심층신경망 & 합성곱신경망

강필성
고려대학교 산업경영공학부
Bflysoft & WIGO AI LAB

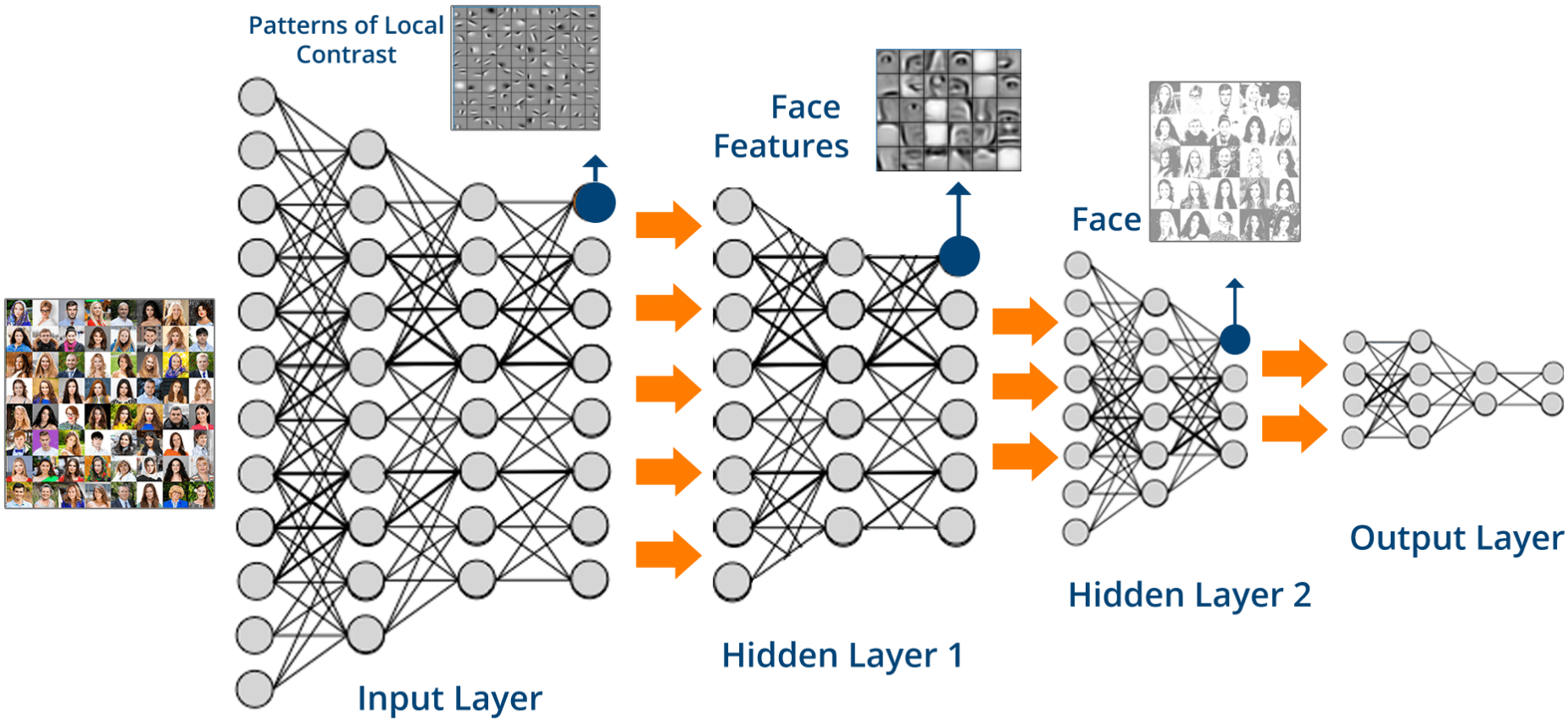
AGENDA

01 심층 신경망: Deep Neural Network

02 합성곱 신경망: Convolutional Neural Network

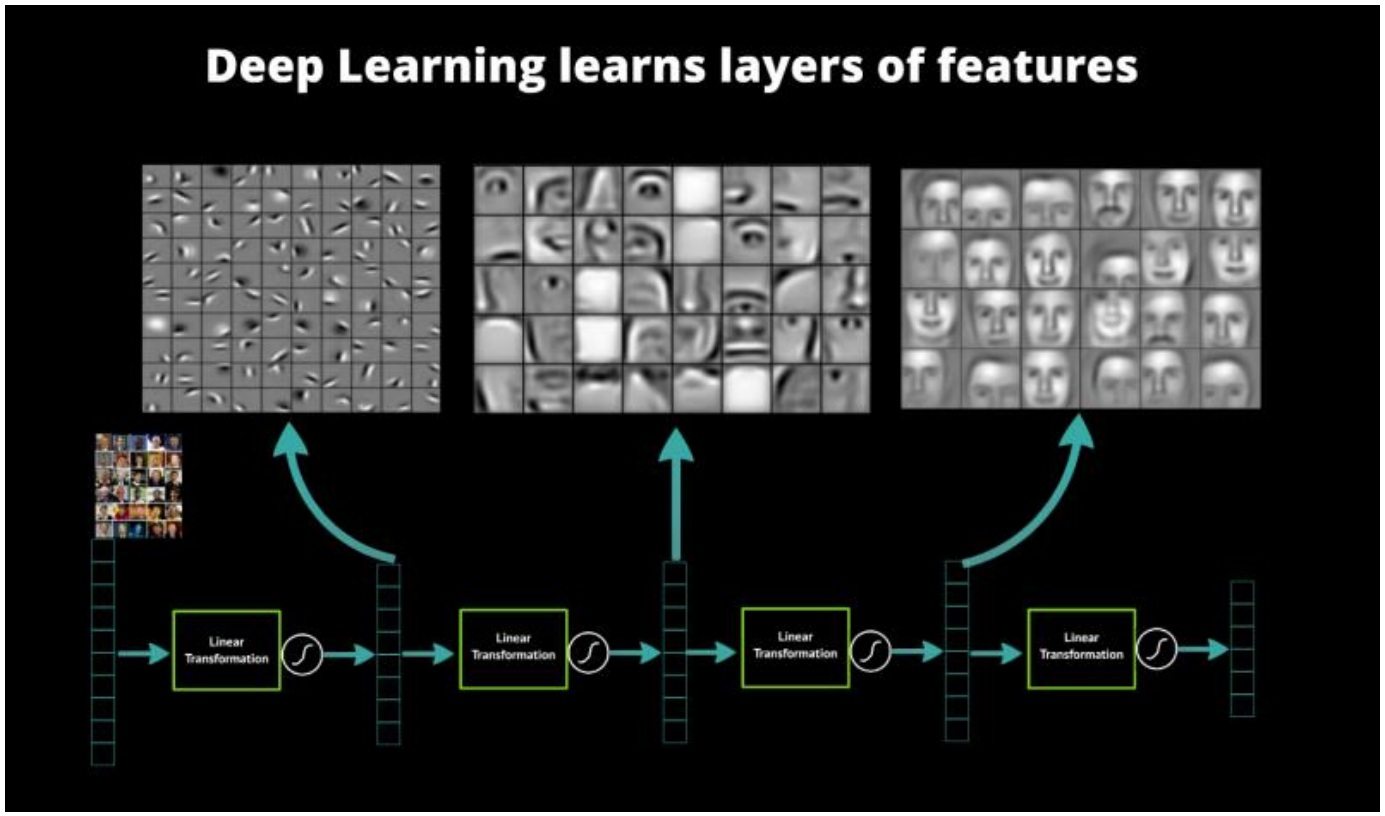
심층 신경망: Deep Neural Networks (DNN)

- Deep Neural Network (DNN)
  - ✓여러 층으로 이루어진 복잡한 구조의 인공신경망



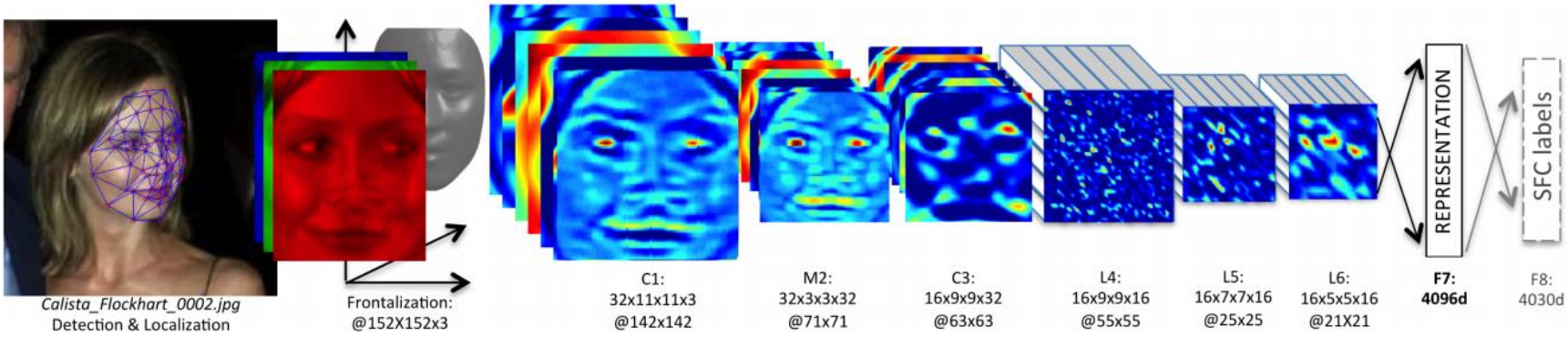
심층 신경망: Deep Neural Networks (DNN)

- 왜 DNN인가?
  - ✓ 고차원의 데이터(이미지, 음성, 텍스트 등)에 대해 속성을 정의하지 않고 원 데이터 (Raw Data)를 그대로 사용해도 알아서 속성을 판별할 수 있음



심층 신경망: Deep Neural Networks (DNN)

- 왜 DNN인가?
  - ✓기존의 기계학습 방법론에 비해 우수한 예측 정확도
    - 특정 분야(음성, 이미지 인식 등)에서는 (인간의 능력과 유사할 정도로) 월등히 우수한 성능을 나타내기도 함



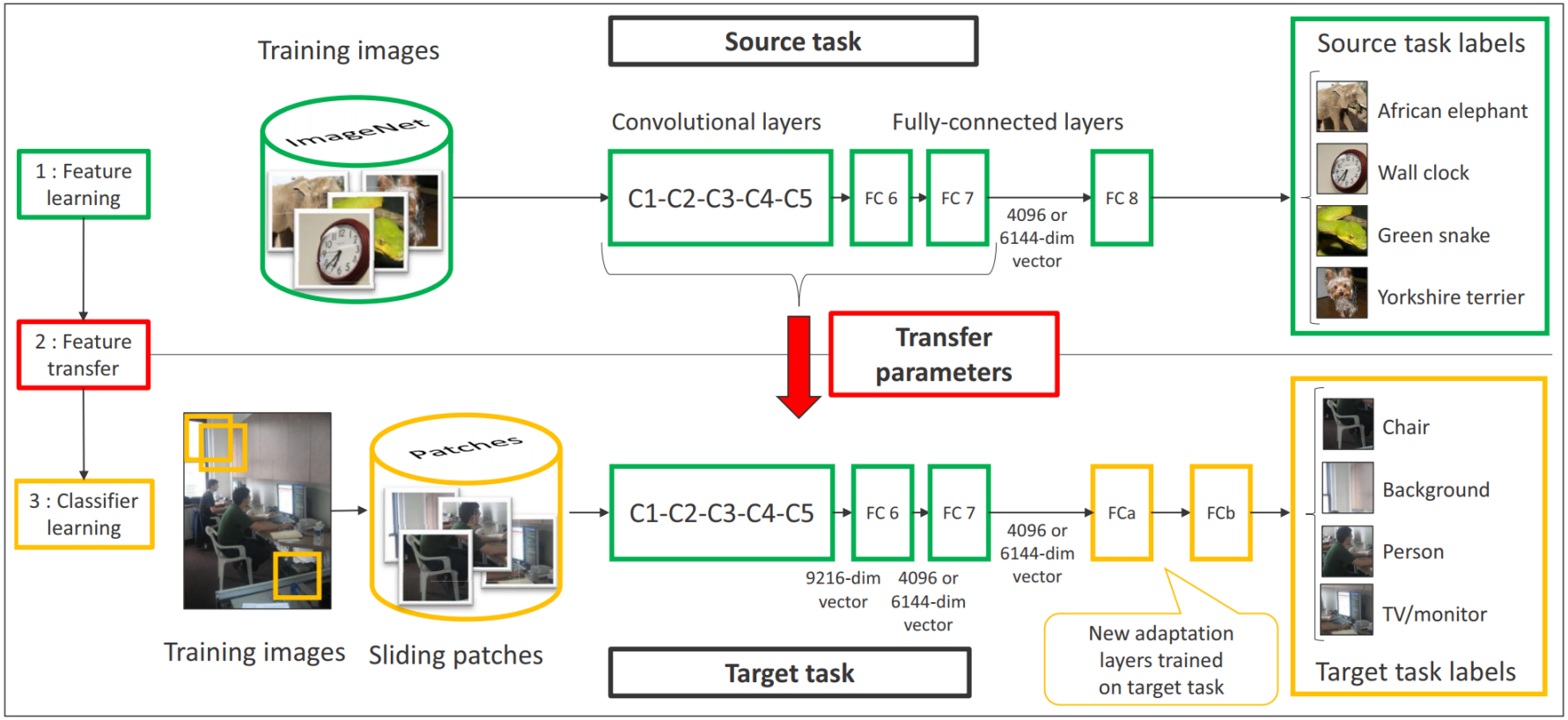
Taigman, Y. et al. (2014). DeepFace: Closing the Gap to Human-Level Performance in Face Verification, CVPR'14.

Recognition Accuracy for Labeled Faces in the Wild (LFW) dataset (13,233 images, 5,749 people)

- Human: 95% vs. DeepFace in Facebook: 97.35%

심층 신경망: Deep Neural Networks (DNN)

- 왜 DNN인가?
  - ✓특정 데이터(도메인)에서 학습된 구조를 다른 데이터(도메인)에서도 사용할 수 있음 (Transfer Learning)



Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1717-1724).

심층 신경망: Deep Neural Networks (DNN)

- 왜 DNN인가?
  - ✓지금까지 사람은 쉽게 잘 하나 기계가 잘 못하던 과업(Task)들에 대해서 진일보한 성능을 보여줌 (음성인식, 시각인지, 언어인지 등)

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

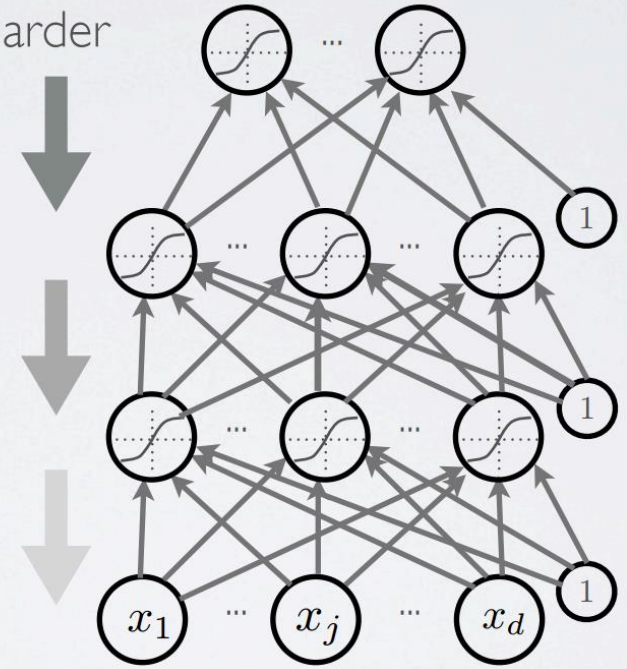
Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... & Bengio, Y. (2015, June). Show, attend and tell: Neural image caption generation with visual attention. In International Conference on Machine Learning (pp. 2048-2057).

심층 신경망: Deep Neural Networks (DNN)

- 과거에는 무엇이 문제였는가? I) 학습이 잘 안됨!
  - ✓ Vanishing gradient problem: 은닉층의 수가 증가함에 따라 하위 층으로 갈수록 Gradient가 0이 될 가능성이 높아져서 Error Backpropagation이 잘 되지 않음

Topics: why training is hard

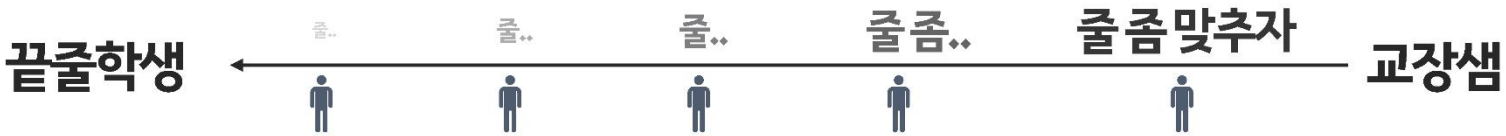
- First hypothesis: optimization is harder (underfitting)
  - vanishing gradient problem
  - saturated units block gradient propagation
- This is a well known problem in recurrent neural networks

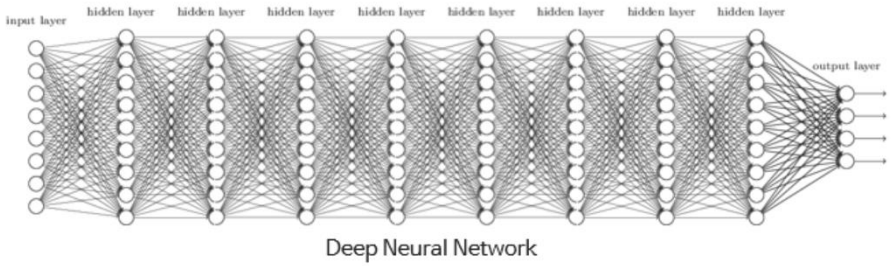


https://dl.dropboxusercontent.com/u/19557502/7_02_difficulty_of_training.pdf

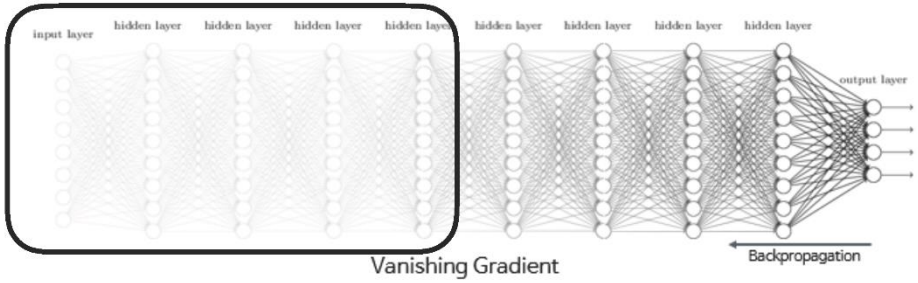
심층 신경망: Deep Neural Networks (DNN)

Vanishing gradient 현상 : 레이어가 깊을 수록 업데이트가 사라져간다.
그래서 fitting이 잘 안됨(underfitting)





학습이 잘 안됨



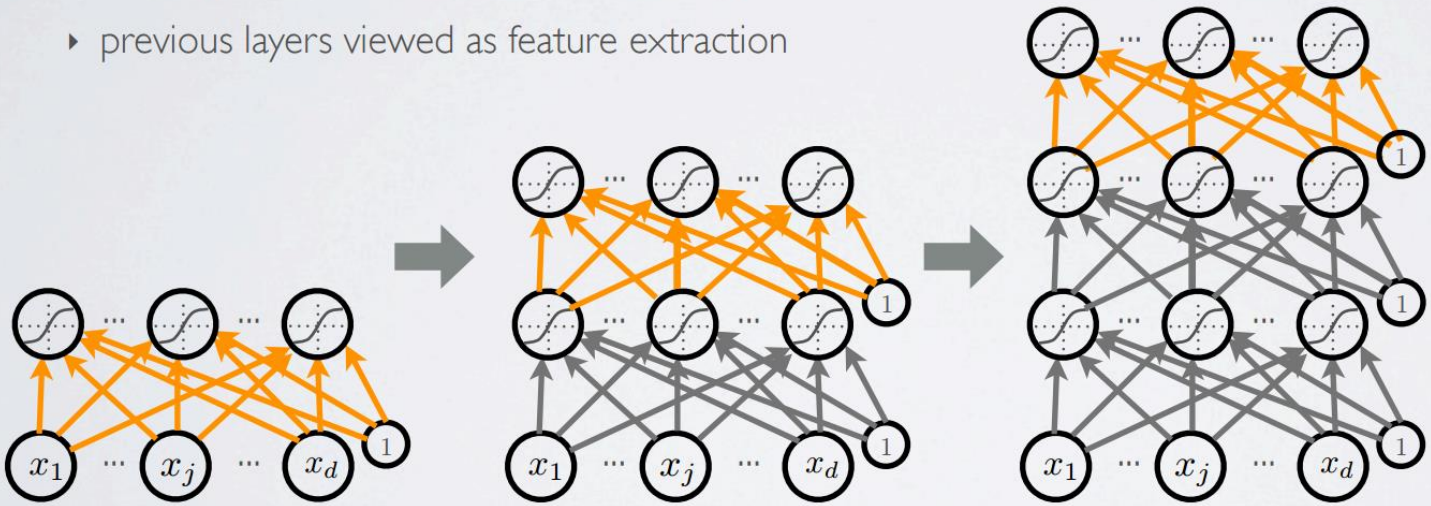
하용호, 백날 자습해도 이해 안가던 딥러닝, 머리속에 인스톨 시켜드립니다

심층 신경망: Deep Neural Networks (DNN)

