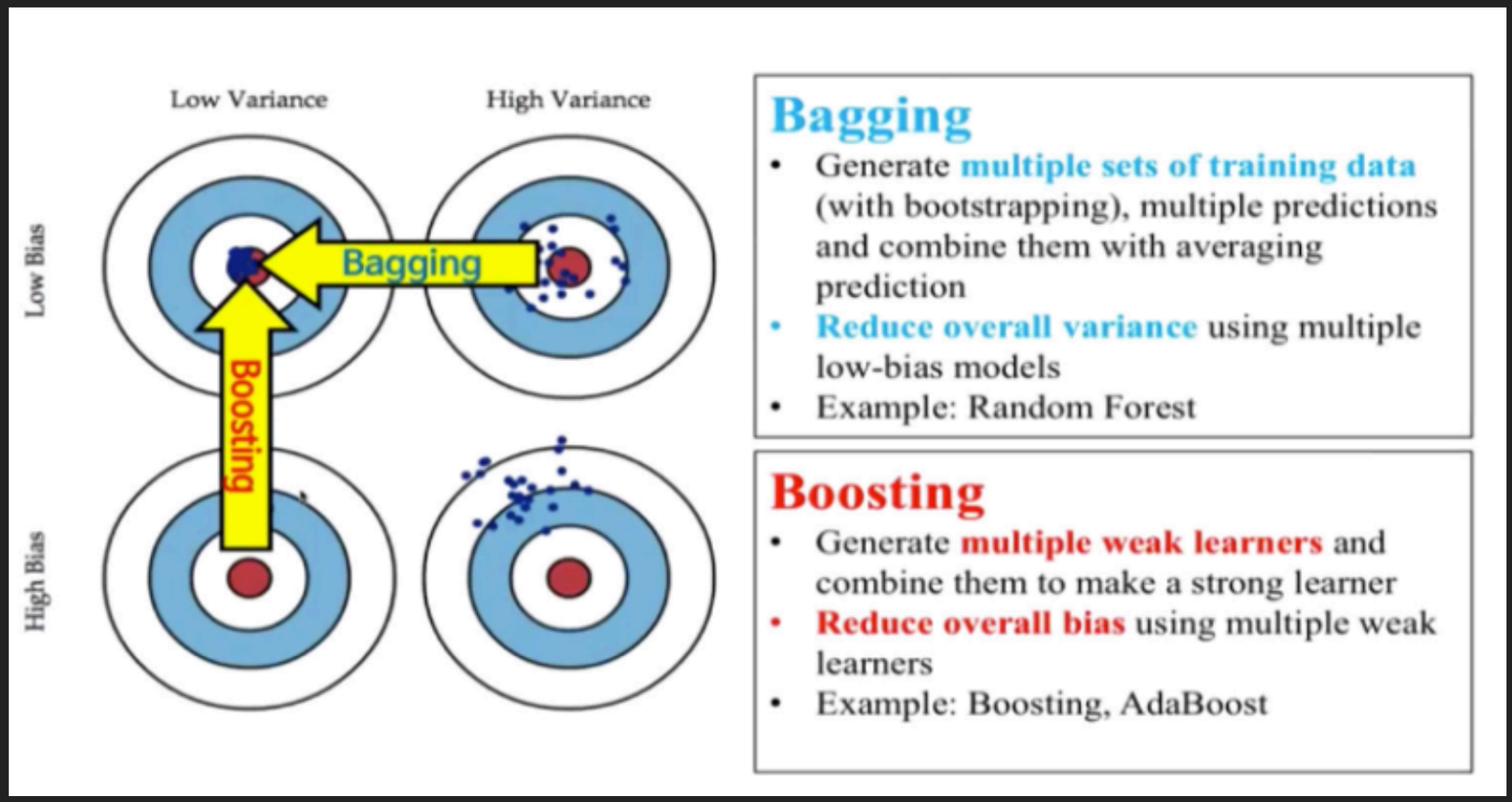
Representational sparsity describes a representation where many of elements are zero.

- Sparse weights(L1 decay):

- Sparse activations

Norm penalty regularization of representations is performed by adding to the loss function a norm penalty on the **representations**

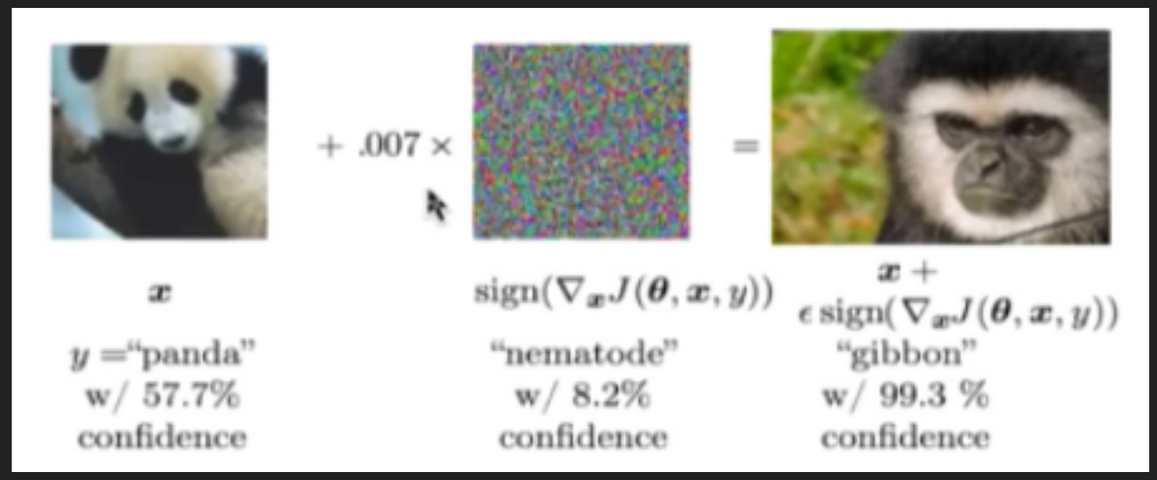
8. Bagging and Other Ensemble Methods



Ian Goodfellow http://www.deeplearningbook.org

9. Adversarial Training

In many cases, neural networks have begun to reach human performance.
Examples that a human connot tell the the difference results in highly different predictions. These examples are called the **adversarial examples**



Adversarial training is done via **virtual adversarial examples**