- 해결책 1-1: 층별로 차근차근 학습을 시켜보자
  - ✓ 벽돌집을 쌓을 때 벽돌을 한 줄씩 쌓아가듯이…
  - ✓ Deep Learning의 대부 Hinton 교수님이 처음 성공 → 근데 지금은 잘 안씀

Topics: unsupervised pre-training

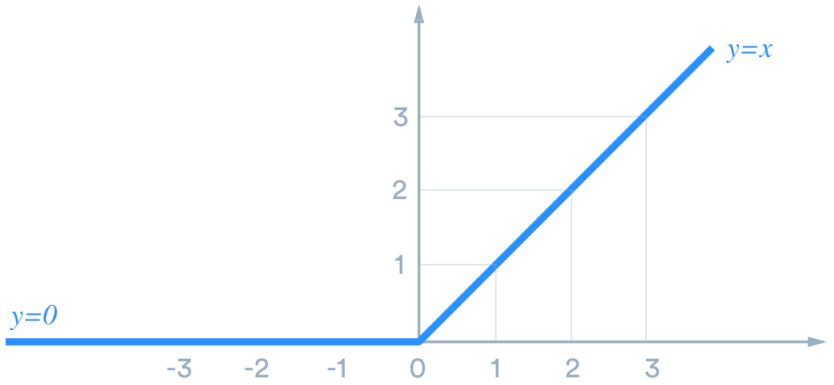
- We will use a greedy, layer-wise procedure
  - train one layer at a time, from first to last, with unsupervised criterion
  - fix the parameters of previous hidden layers
  - previous layers viewed as feature extraction

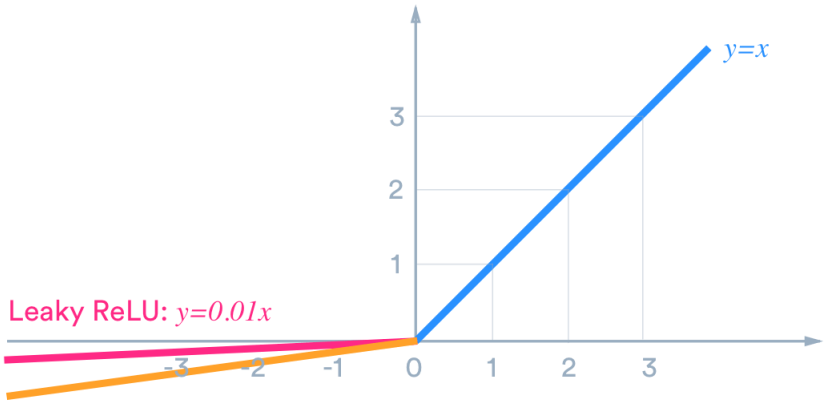


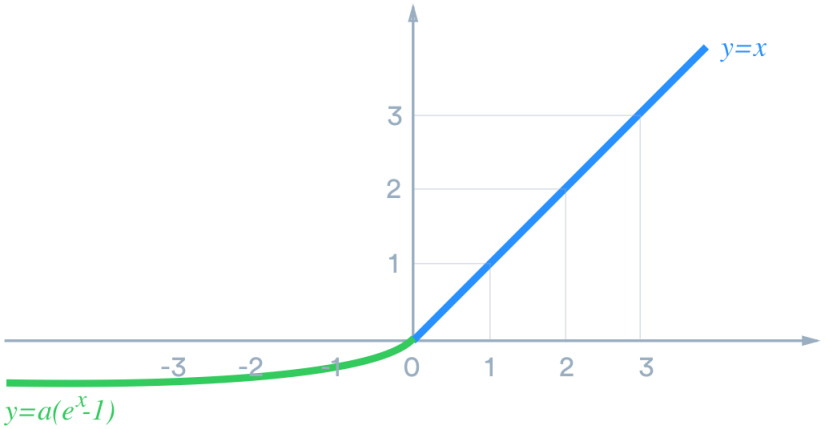
https://dl.dropboxusercontent.com/u/19557502/7_03_unsupervised_pretraining.pdf

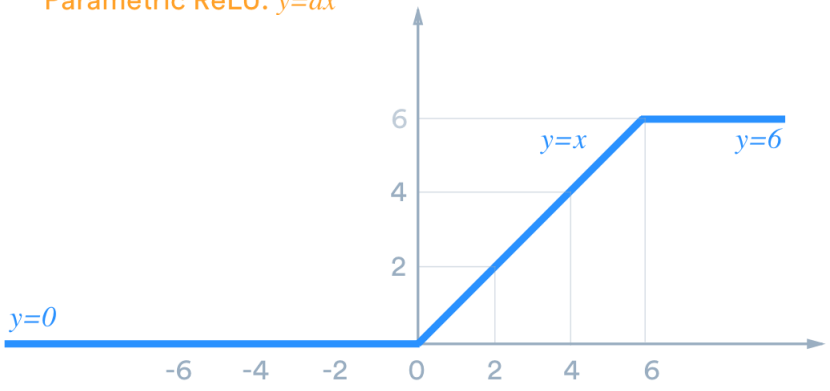
심층 신경망: Deep Neural Networks (DNN)

- 해결책 1-2: Gradient가 멀리까지 가도록 새로운 활성 함수를 쓰자
  - ✓ Rectified Linear Unit (ReLU) 사용 → 요즘에는 거의 모든 DNN은 ReLU 및 ReLU를 변형한 활성 함수 사용





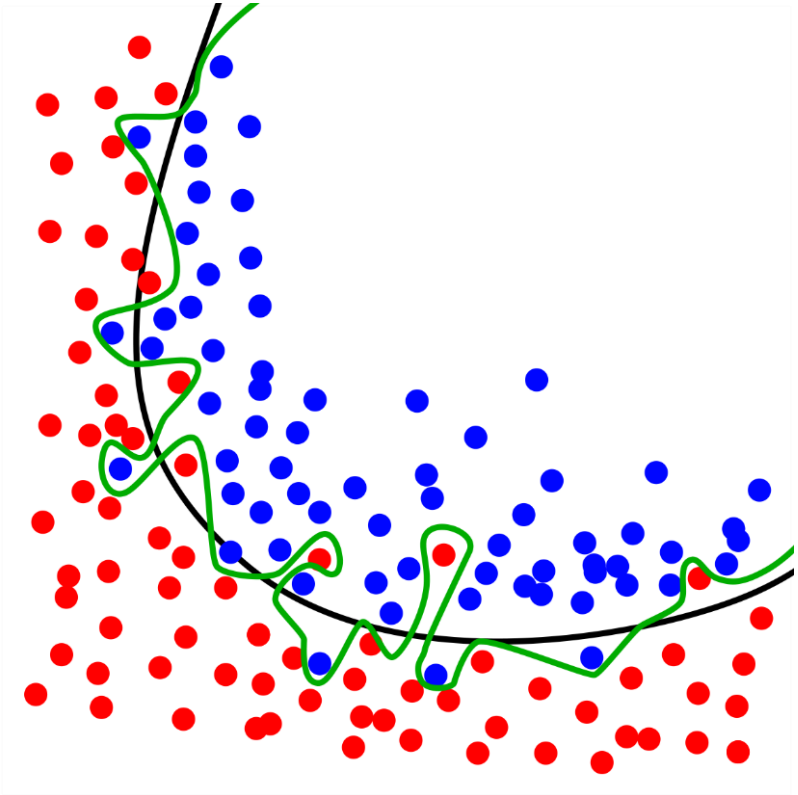




https://www.tinymind.com/learn/terms/relu

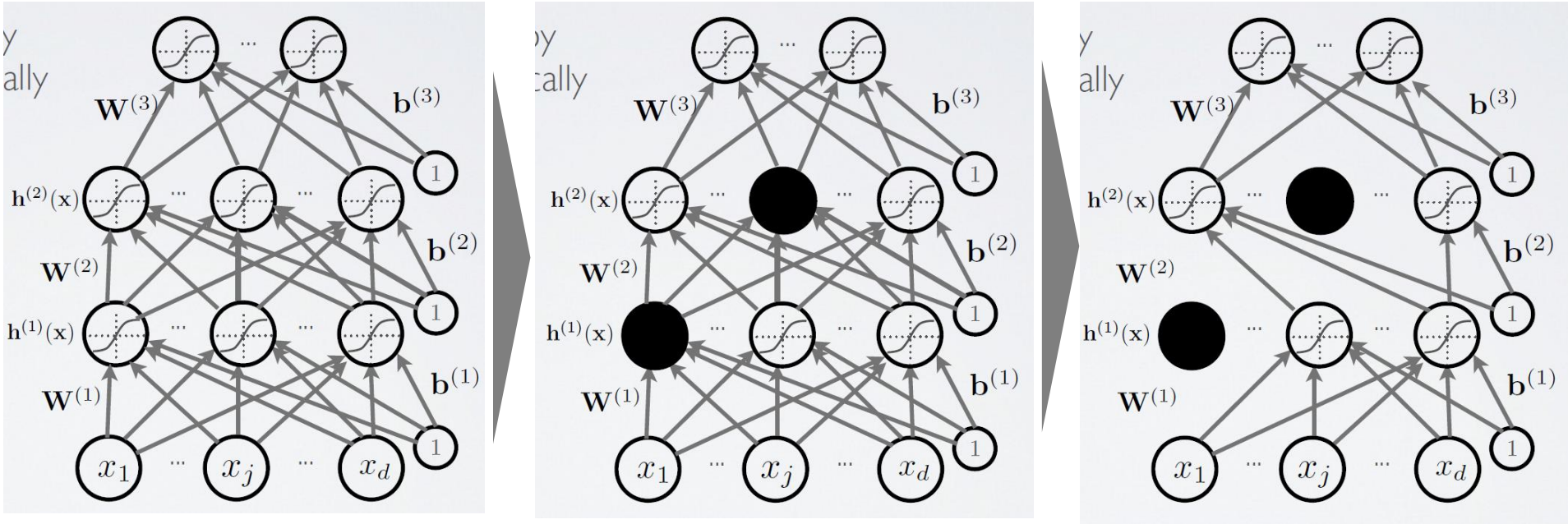
심층 신경망: Deep Neural Networks (DNN)

- **과거에는 무엇이 문제였는가?** 2) 구조가 너무 복잡하여 과적합 위험이 있음
  - ✓ 은닉층과 은닉노드의 수 증가로 어떤 형태의 함수(회귀)나 분류 경계면(분류)도 찾아낼 가능성이 있음
  - ✓ 데이터의 숫자가 적을 때는 이 특징이 꼭 좋은 것만은 아님
    - 검은색을 찾으랬더니…
    - 초록색을 찾고 있네…



심층 신경망: Deep Neural Networks (DNN)

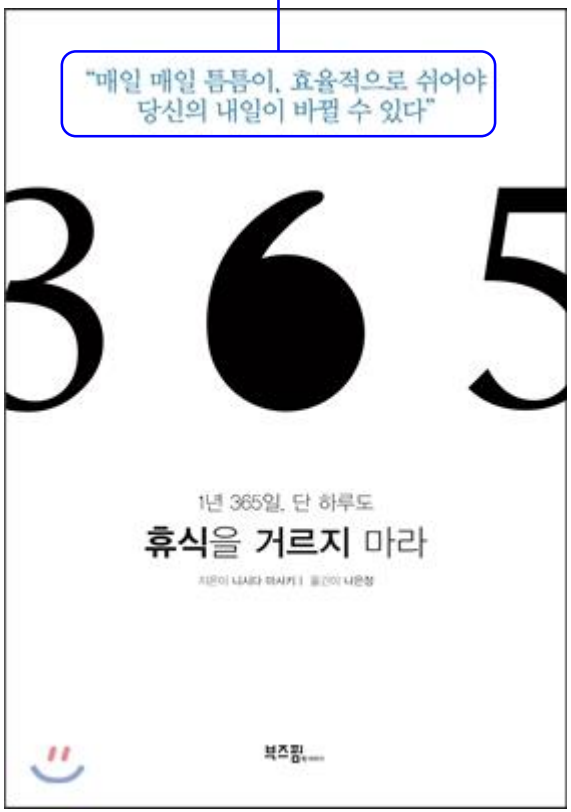
- 해결책 2: Drop-Out
  - ✓부분 최적화의 합 ≠ 전체 최적화
  - ✓학습 과정에서 일부러 특정 노드들에 연결된 가중치들은 업데이트를 하지 말자
  - ✓어떤 노드를 업데이트 시키지 않을 것인가는 무작위로 그때그때 바꿔가며 결정하자



심층 신경망: Deep Neural Networks (DNN)

- 해결책 2: Drop-Out
  - ✓부분 최적화의 합 ≠ 전체 최적화
  - ✓우리네 일상도 별반 다르지 않습니다…

사람도 쉬고, hidden node도 쉬고…

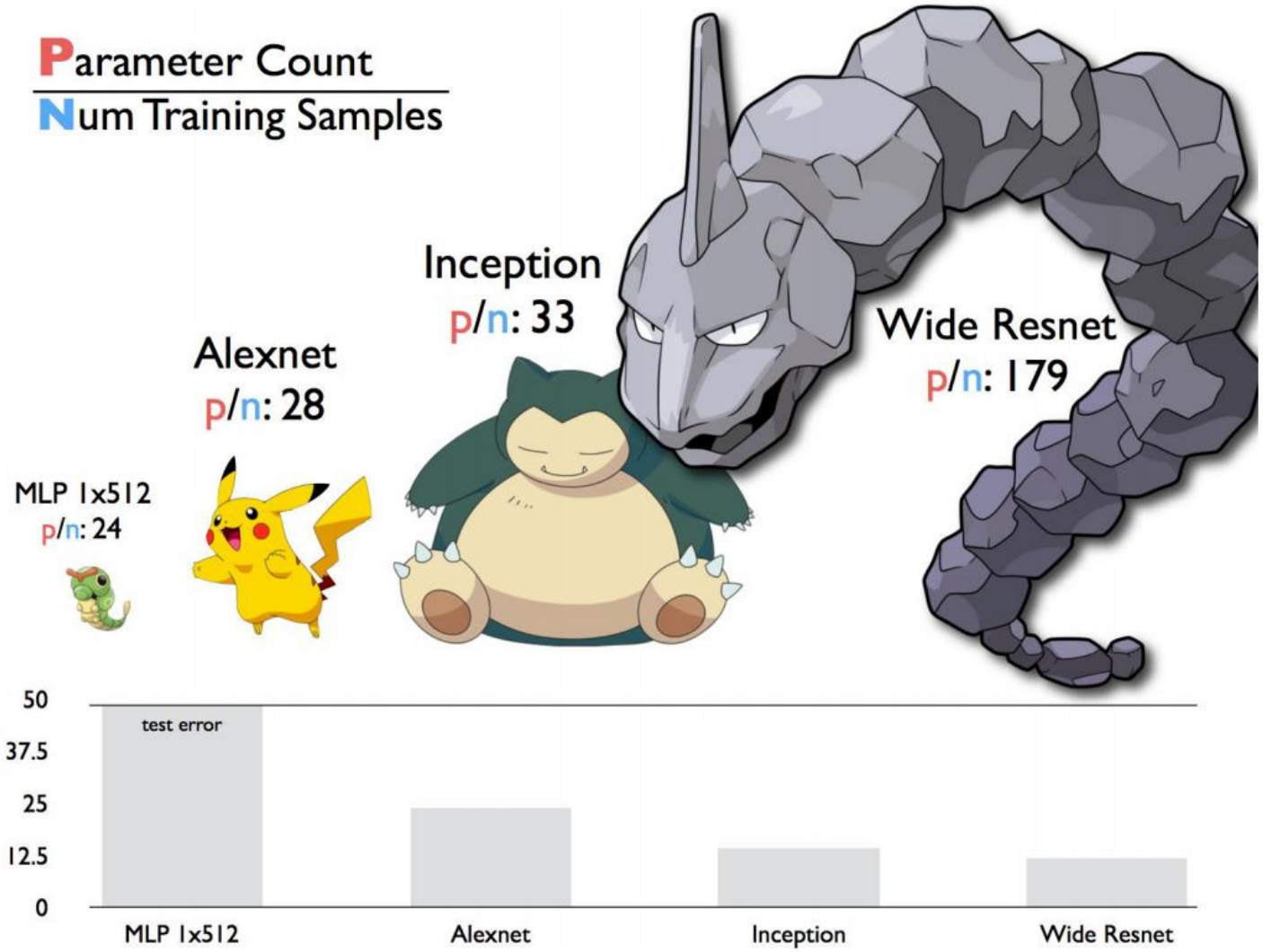


http://www.yes24.com/24/goods/13144172

https://www.superookie.com/contents/58d3d99f8b129f44aa761811

심층 신경망: Deep Neural Networks (DNN)

- 과적합 방지에 성공한 High Capacity 모델의 효과



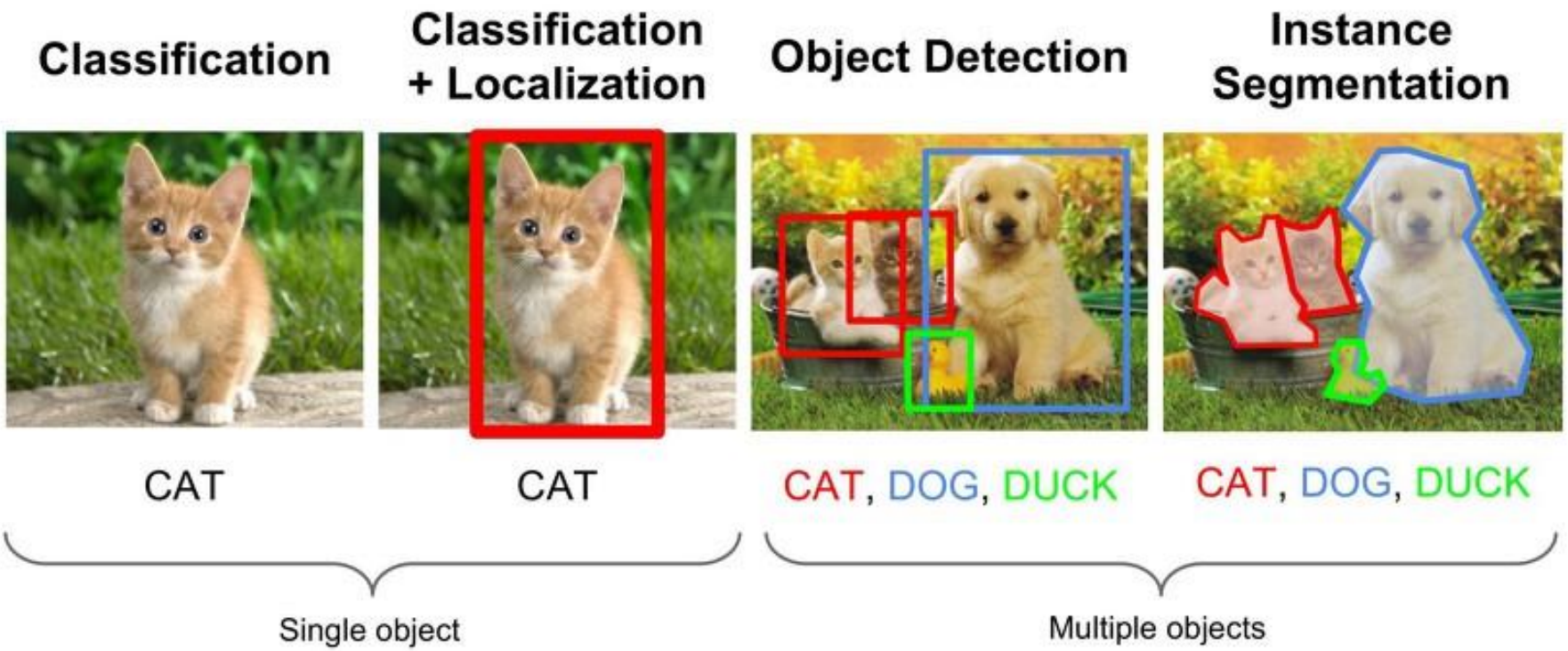
AGENDA

01 심층 신경망: Deep Neural Network

02 합성곱 신경망: Convolutional Neural Network

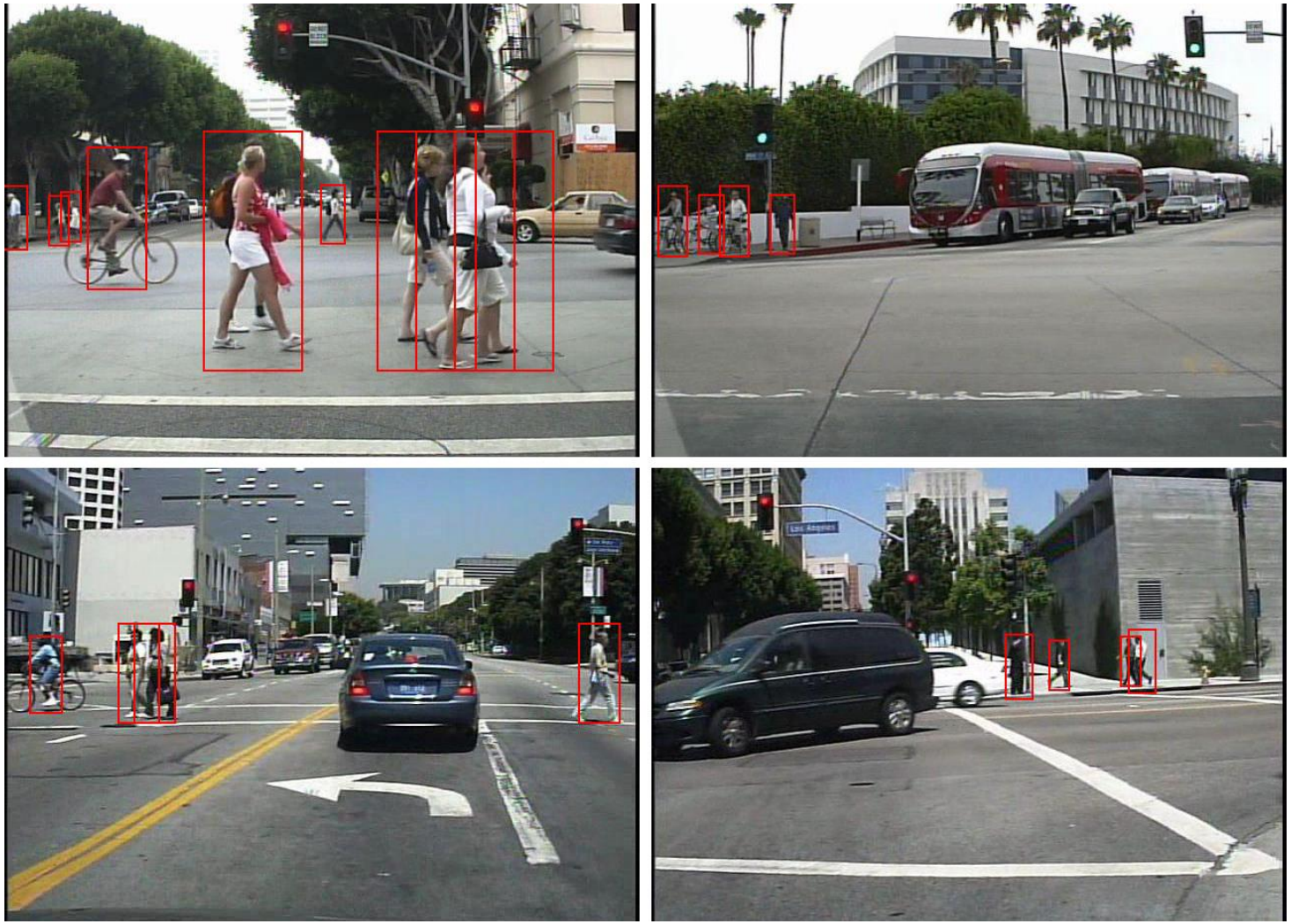
Convolutional Neural Network (CNN)

- **합성곱 신경망**
  - ✓ Convolution 연산을 통해 이미지로부터 필요한 특질(feature)을 스스로 학습할 수 있는 능력을 갖춘 심층 신경망 구조
- **이미지 인식 종류**



Convolutional Neural Network (CNN)

- 일반적 활용 사례: 보행자 인식 (자율 주행 자동차의 안전한 운행을 위해 필요)



Convolutional Neural Network (CNN)

- 일반적 활용 사례: 얼굴 및 표정 인식

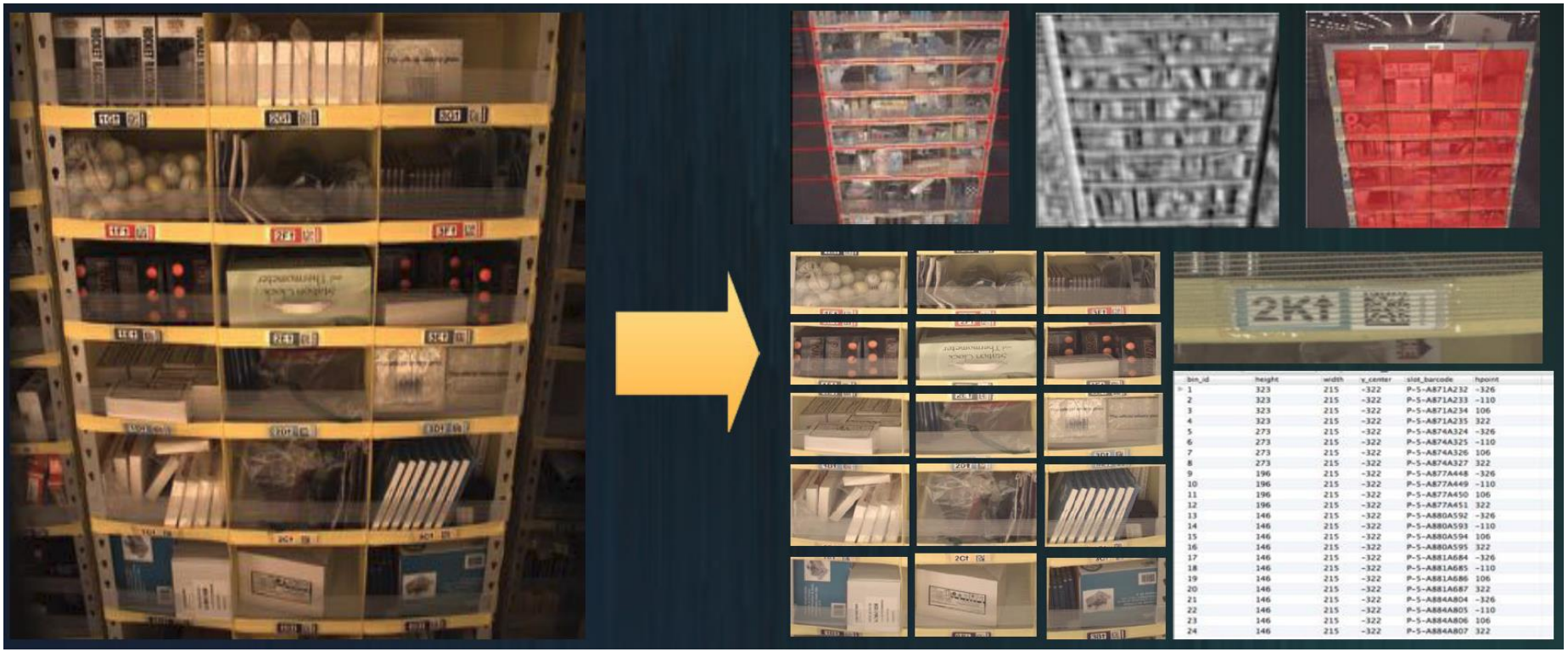
누가 지각/결석을 했지?

수업이 얼마나 따분한가?

출석을 부르며 수업 시간을 때울 수 있었던 즐거움이 사라지는 건 비밀…

Convolutional Neural Network (CNN)

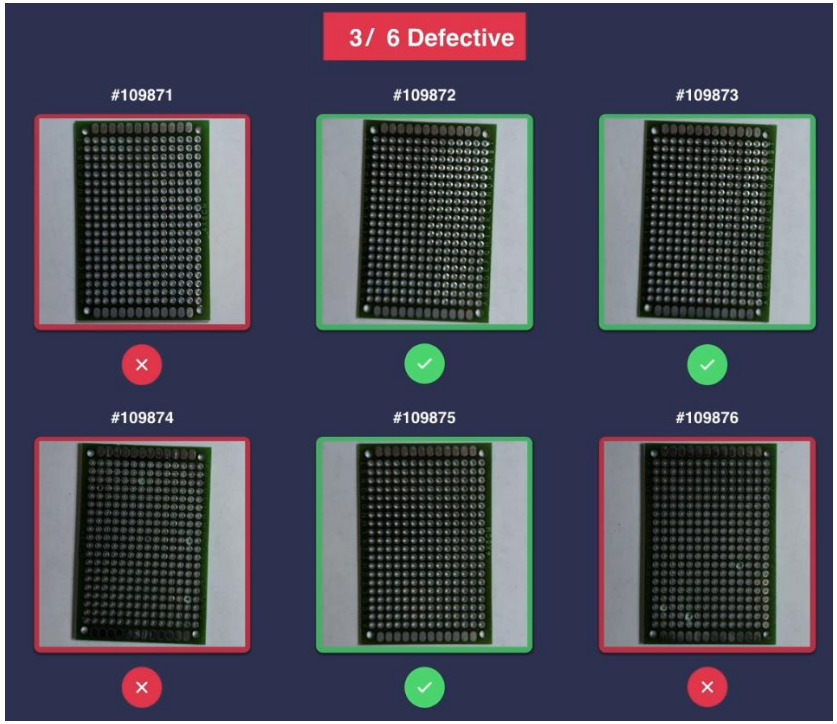
- 비즈니스 활용 사례: 재고 파악

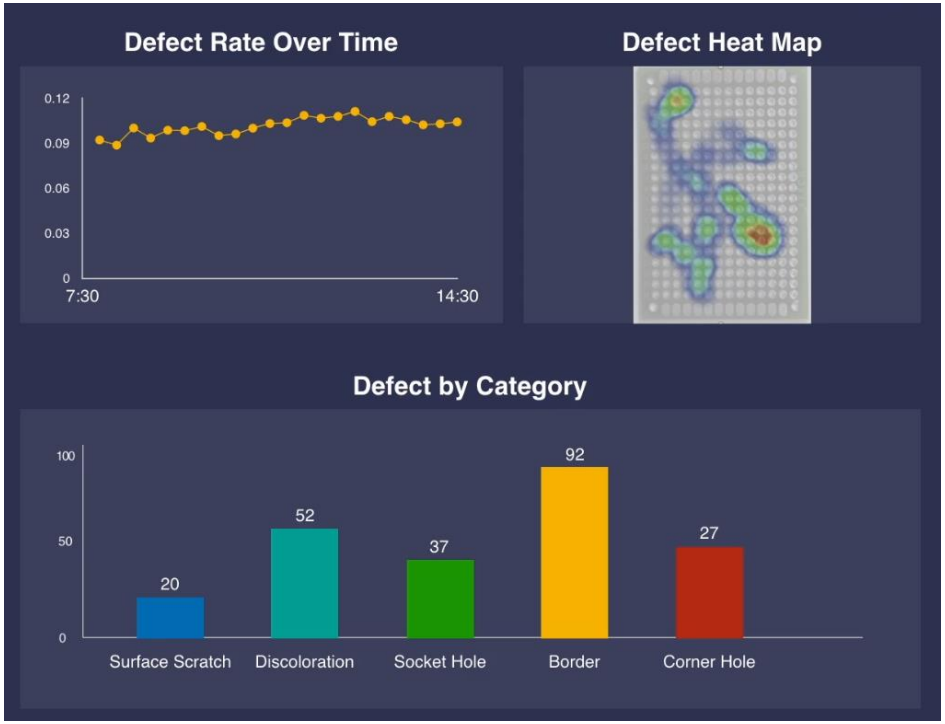


https://www.slideshare.net/awskorea/amazon-deeplearningcasesandmlonaws

Convolutional Neural Network (CNN)

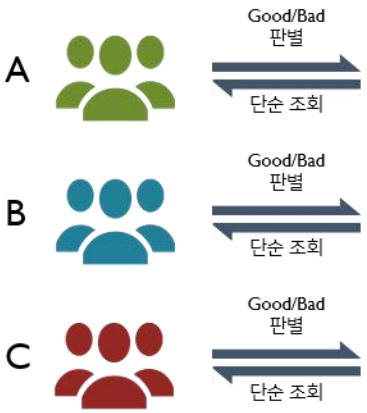
- 제조업 활용 사례 1: 불량 제품 및 원인 영역 탐지 (by landing.ai)

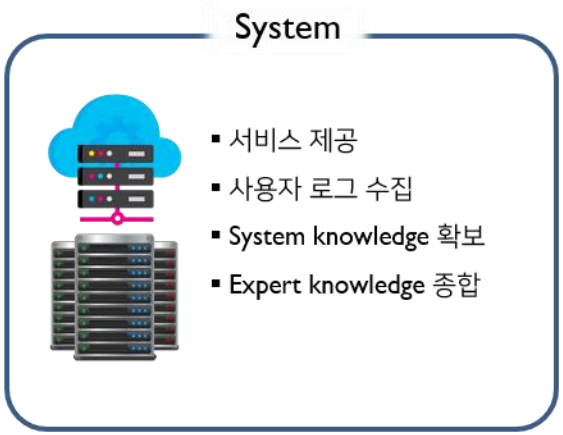


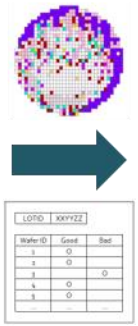


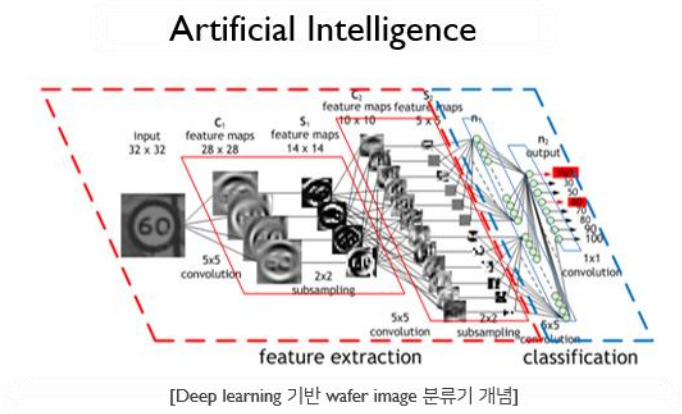
Convolutional Neural Network (CNN)

- 제조업 활용 사례 2: 불량 제품 및 원인 영역 탐지 (by DSBA@KU with Samsung)

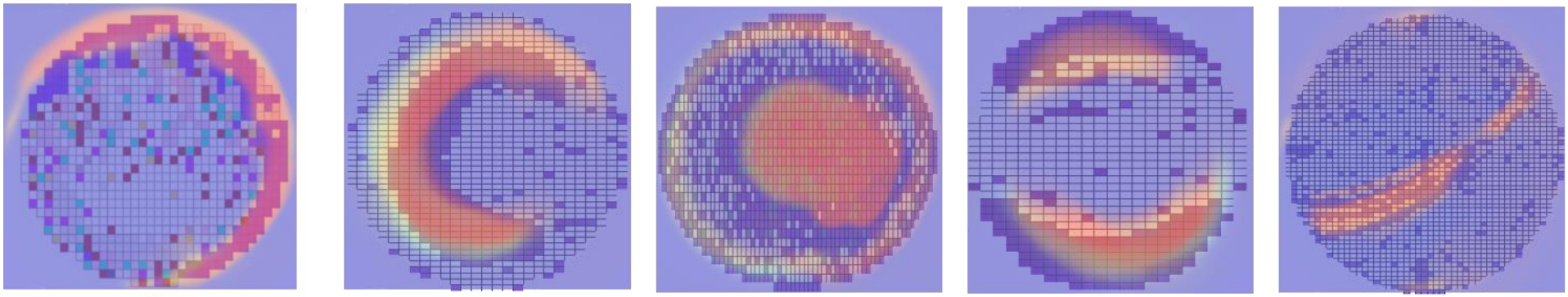






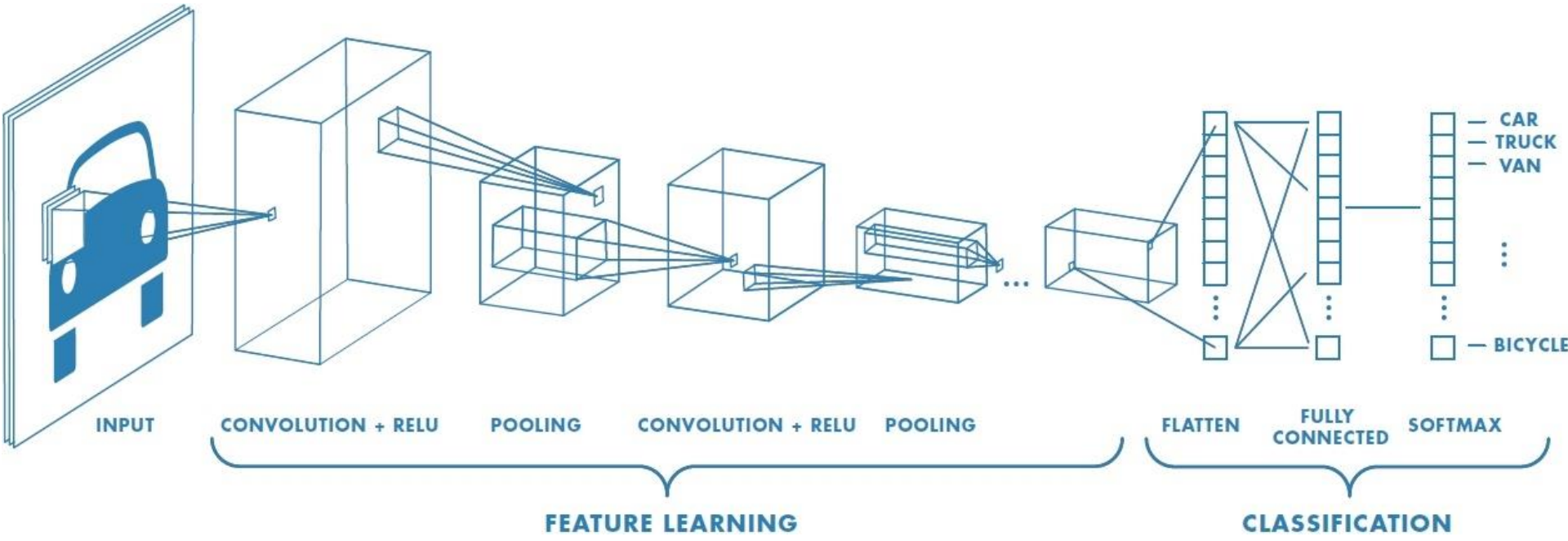


[Deep learning 기반 wafer image 분류기 개념]



Convolutional Neural Network (CNN)

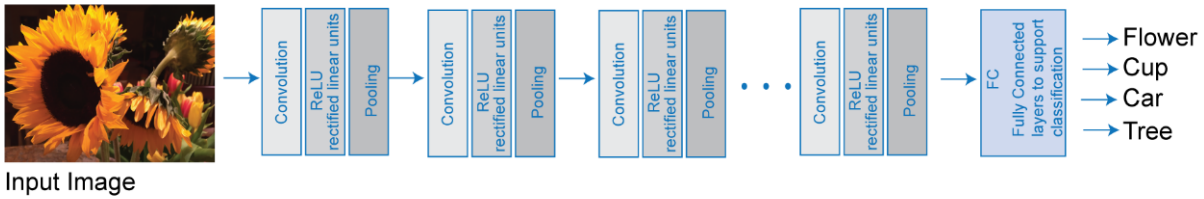
- CNN 작동 과정
  - ✓일반적인 CNN은 Convolution 연산, Activation 연산(대부분 ReLU), Pooling 연산의 반복으로 구성됨 (Feature Learning)
  - ✓일정 횟수 이상의 Feature Learning 과정 이후에는 Flatten 과정을 통해 이미지가 1차원의 벡터로 변환됨

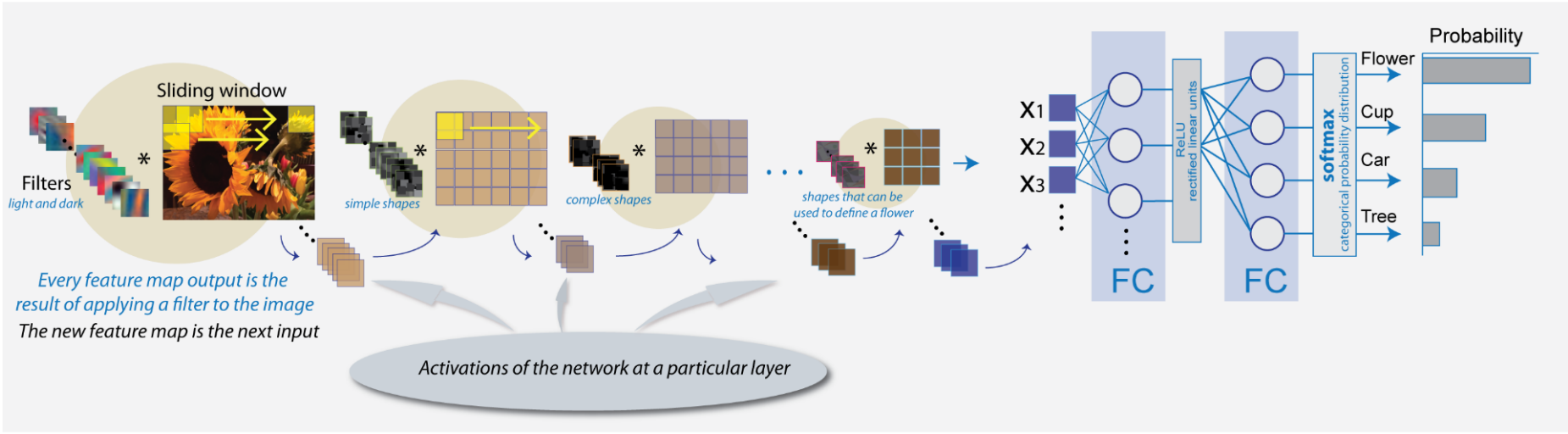


https://kr.mathworks.com/solutions/deep-learning/convolutional-neural-network.html

Convolutional Neural Network (CNN)

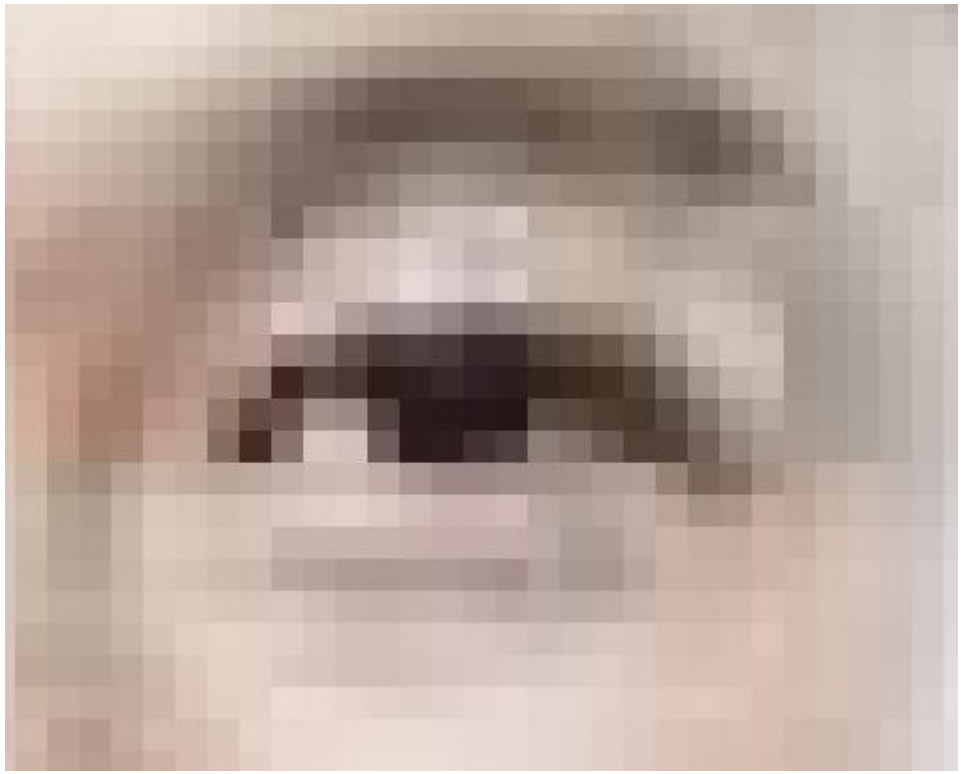
- CNN 작동 과정
  - ✓일반적인 CNN은 Convolution 연산, Activation 연산(대부분 ReLU), Pooling 연산의 반복으로 구성됨 (Feature Learning)
  - ✓일정 횟수 이상의 Feature Learning 과정 이후에는 Flatten 과정을 통해 이미지가 1차원의 벡터로 변환됨





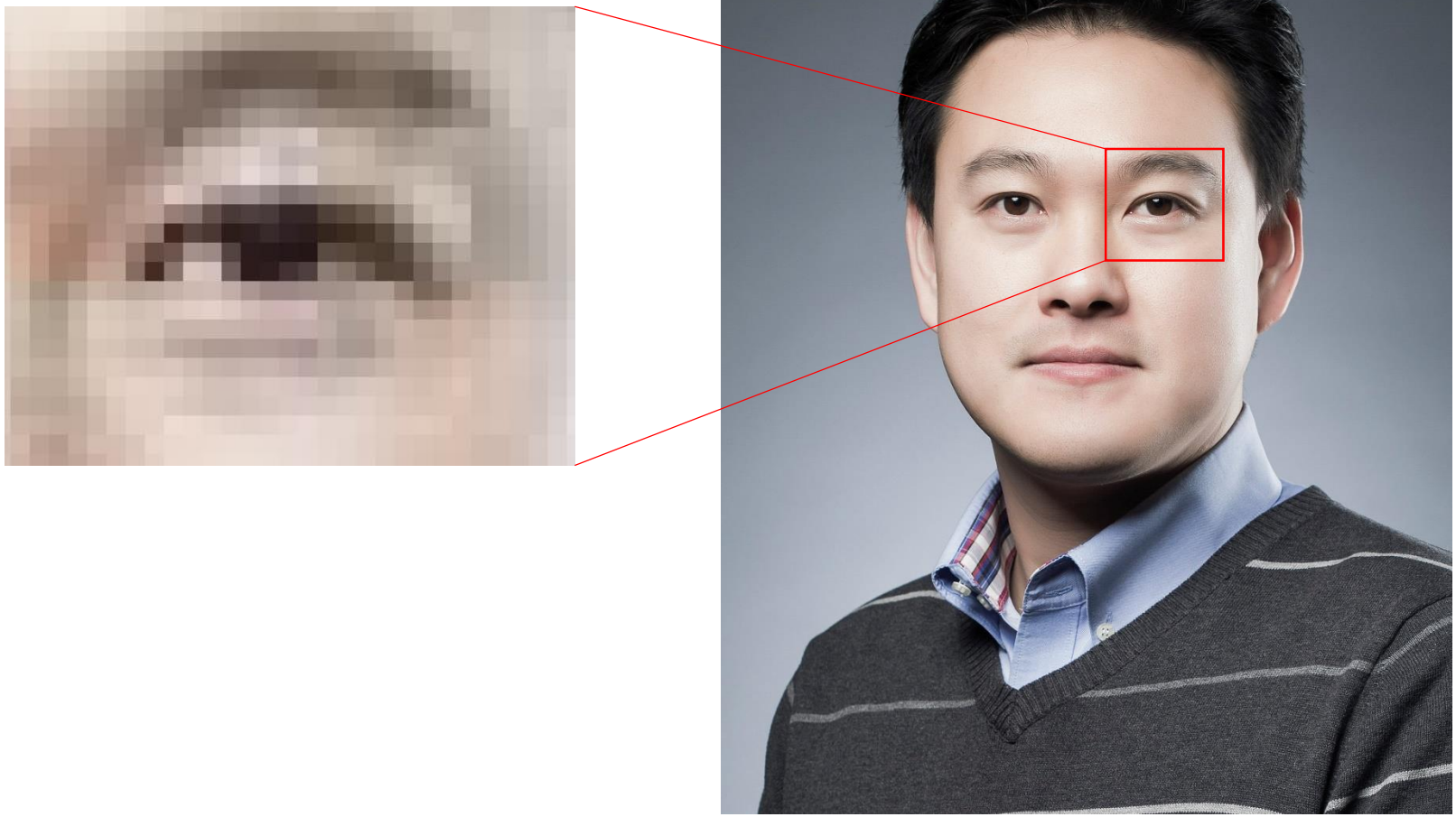
CNN Basics: Image Representation

- Image Representation: 이미지를 어떻게 컴퓨터한테 숫자로 인식시킬 것인가?
  - ✓누구의 눈일까요?



CNN Basics: Image Representation

- Image Representation: 이미지를 어떻게 컴퓨터한테 숫자로 인식시킬 것인가?
  - ✓제 눈입니다…



CNN Basics: Image Representation

- 컬러 이미지는 3차원의 Tensor로 표현됨: Width X Height X 3 (RGB)

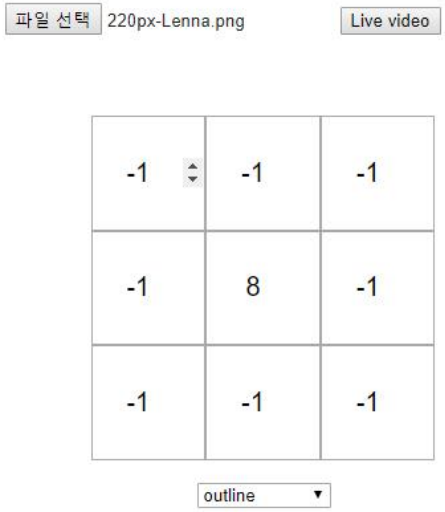

```
> pskang[1:10, 1:10, 1]
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]      [,9]     [,10]
 [1,] 0.2784314 0.2784314 0.2745098 0.2745098 0.2745098 0.2745098 0.2705882 0.2705882 0.2862745 0.2901961
 [2,] 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2666667 0.2705882
 [3,] 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2705882 0.2745098
 [4,] 0.2705882 0.2705882 0.2745098 0.2745098 0.2745098 0.2745098 0.2784314 0.2784314 0.2823529 0.2823529
 [5,] 0.2705882 0.2705882 0.2705882 0.2745098 0.2745098 0.2784314 0.2784314 0.2784314 0.2784314 0.2784314
 [6,] 0.2705882 0.2705882 0.2745098 0.2745098 0.2745098 0.2745098 0.2784314 0.2784314 0.2823529 0.2784314
 [7,] 0.2705882 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2784314 0.2941176 0.2862745
 [8,] 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2901961 0.2823529
 [9,] 0.2745098 0.2705882 0.2745098 0.2784314 0.2823529 0.2823529 0.2745098 0.2666667 0.2784314 0.2784314
[10,] 0.2784314 0.2784314 0.2784314 0.2823529 0.2862745 0.2862745 0.2784314 0.2745098 0.2745098 0.2745098
```

```
> pskang[1:10, 1:10, 2]
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]      [,9]     [,10]
 [1,] 0.2980392 0.2980392 0.2941176 0.2941176 0.2941176 0.2941176 0.2901961 0.2901961 0.2980392 0.3019608
 [2,] 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2784314 0.2823529
 [3,] 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2823529 0.2862745
 [4,] 0.2901961 0.2901961 0.2941176 0.2941176 0.2941176 0.2941176 0.2980392 0.2980392 0.2941176 0.2941176
 [5,] 0.2901961 0.2901961 0.2901961 0.2941176 0.2941176 0.2980392 0.2980392 0.2980392 0.2901961 0.2901961
 [6,] 0.2901961 0.2901961 0.2941176 0.2941176 0.2941176 0.2941176 0.2980392 0.2980392 0.2941176 0.2901961
 [7,] 0.2901961 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2980392 0.3058824 0.2980392
 [8,] 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.3019608 0.2941176
 [9,] 0.2862745 0.2823529 0.2862745 0.2901961 0.2941176 0.2941176 0.2862745 0.2784314 0.2901961 0.2901961
[10,] 0.2901961 0.2901961 0.2901961 0.2941176 0.2980392 0.2980392 0.2901961 0.2862745 0.2862745 0.2862745
```

```
> pskang[1:10, 1:10, 3]
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]      [,9]     [,10]
 [1,] 0.3215686 0.3215686 0.3176471 0.3176471 0.3176471 0.3176471 0.3137255 0.3137255 0.3254902 0.3294118
 [2,] 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3058824 0.3098039
 [3,] 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3098039 0.3137255
 [4,] 0.3137255 0.3137255 0.3176471 0.3176471 0.3176471 0.3176471 0.3215686 0.3215686 0.3215686 0.3215686
 [5,] 0.3137255 0.3137255 0.3137255 0.3176471 0.3176471 0.3215686 0.3215686 0.3215686 0.3176471 0.3176471
 [6,] 0.3137255 0.3137255 0.3176471 0.3176471 0.3176471 0.3176471 0.3215686 0.3215686 0.3215686 0.3176471
 [7,] 0.3137255 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3215686 0.3333333 0.3254902
 [8,] 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3294118 0.3215686
 [9,] 0.3058824 0.3019608 0.3058824 0.3098039 0.3137255 0.3137255 0.3058824 0.2980392 0.3098039 0.3098039
[10,] 0.3098039 0.3098039 0.3098039 0.3137255 0.3176471 0.3176471 0.3098039 0.3058824 0.3058824 0.3058824
```

CNN Basics: Convolution

- 문제점
  - ✓ 모든 픽셀을 서로 다른 가중치로 연결하게 되면 Input layer와 First hidden layer 사이에 너무 많은 weights가 생김
- Image Convolution (Filter, Kernel)
  - ✓ 특정 속성을 탐지하는데 사용하는 matrix (예: edge detection)



She appeared on the cover of the journal *Optical Engineering* in 1991, and the model herself was invited to a conference of the Image Science and Technology society in 1997. (덕중의 덕은 양덕이니라)

https://www.software.ac.uk/blog/2016-10-03-how-photo-playboy-became-part-scientific-culture

CNN Basics: Convolution

- Image Convolution (Filter, Kernel)
  - ✓ 특정 속성을 탐지하는데 사용하는 matrix

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input

1	0	1
0	1	0
1	0	1

Filter / Kernel

https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Basics: Convolution

- Image Convolution (Filter, Kernel)
  - ✓ 특정 속성을 탐지하는데 사용하는 matrix

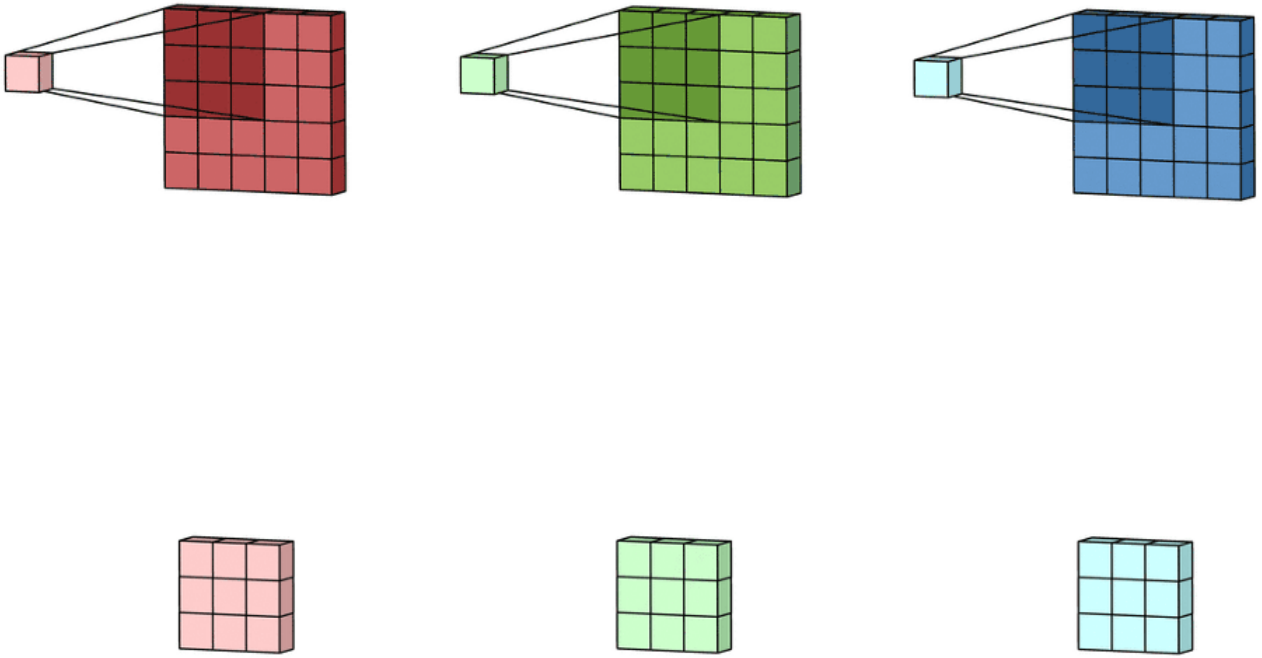
1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Basics: Convolution

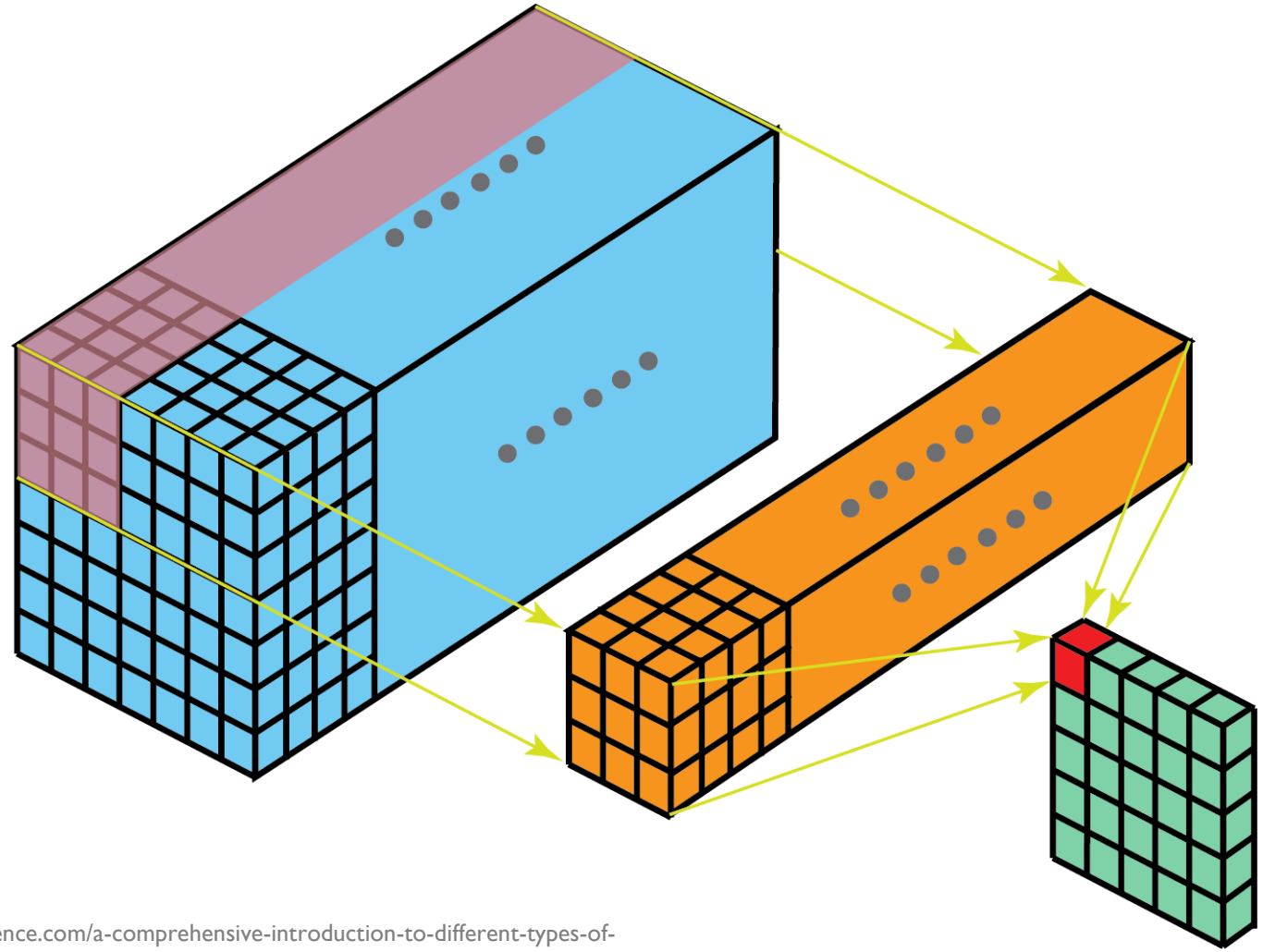
- 이미지는 3차원 Tensor(가로 픽셀, 세로 픽셀, RGB)이므로 필터 역시 3차원임



https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

CNN Basics: Convolution

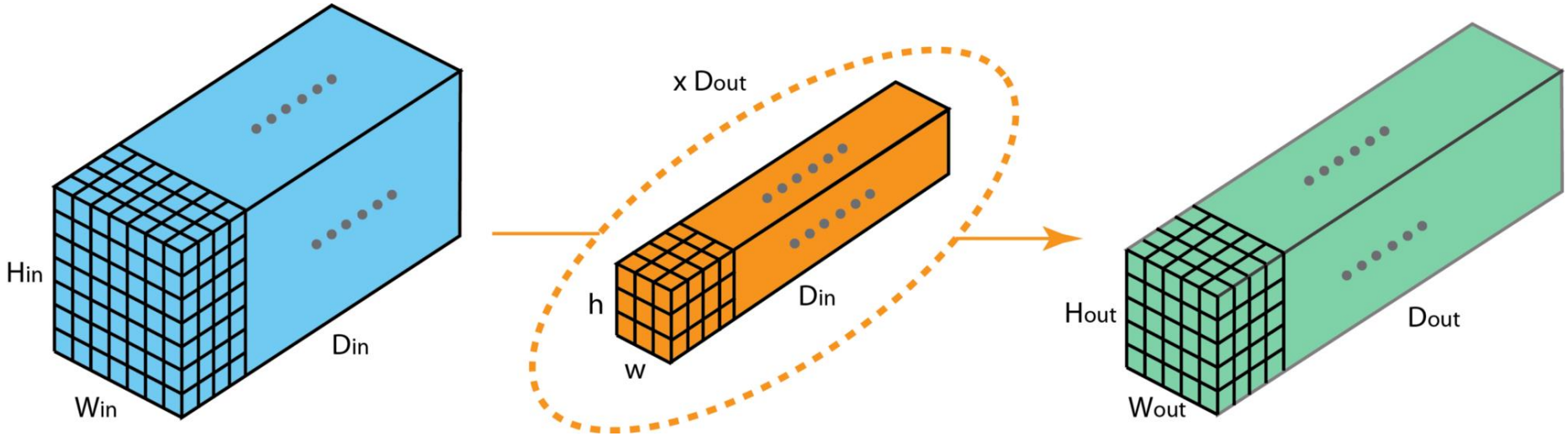
- 이를 일반화하면…



https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

CNN Basics: Convolution

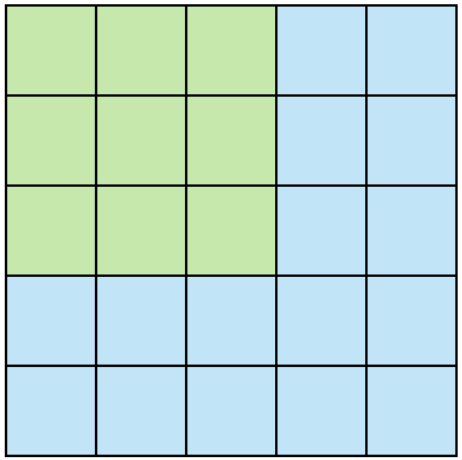
- 이를 일반화하면…



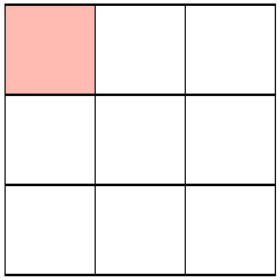
https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

CNN Basics: Convolution

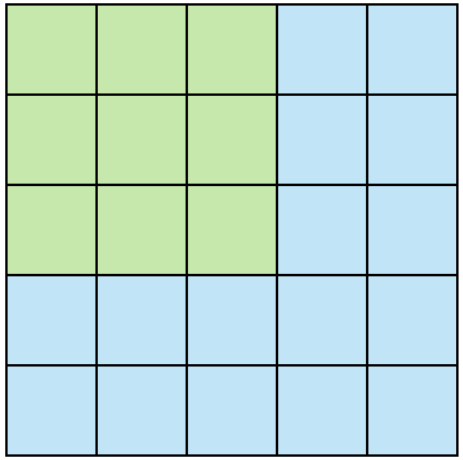
- Image Convolution (Filter, Kernel)
  - ✓ 이슈: 한번에 한 칸씩 이동하면 너무 오래 걸리지 않을까?
  - ✓ 해결책: Filter가 한번에 여러 칸을 이동하도록 허용하자 (Stride)



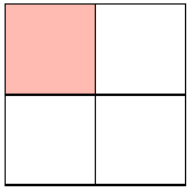
Stride 1



Feature Map



Stride 2



Feature Map

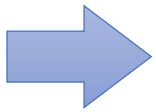
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Basics: Convolution

- Image Convolution (Filter, Kernel)
  - ✓ 문제점: 가장자리에 있는 픽셀들은 중앙에 위치한 픽셀들에 비해 Convolution 연산이 적게 수행됨
  - ✓ 해결책: Padding (원래 이미지 테두리에 0의 값을 갖는 pad를 추가)

3	1	1	2	8	4
1	0	7	3	2	6
2	3	5	1	1	3
1	4	1	2	6	5
3	2	1	3	7	2
9	2	6	2	5	1

Padding with size 1

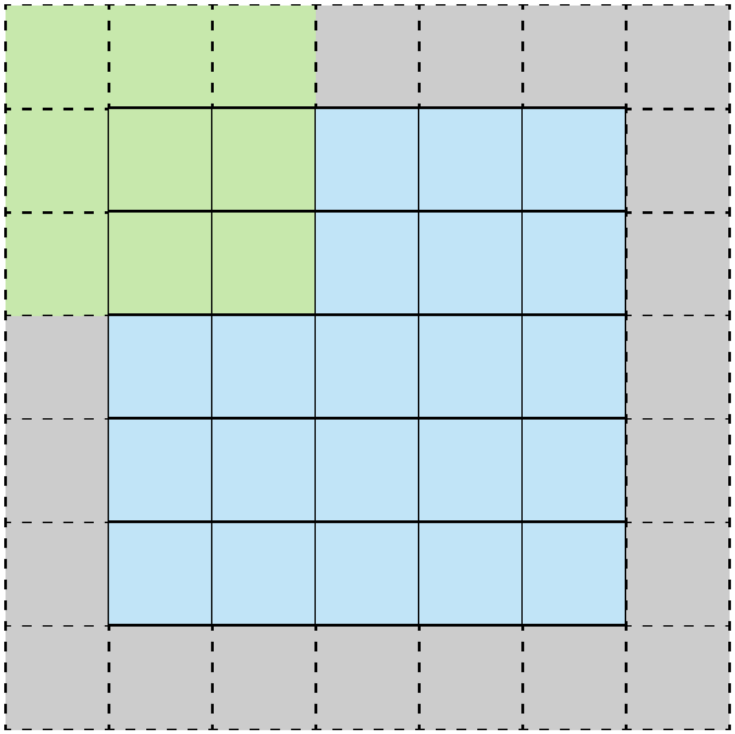


0	0	0	0	0	0	0	0
0	3	1	1	2	8	4	0
0	1	0	7	3	2	6	0
0	2	3	5	1	1	3	0
0	1	4	1	2	6	5	0
0	3	2	1	3	7	2	0
0	9	2	6	2	5	1	0
0	0	0	0	0	0	0	0

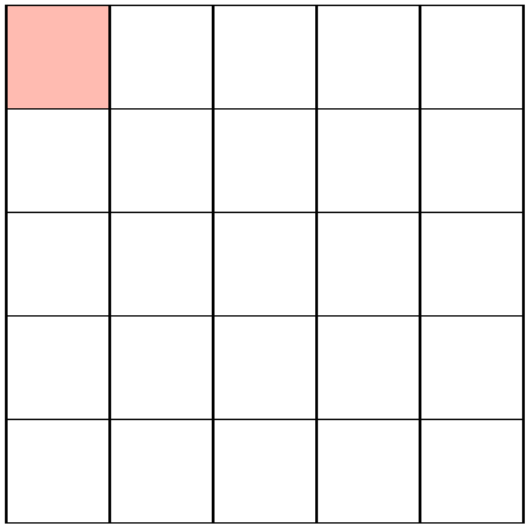
https://medium.com/machine-learning-bites/deeplearning-series-convolutional-neural-networks-a9c2f2ee1524

CNN Basics: Convolution

- Image Convolution (Filter, Kernel)
  - ✓ 해결책: Padding (원래 이미지 테두리에 0의 값을 갖는 pad를 추가)



Stride 1 with Padding



Feature Map

CNN Basics: Convolution

- Image Convolution (Filter, Kernel)
  - ✓ Convolution with padding

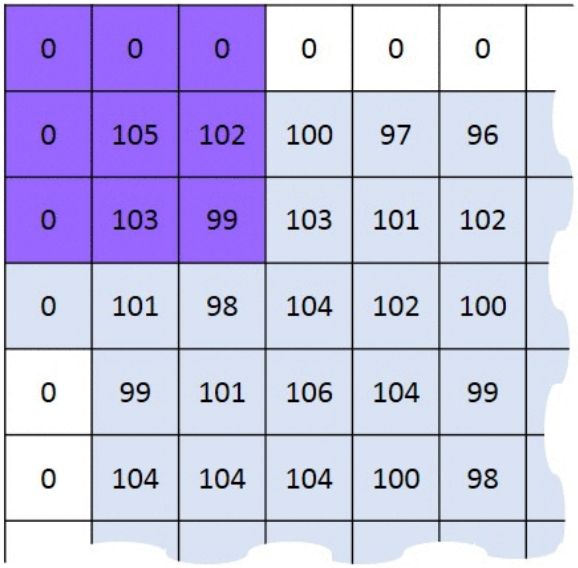
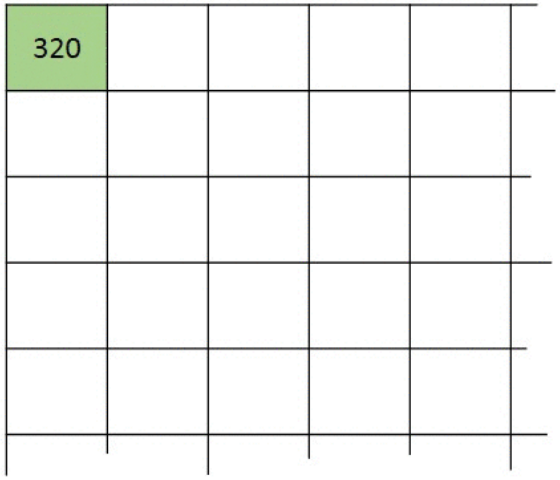


Image Matrix

Kernel Matrix

0	-1	0
-1	5	-1
0	-1	0

$$0*0+0*-1+0*0$$
$$+0*-1+105*5+102*-1$$
$$+0*0+103*-1+99*0 = 320$$



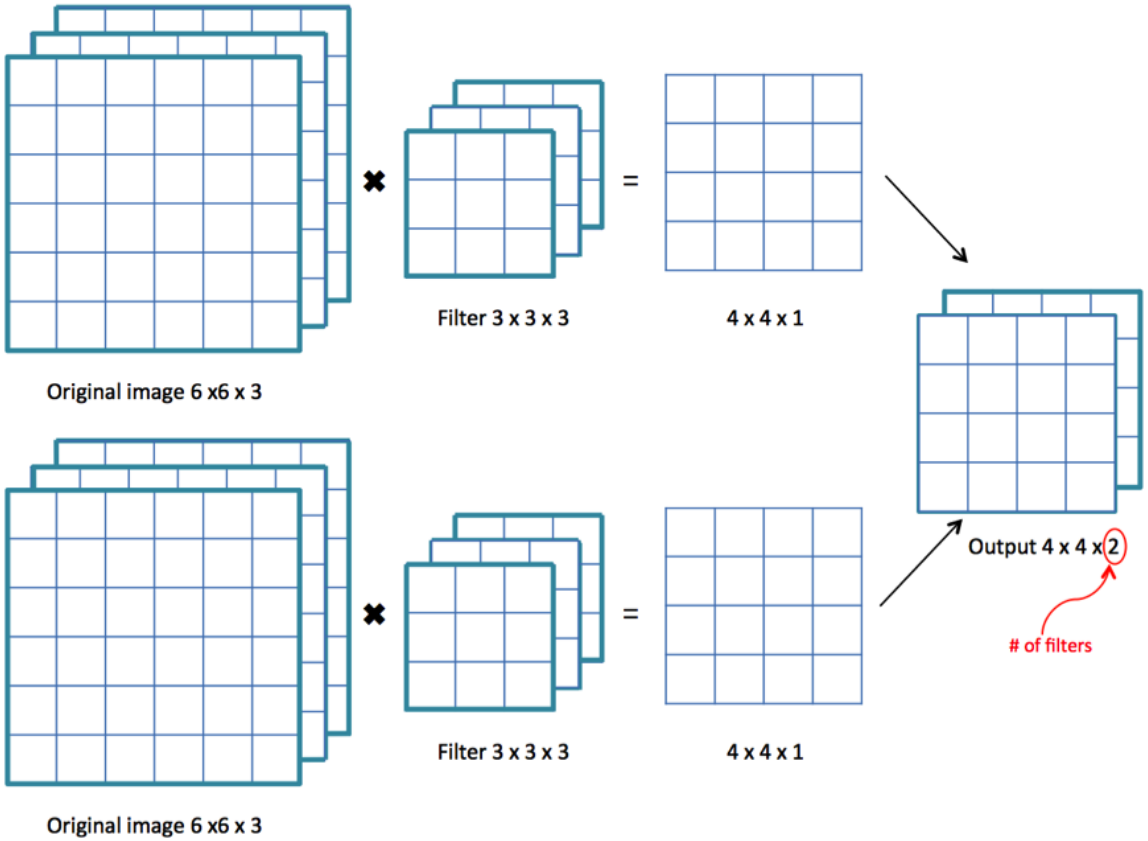
Output Matrix

Convolution with horizontal and vertical strides = 1

CNN Basics: Convolution

- Padding과 Stride 크기에 따른 Output Size

$$\text{Output size} = \left(\frac{H + 2P - F}{S} + 1\right) \times \left(\frac{W + 2P - F}{S} + 1\right)$$

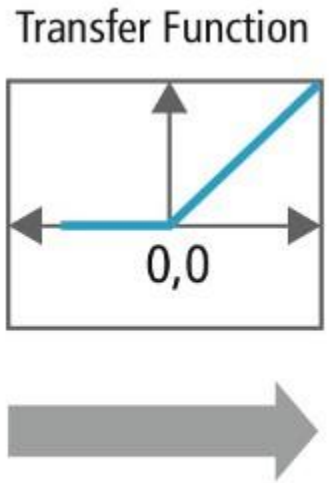


https://medium.com/machine-learning-bites/deeplearning-series-convolutional-neural-networks-a9c2f2ee1524

CNN Basics: Activation

- Activation
  - ✓ Convolution을 통해 학습된 값들의 비선형 변환을 수행
  - ✓ 대부분 Rectified Linear Unit (ReLU)을 사용

15	20	-10	35
18	-110	25	100
20	-15	25	-10
101	75	18	23

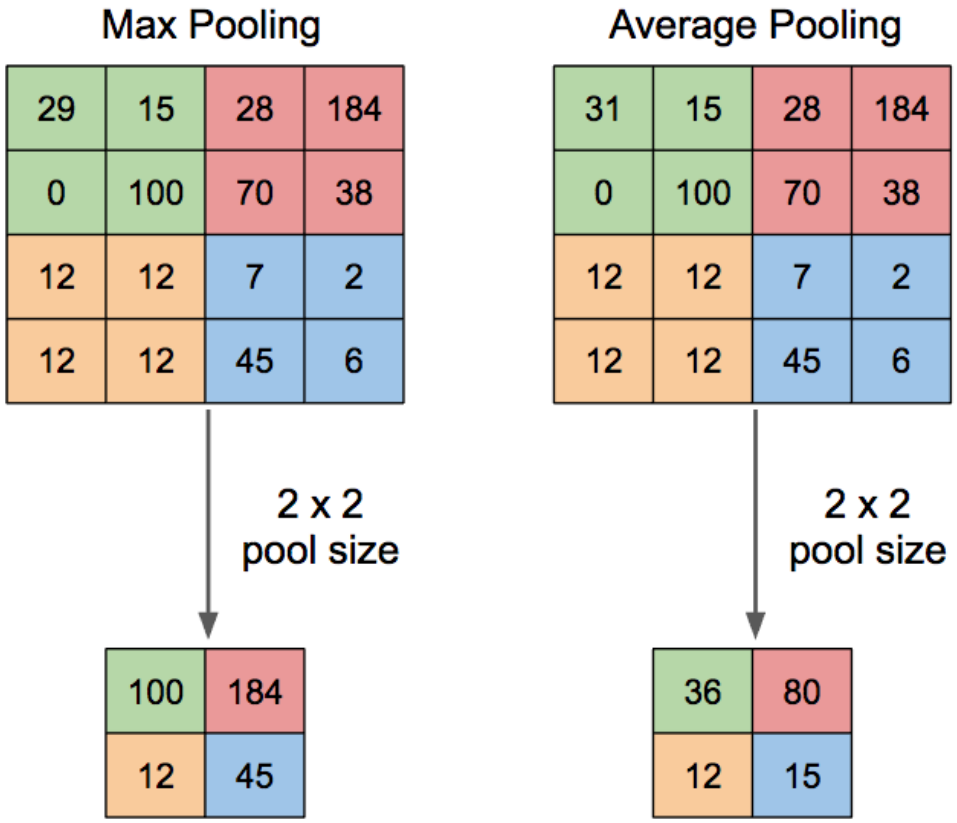


15	20	0	35
18	0	25	100
20	0	25	0
101	75	18	23

https://medium.com/data-science-group-iitr/building-a-convolutional-neural-network-in-python-with-tensorflow-d251c3ca8117

CNN Basics: Pooling

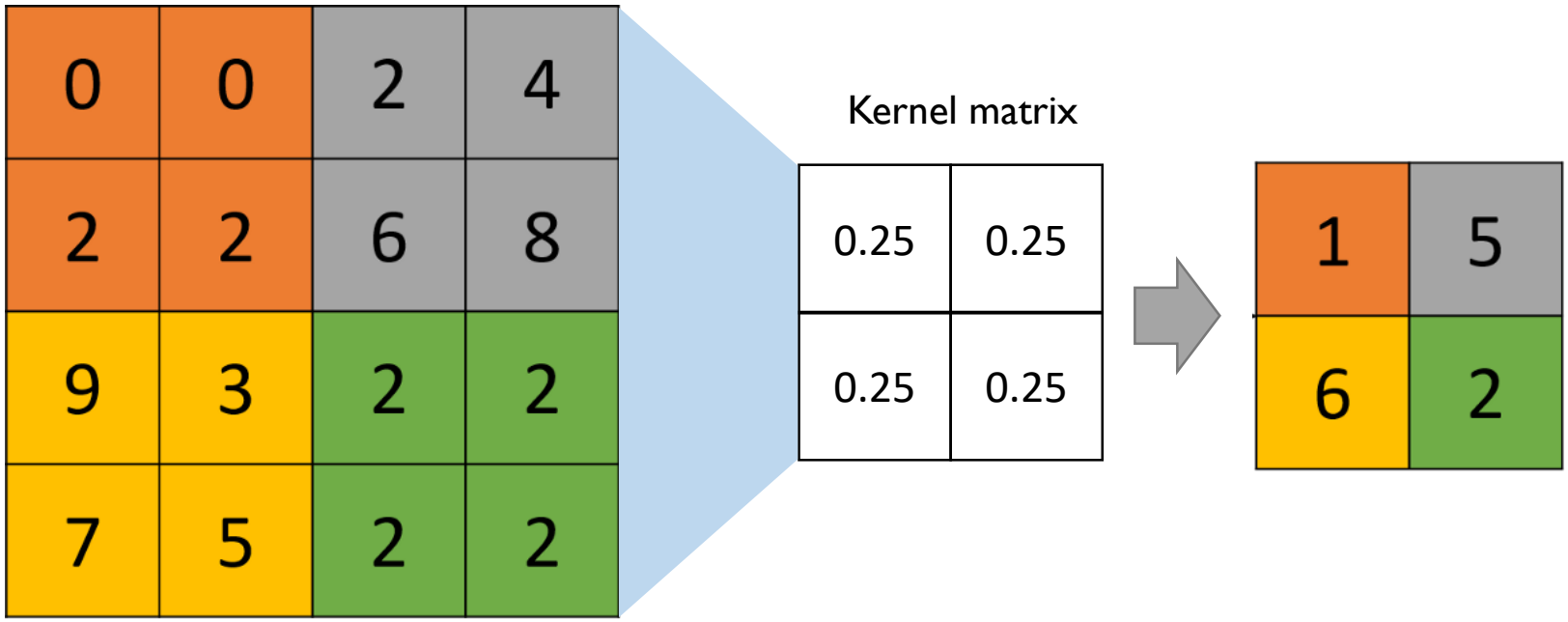
- Pooling
  - ✓ 문제점: 고차원의 Tensor를 보다 Compact하게 축약해야 하지 않을까?
  - ✓ Pooling: 일정 영역의 정보를 축약하는 역할



https://pythonmachinelearning.pro/introduction-to-convolutional-neural-networks-for-vision-tasks/

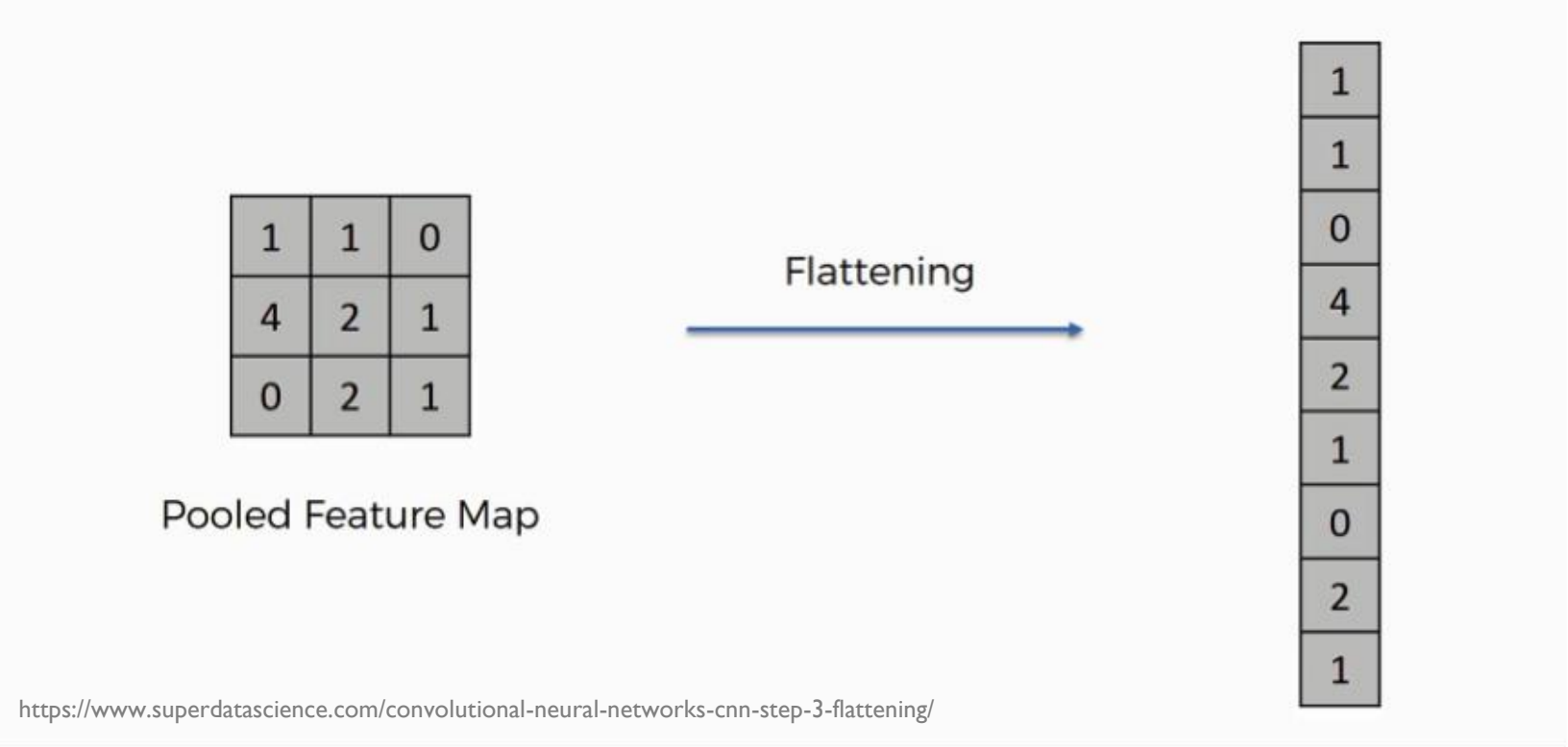
CNN Basics: Strided Convolution

- Strided Convolution
  - ✓ Average Pooling은 strided convolution의 특수한 케이스: 모든 weight가 1/n

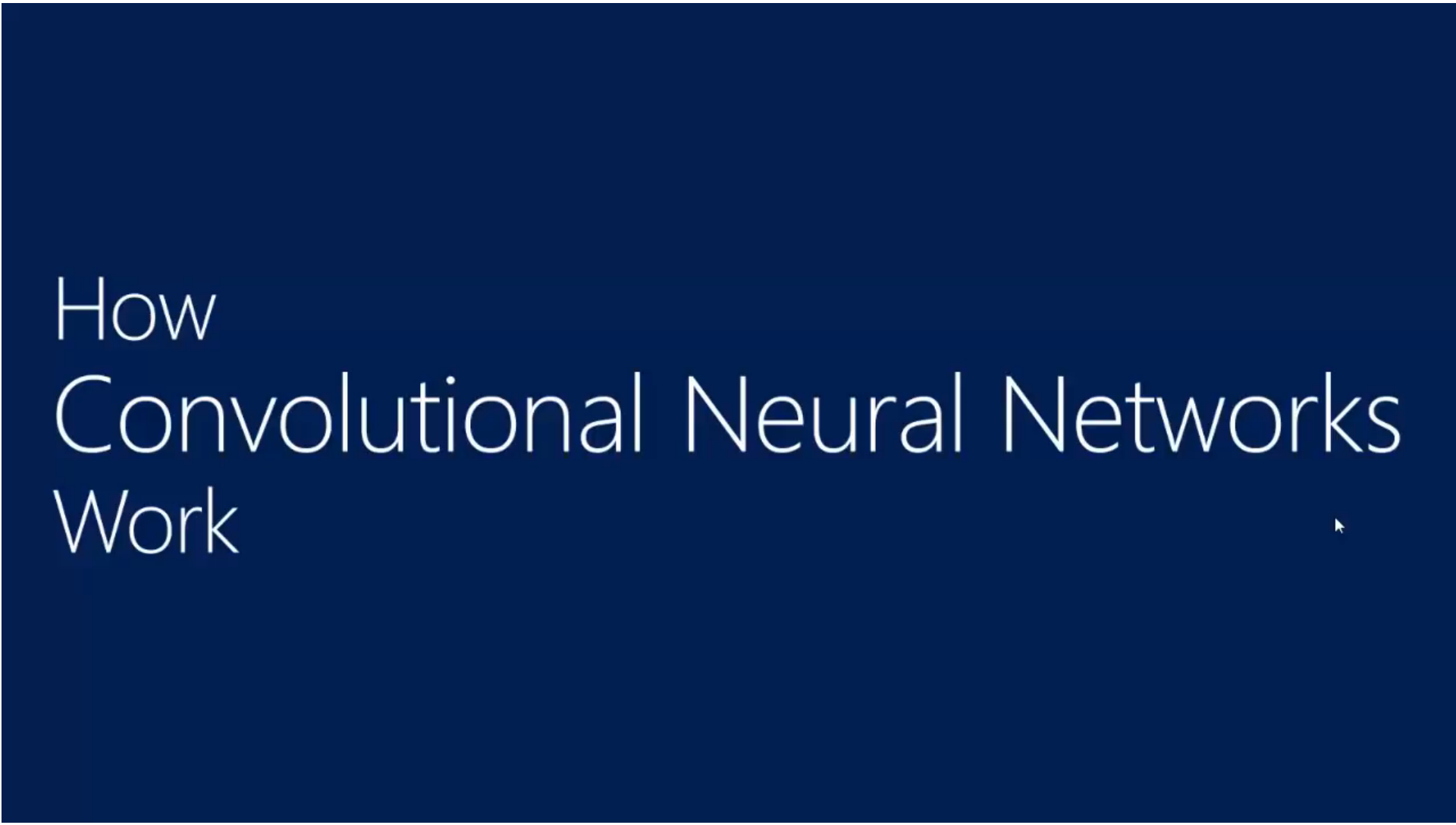


CNN Basics: Flatten

- **평탄화**: Flatten
  - ✓ 2차원/3차원의 Matrix/Tensor 구조를 1차원의 Vector로 변환하는 과정



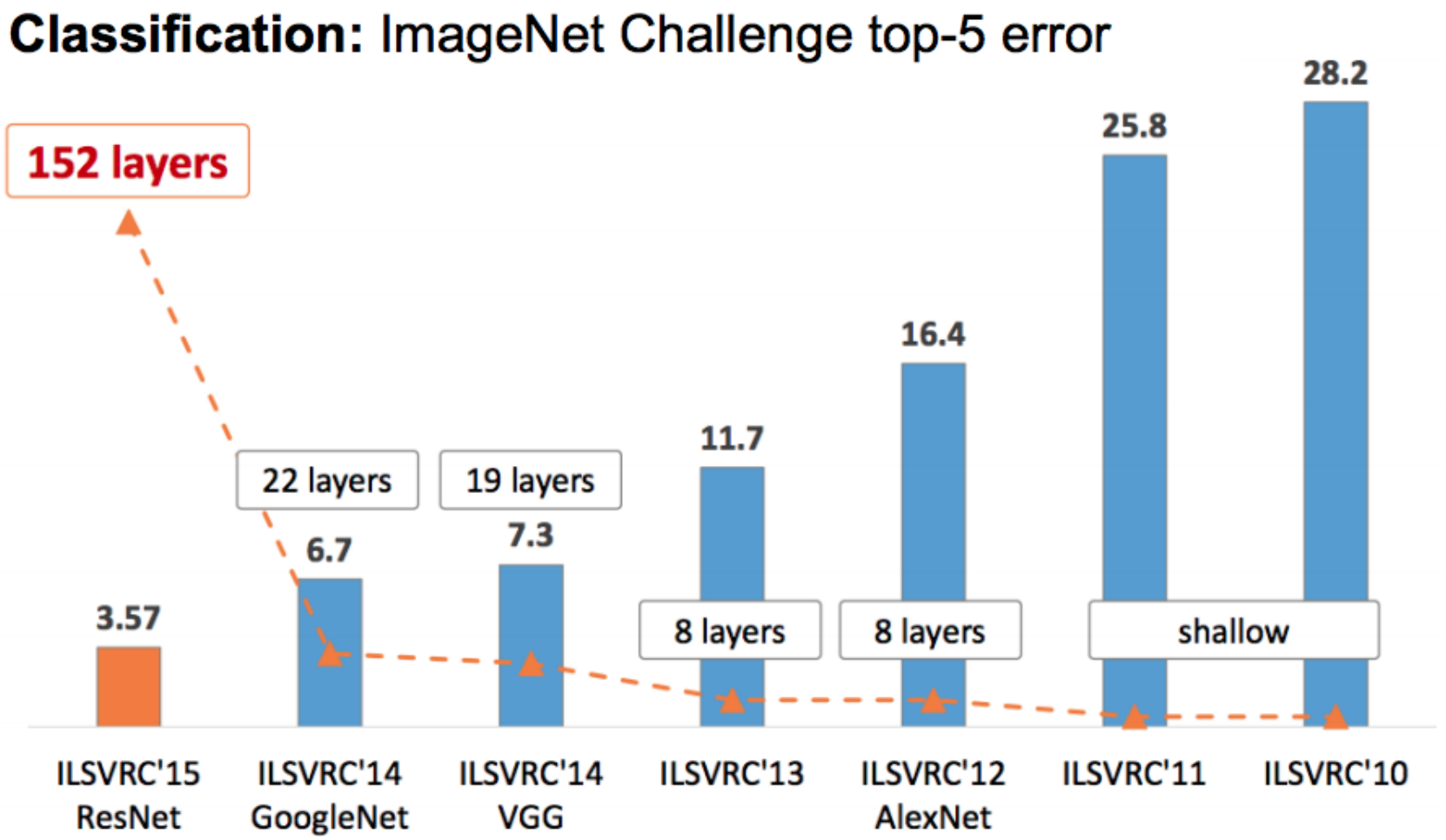
CNN Procedure



https://www.youtube.com/watch?time_continue=797&v=FmpDIaiMIeA

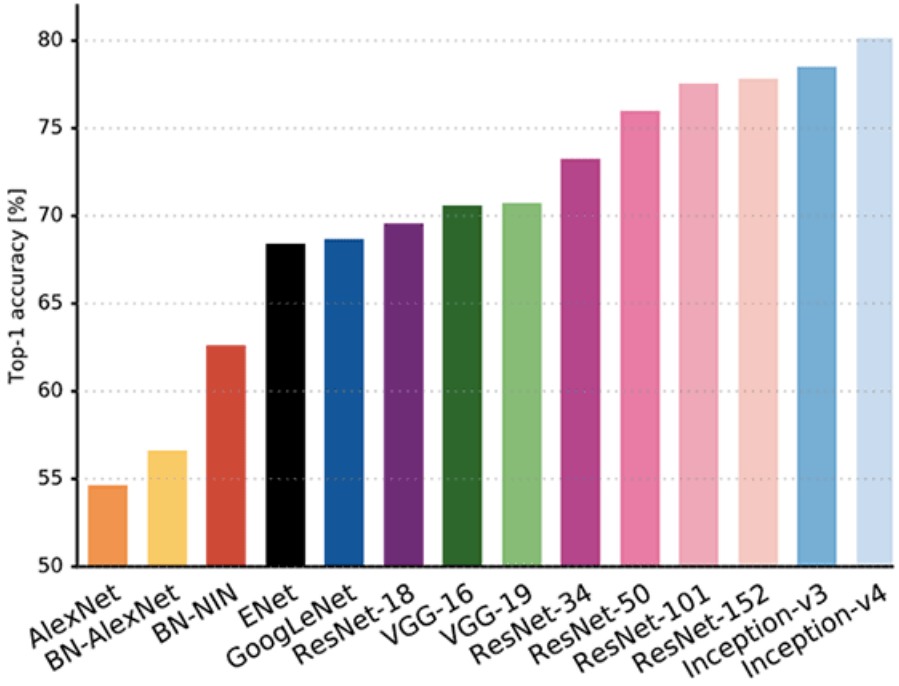
Convolutional Neural Network (CNN)

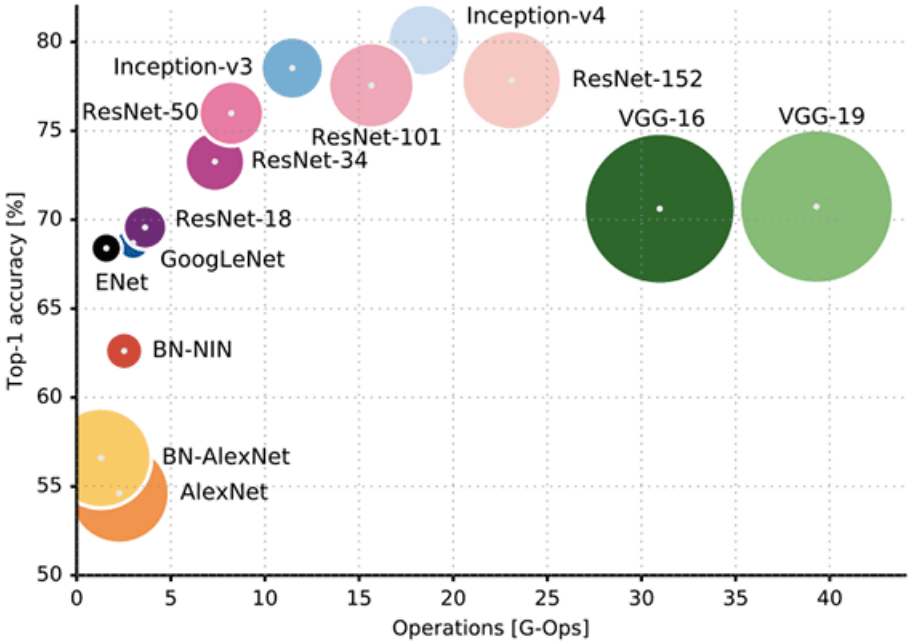
- Power of deep layers in CNN
  - ✓ Layer의 수가 많아질수록 이미지 분류 성능이 증가하는 추세를 보임



Convolutional Neural Network (CNN)

- Power of deep layers in CNN
  - ✓시간이 지날수록 효율적이며(Operation 수가 적으며) 정확한 모델 구조가 등장

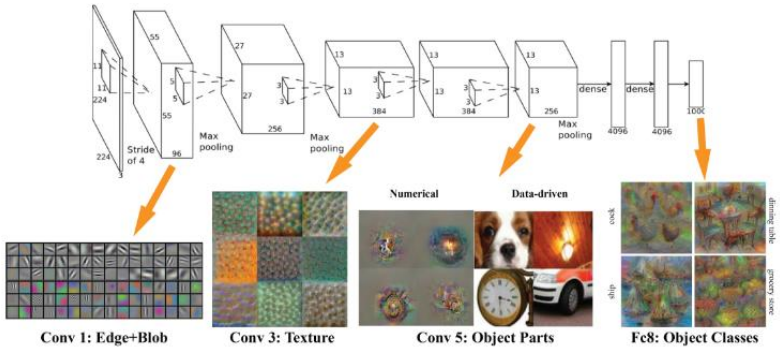




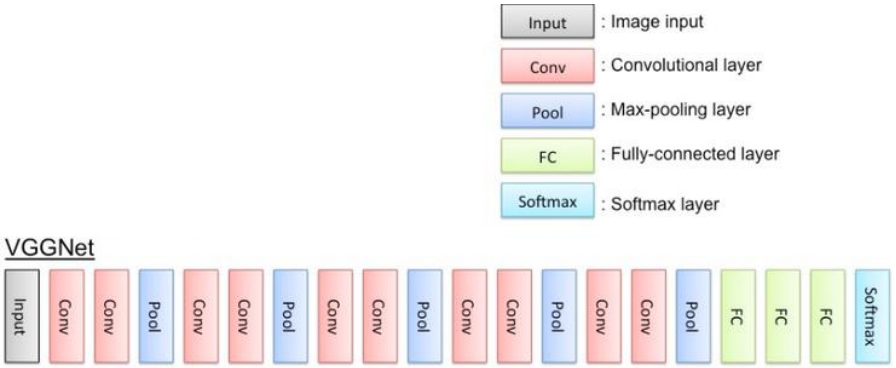
https://www.topbots.com/14-design-patterns-improve-convolutional-neural-network-cnn-architecture/

Convolutional Neural Network (CNN)

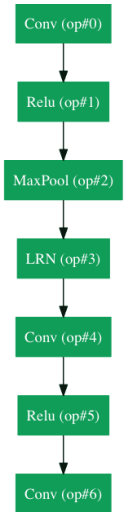
- 대표적인 CNN 구조



Alexnet

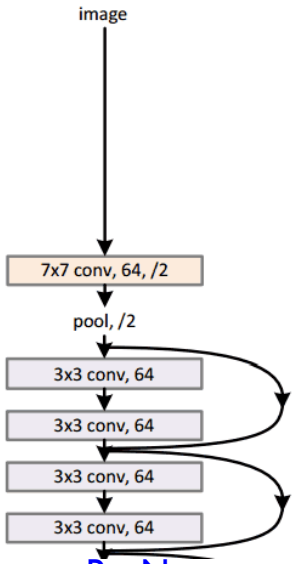


VGGNet



GoogLeNet

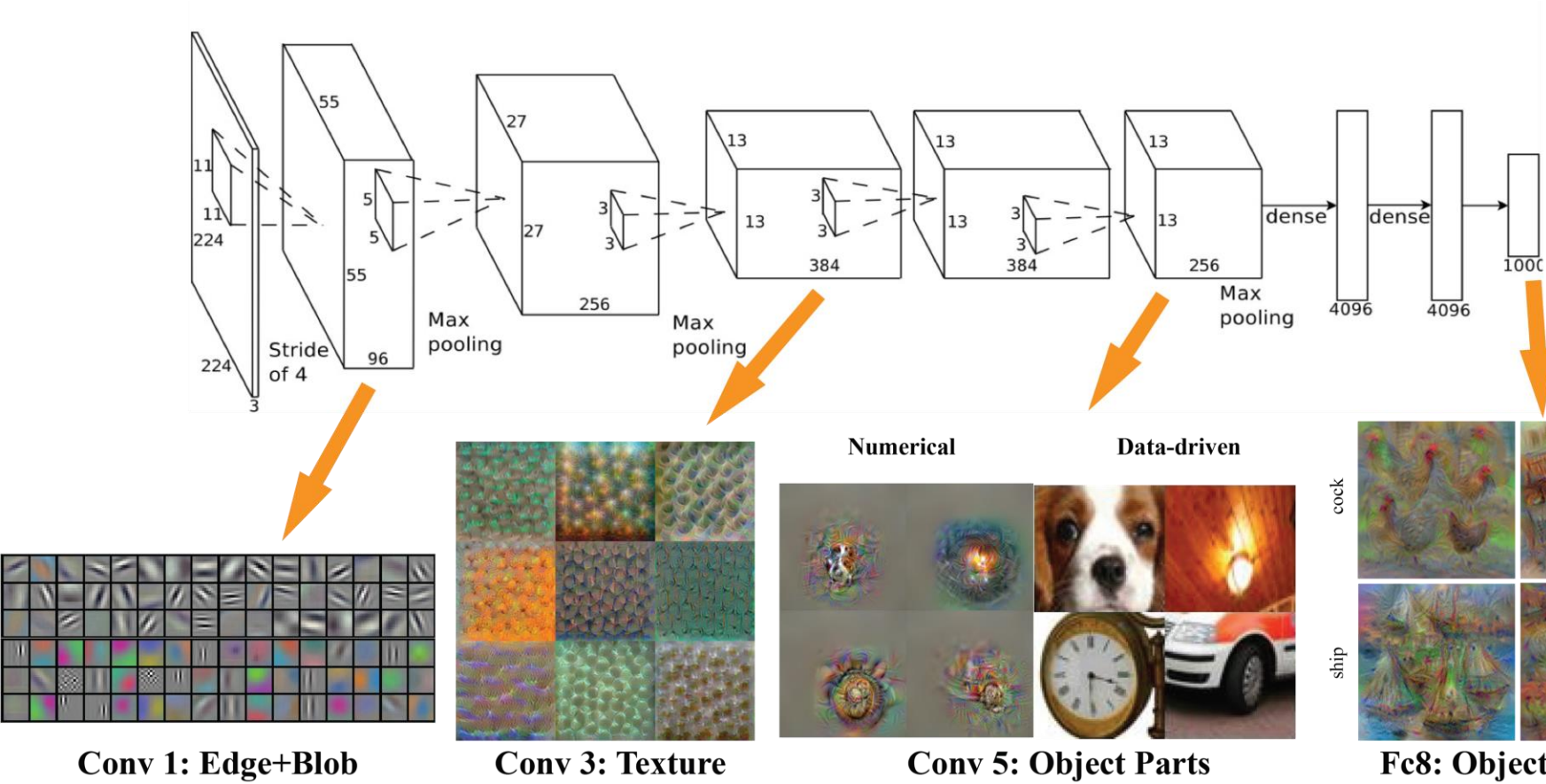
34-layer residual



ResNet

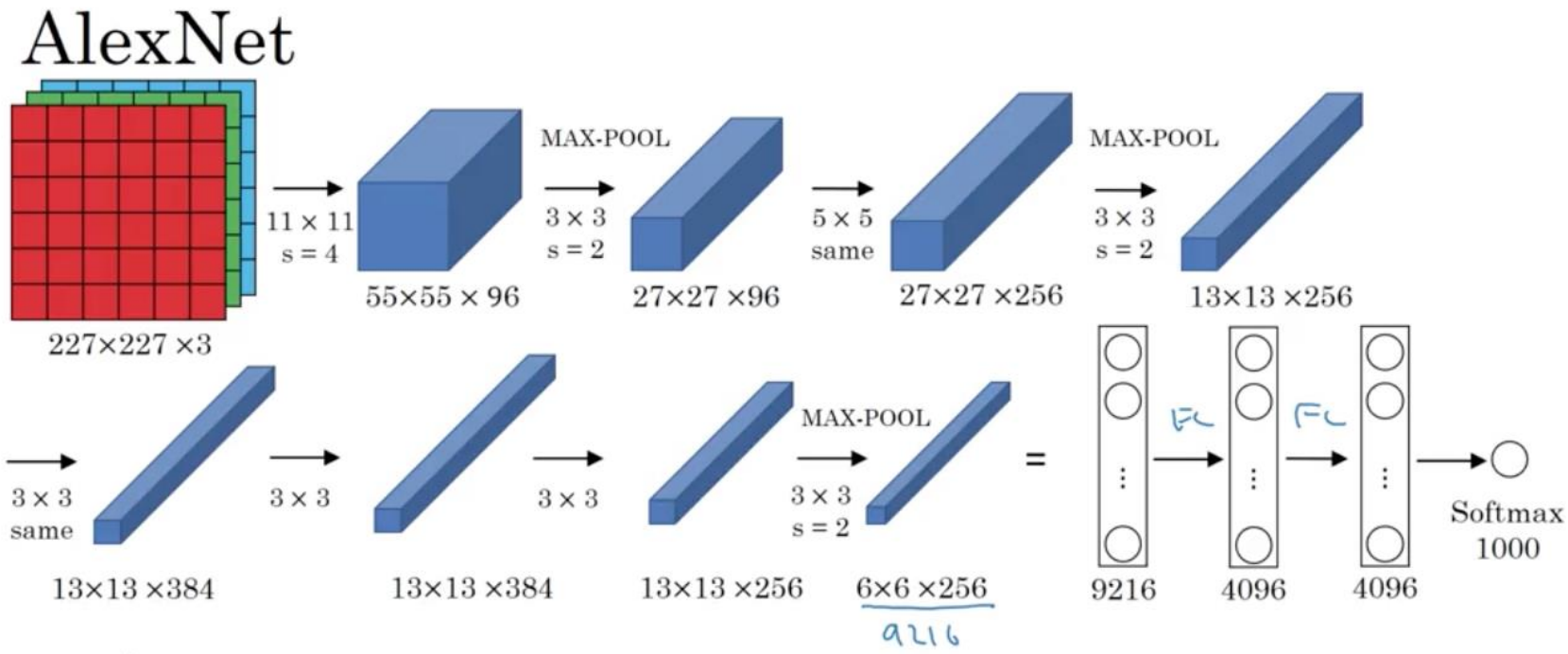
CNN Architecture 1: AlexNet

- AlexNet



CNN Architecture 1: AlexNet

- AlexNet

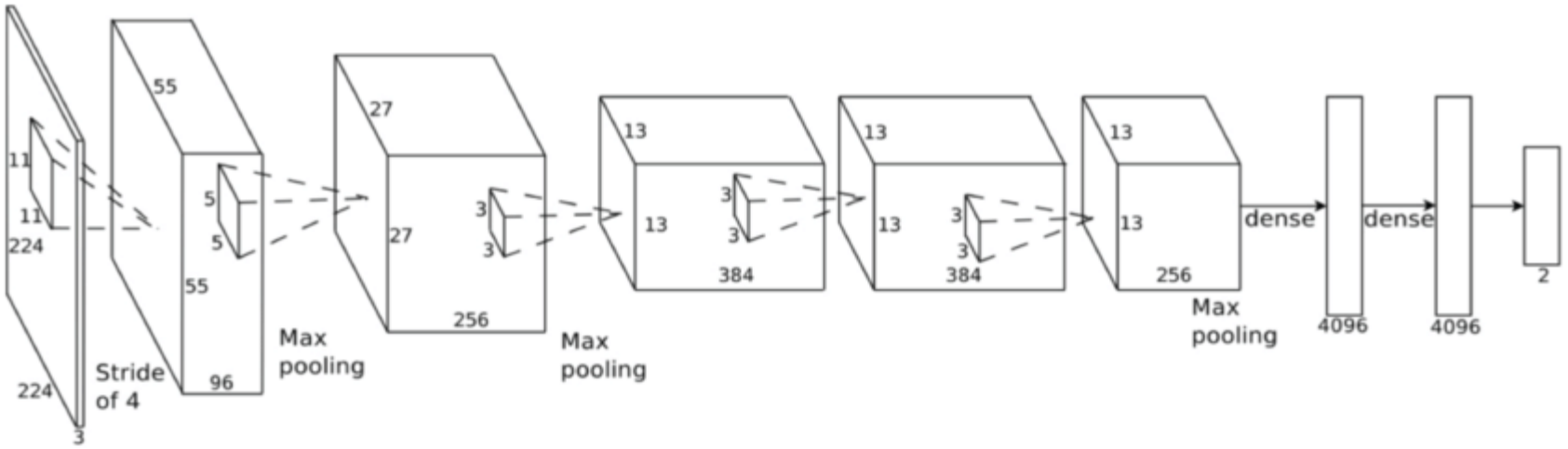


[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

Andrew Ng

CNN Architecture 1: AlexNet

- AlexNet



✓ 224*224 크기의 이미지를 Input으로 사용

✓ 초기 단계에서는 큰 필터 사이즈와 Stride를 사용

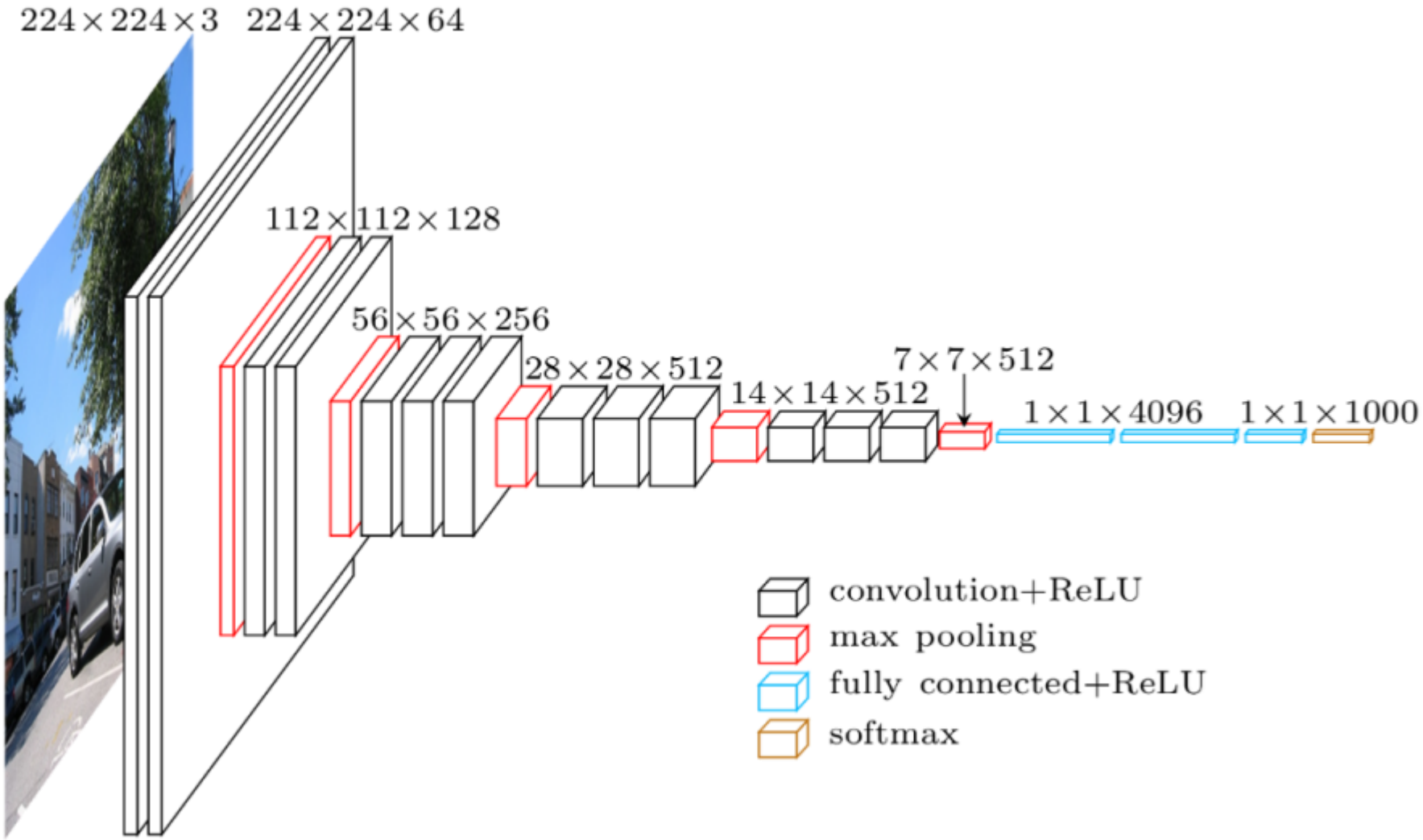
✓ 상위 Layer로 갈수록 작은 필터 사이즈와 Stride를 사용

✓ 2개의 Fully connected layer 존재

✓ 파라미터 총 수: 60 million

CNN Architecture 2:VGGNet

- VGGNet

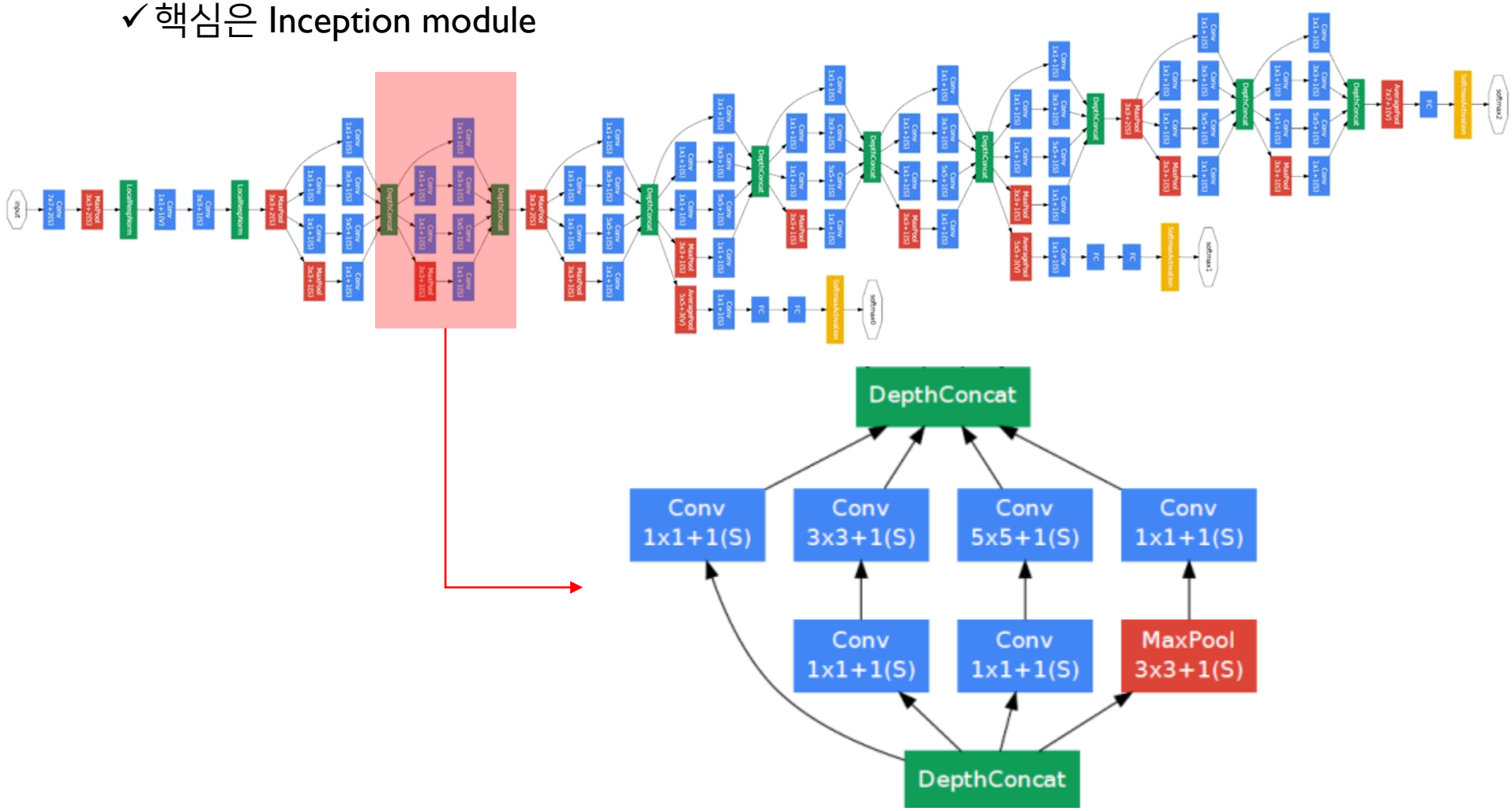


CNN Architecture 2:VGGNet

- VGGNet
  - ✓ AlexNet에 비해 단순하지만 깊은 구조
  - ✓ 3 by 3 convolution with stride 1을 기본 연산으로 하며 중간 중간에 2 by 2 max polling을 수행
  - ✓ 파라미터 총 수: 138 million

CNN Architecture 3: GoogLeNet

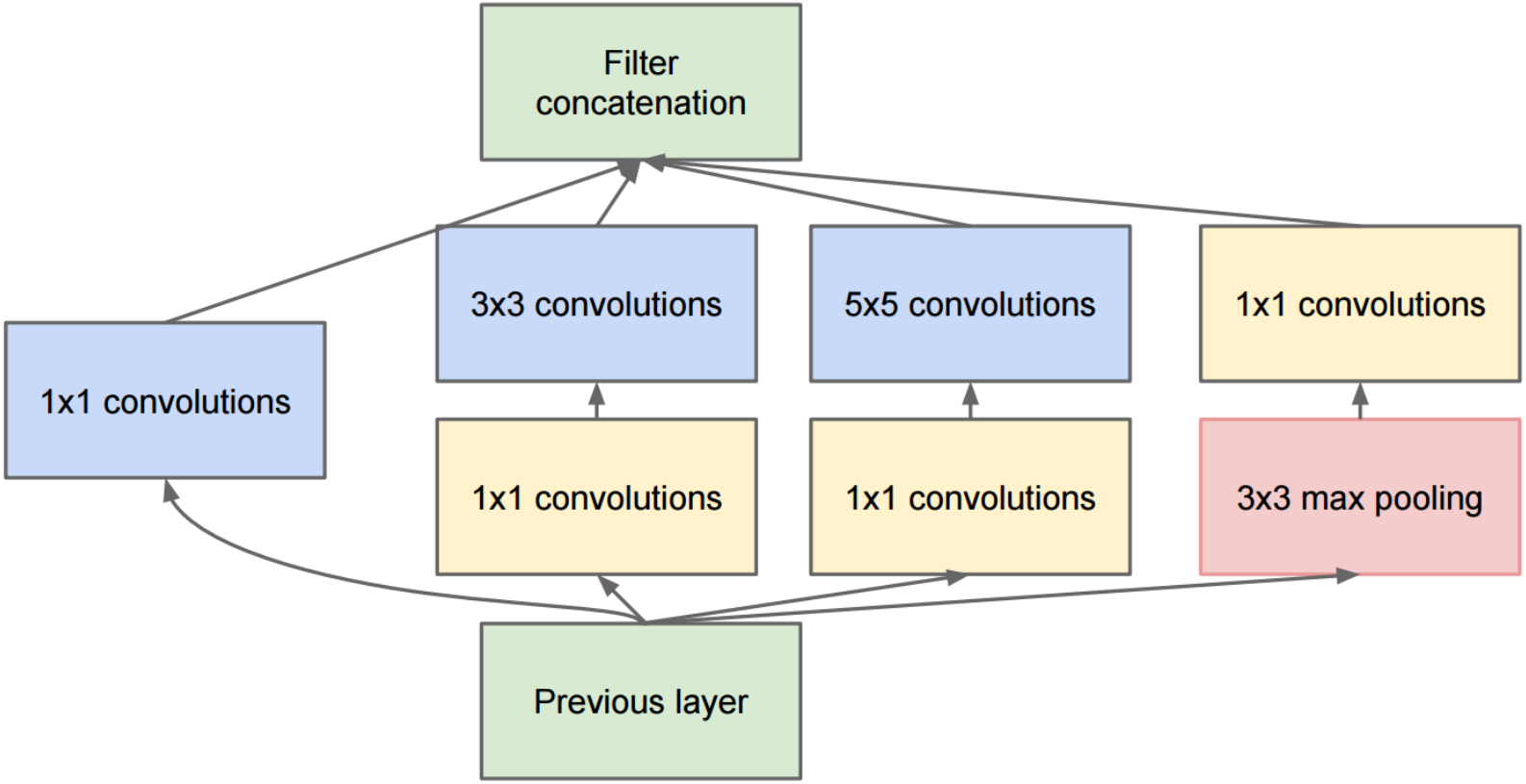
- GoogLeNet
  - ✓핵심은 Inception module



https://medium.com/coinmonks/paper-review-of-googlenet-inception-v1-winner-of-ilsvlc-2014-image-classification-c2b3565a64e7

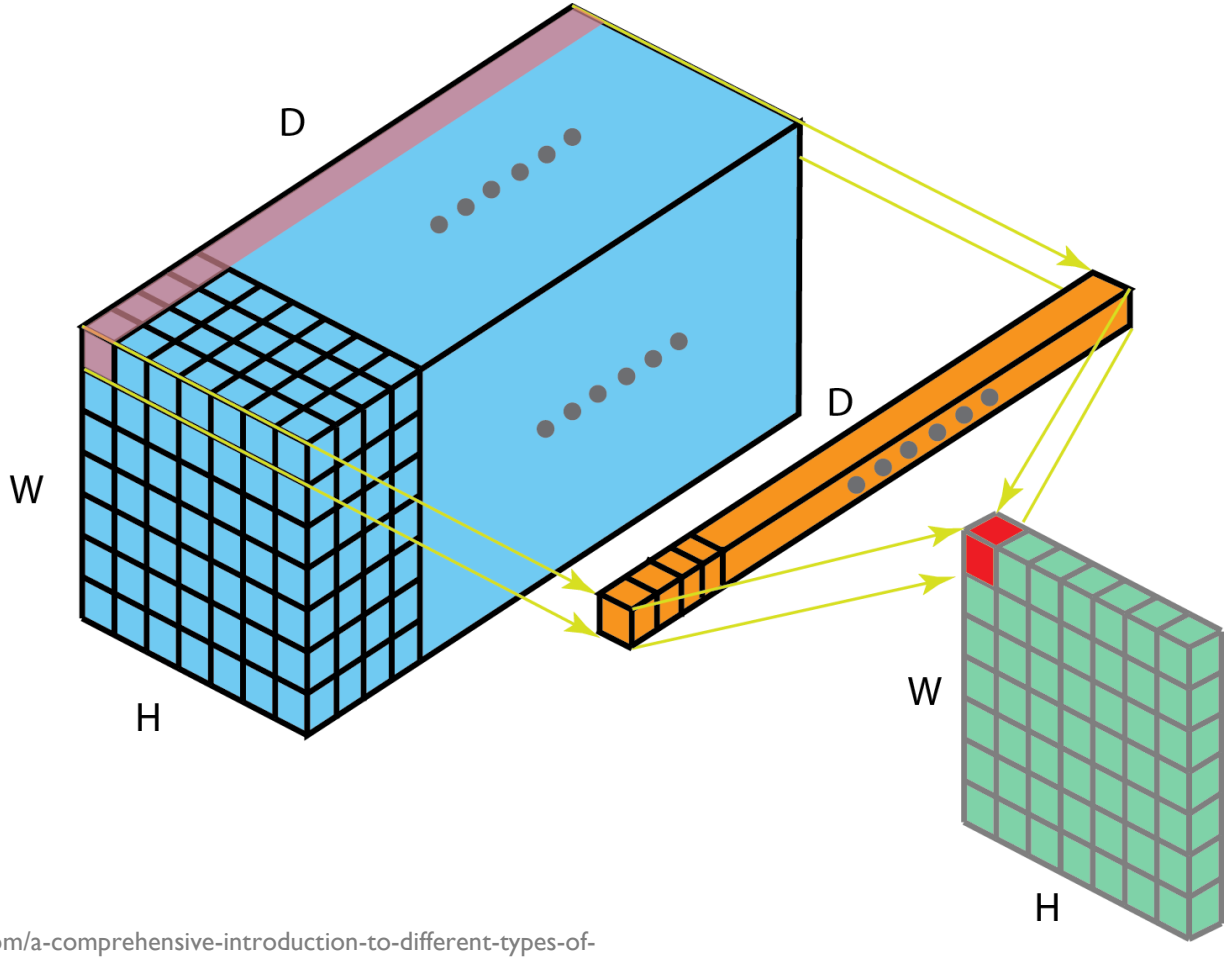
CNN Architecture 3: GoogLeNet

- GoogLeNet
  - ✓ 왜 한 입력 image에 한 종류의 filter size만 적용해야 하는가?
  - ✓ 뭐가 잘 될지 모르니 1 by 1, 3 by 3, 5 by 5, 3 by 3 max pooling을 다 해보자!



CNN Architecture 3: GoogLeNet

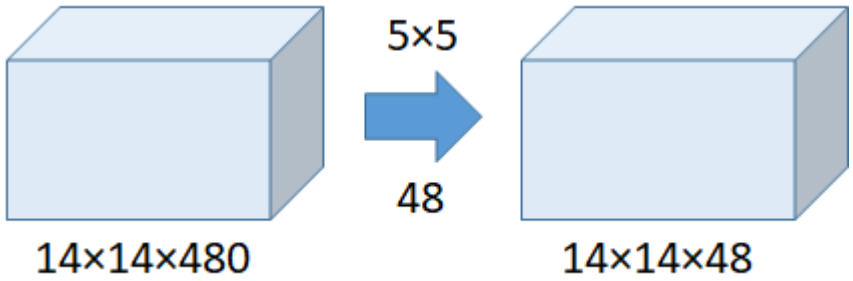
- GoogLeNet
  - ✓ 1 by 1 convolution의 역할: feature map의 depth를 감소시켜 연산량을 감소시킴



https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

CNN Architecture 3: GoogLeNet

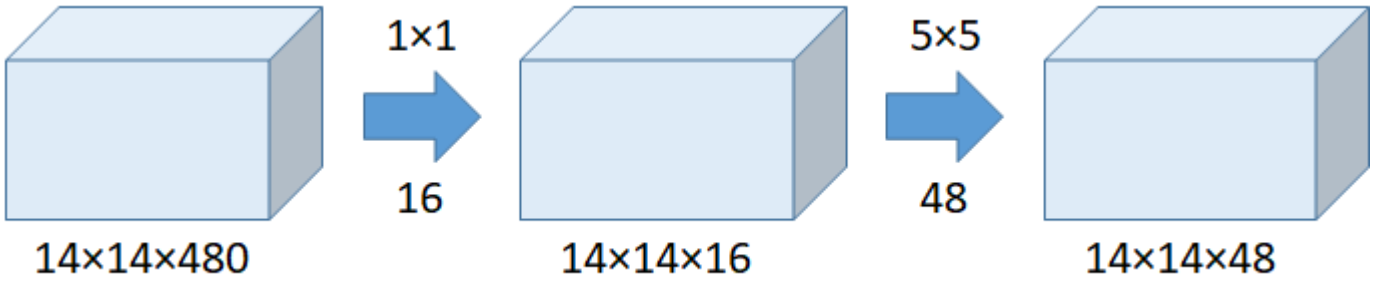
- GoogLeNet
  - ✓ 1 by 1 convolution의 역할: feature map의 depth를 감소시켜 연산량을 감소시킴
  - ✓ 예: 1 by 1 convolution을 수행하지 않고 5 by 5 convolution을 수행



  - 연산량: $(14 \times 14 \times 48) \times (5 \times 5 \times 480) = 112.9\text{M}$

CNN Architecture 3: GoogLeNet

- GoogLeNet
  - ✓ 1 by 1 convolution의 역할: feature map의 depth를 감소시켜 연산량을 감소시킴



  - ✓ 1 by 1 Conv. 연산량: $(14 \times 14 \times 16) \times (1 \times 1 \times 480) = 1.5\text{M}$
  - ✓ 5 by 5 Conv. 연산량: $(14 \times 14 \times 48) \times (5 \times 5 \times 16) = 3.8\text{M}$
  - ✓ 총 연산량: 1.5M + 3.8M = 5.3M << 112.9M

CNN Architecture 3: GoogLeNet

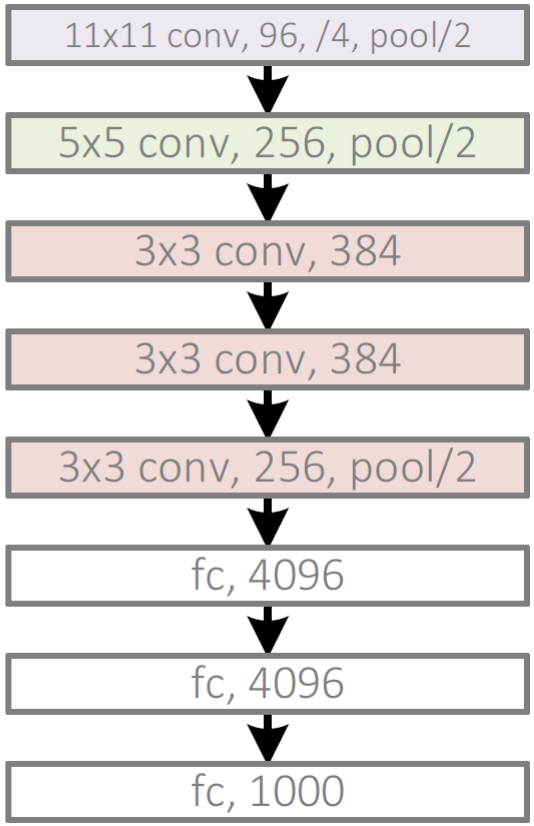
- GoogLeNet

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

CNN Architecture 4: ResNet

- 깊게, 깊게, 더 깊게!

AlexNet, 8 layers
(ILSVRC 2012)

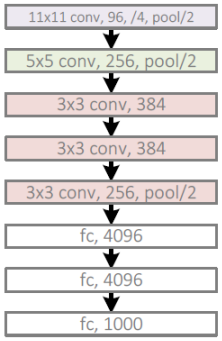


https://icml.cc/2016/tutorials/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf

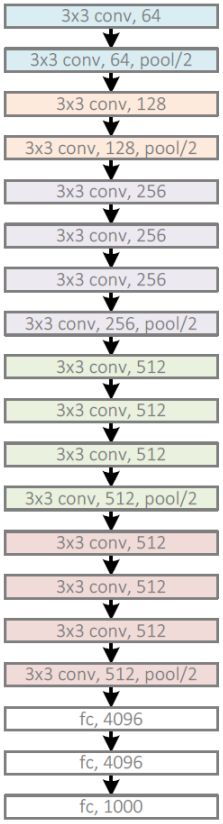
CNN Architecture 4: ResNet

- 깊게, 깊게, 더 깊게!

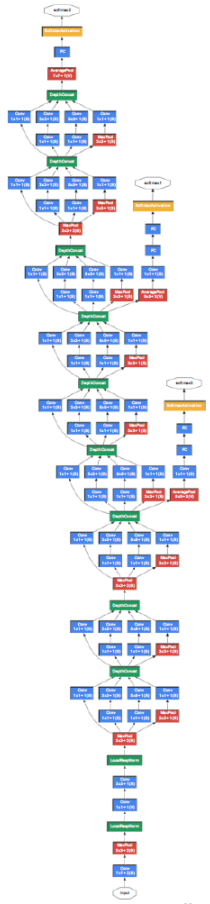
AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)



GoogleNet, 22 layers
(ILSVRC 2014)



CNN Architecture 4: ResNet

- 깊게, 깊게, 더 깊게!

AlexNet, 8 layers
(ILSVRC 2012)

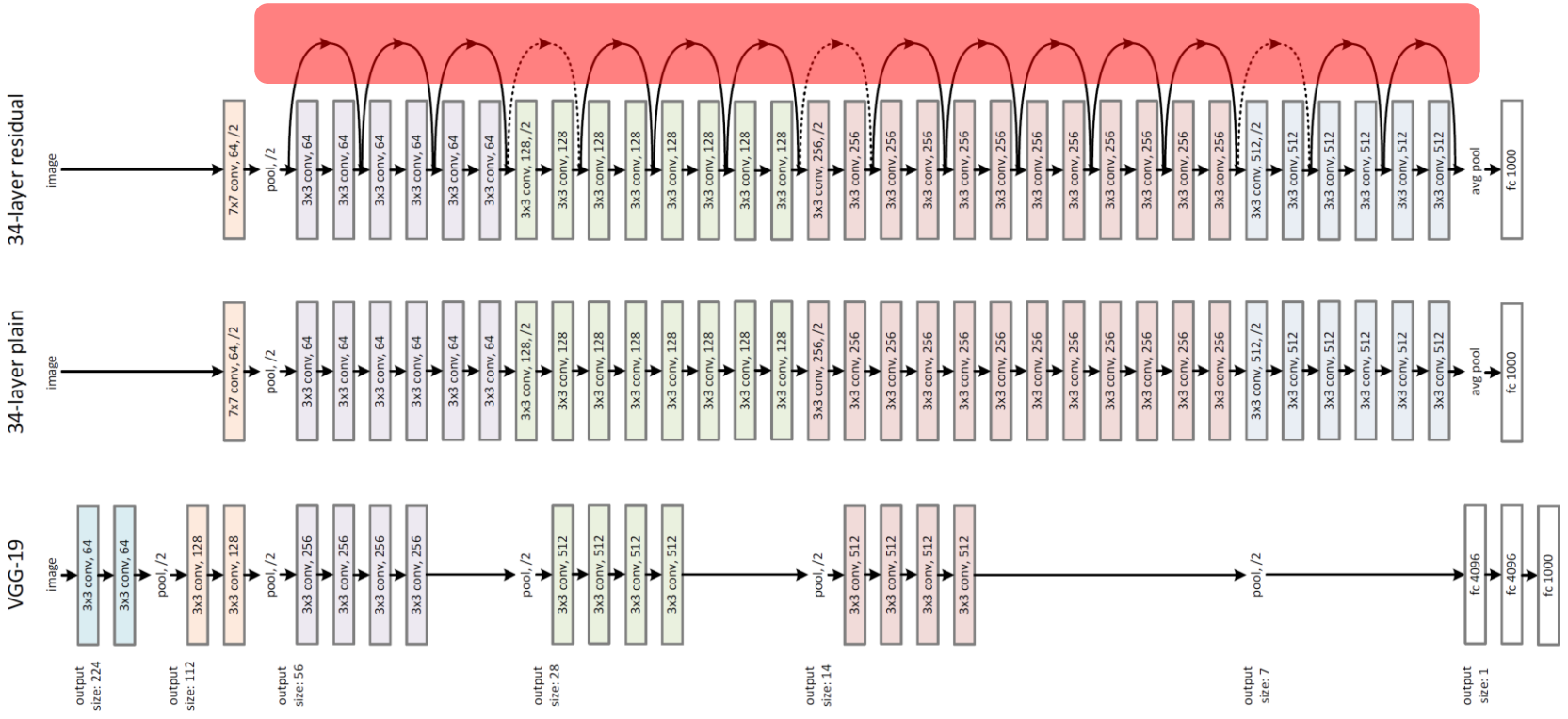
VGG, 19 layers
(ILSVRC 2014)

ResNet, 152 layers
(ILSVRC 2015)

https://icml.cc/2016/tutorials/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf

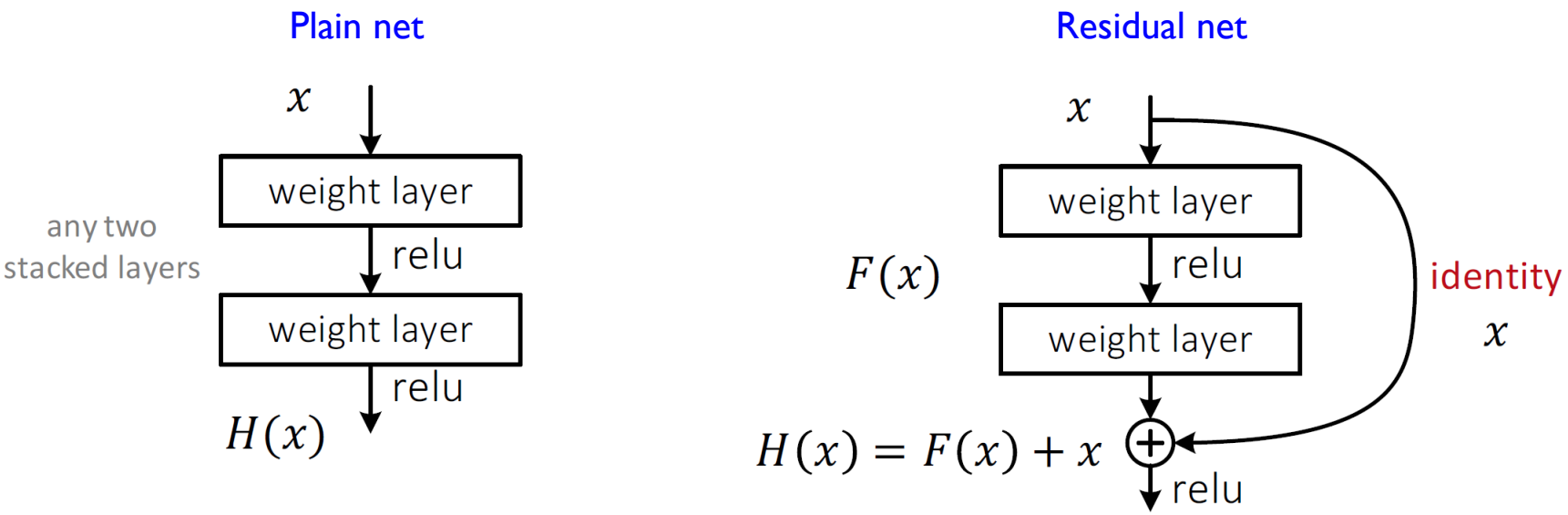
CNN Architecture 4: ResNet

- 핵심은 residual learning



CNN Architecture 4: ResNet

- Residual learning: skip connection

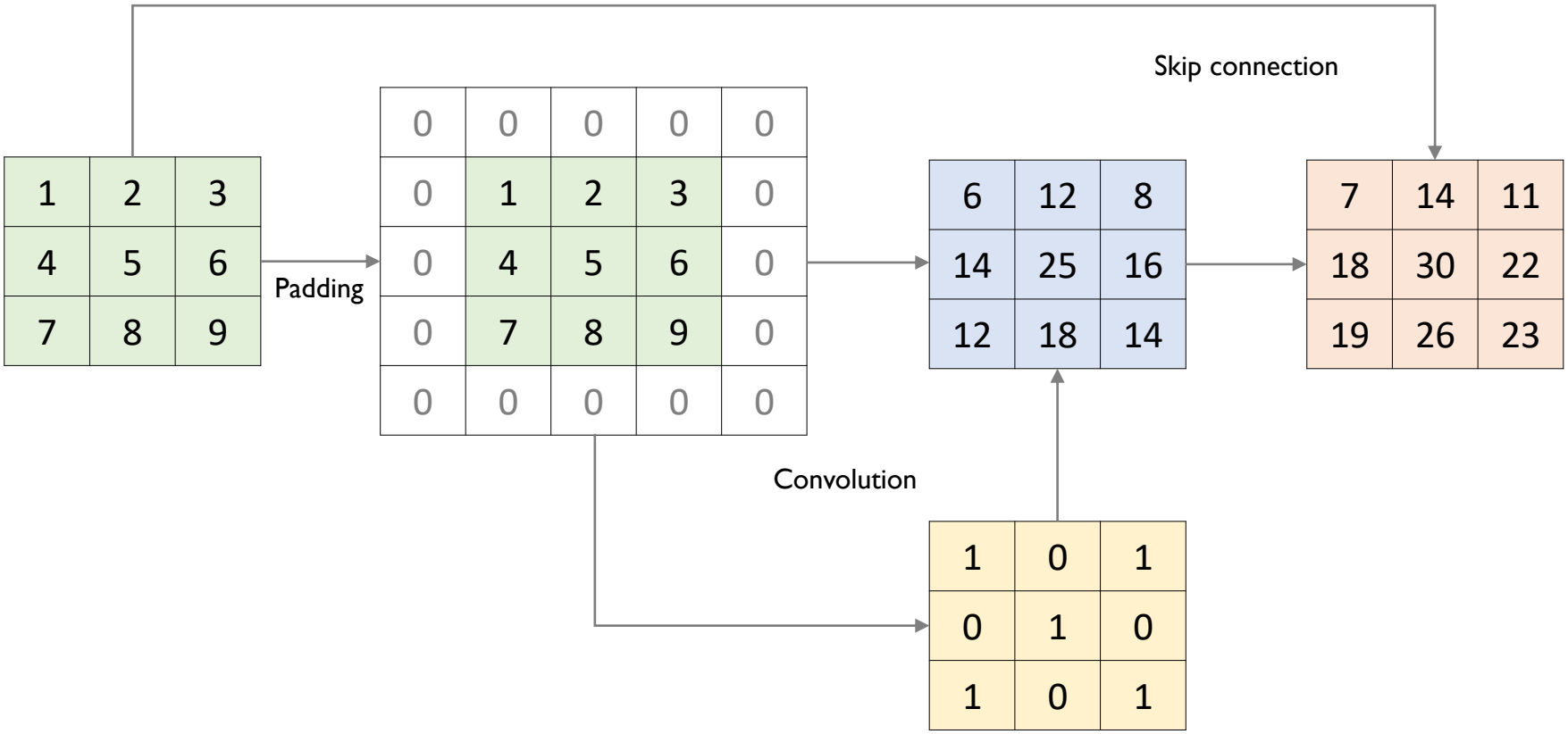


✓ 일정 시점마다 input x 자체를 skip connection을 통해 연결

- Gradient flow가 원활하게 이루어짐 (모델을 깊게 쌓는 것에 대한 부담이 줄어듬)
- Fully connected layer 제거 (모델의 파라미터 수를 줄임)

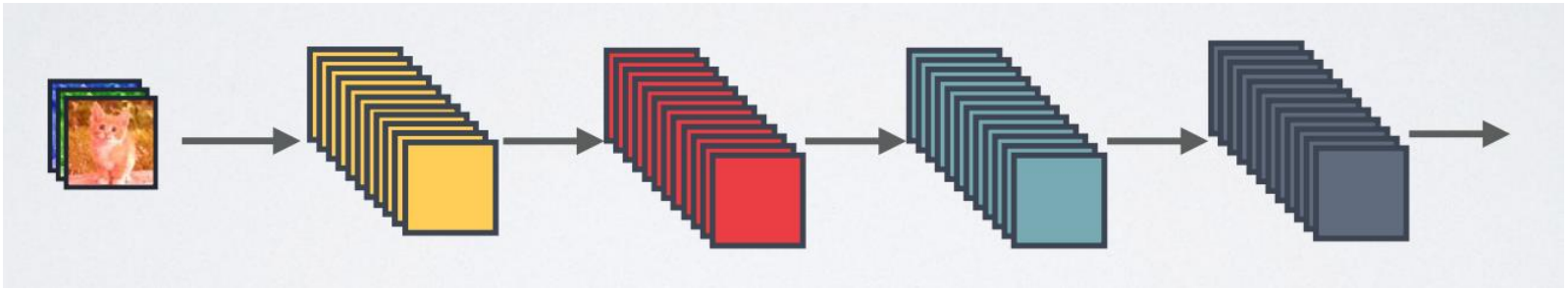
CNN Architecture 4: ResNet

- Residual learning: skip connection example
  - ✓ 3 by 3 convolution

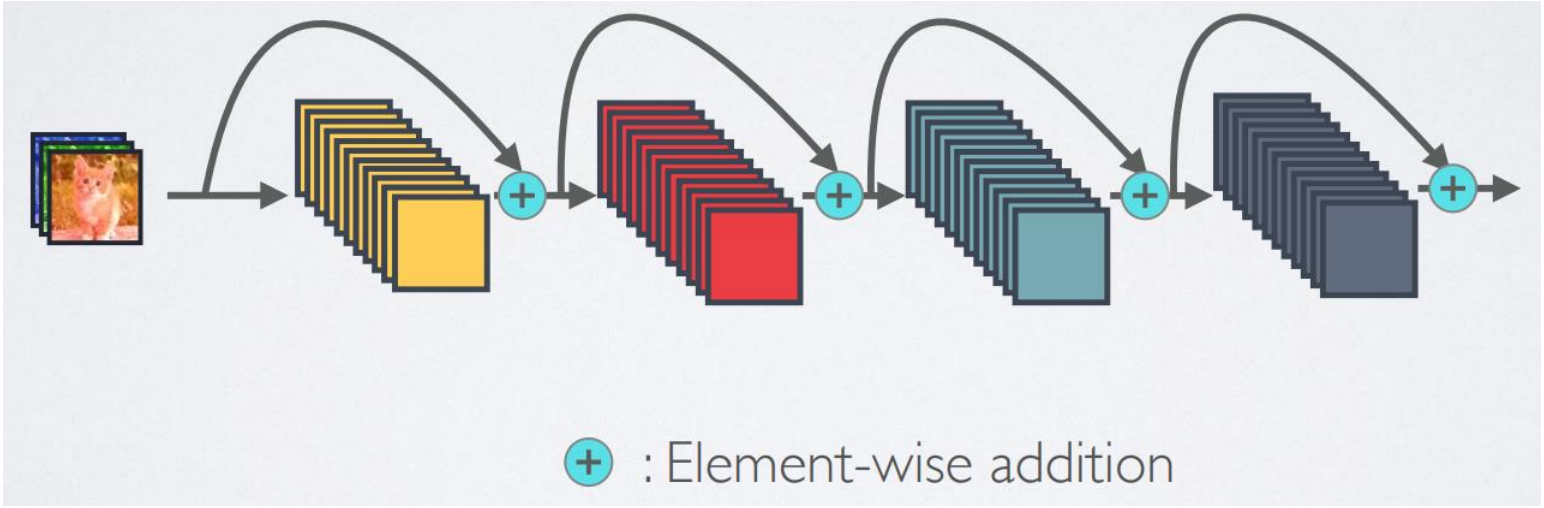


CNN Architecture 5: DenseNet

- Standard Connectivity

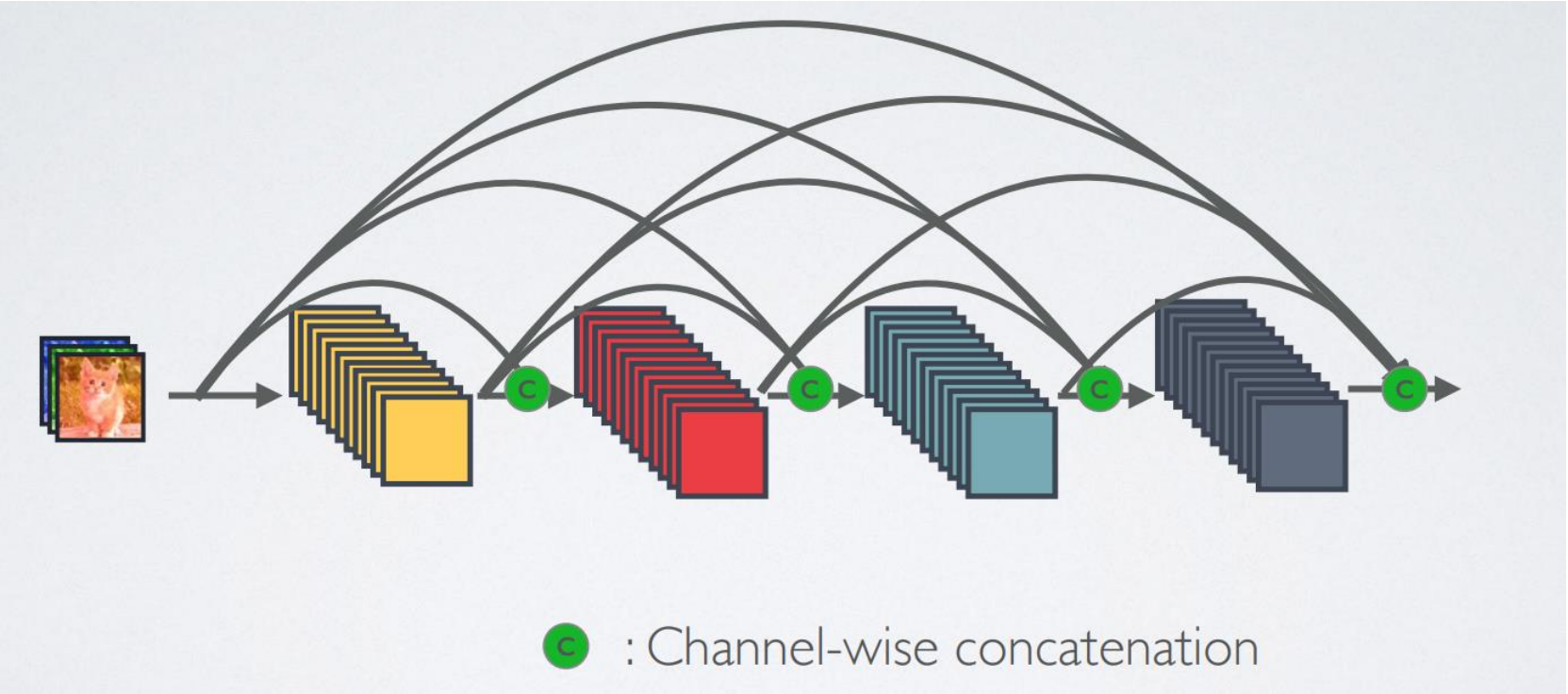


- Resnet Connectivity



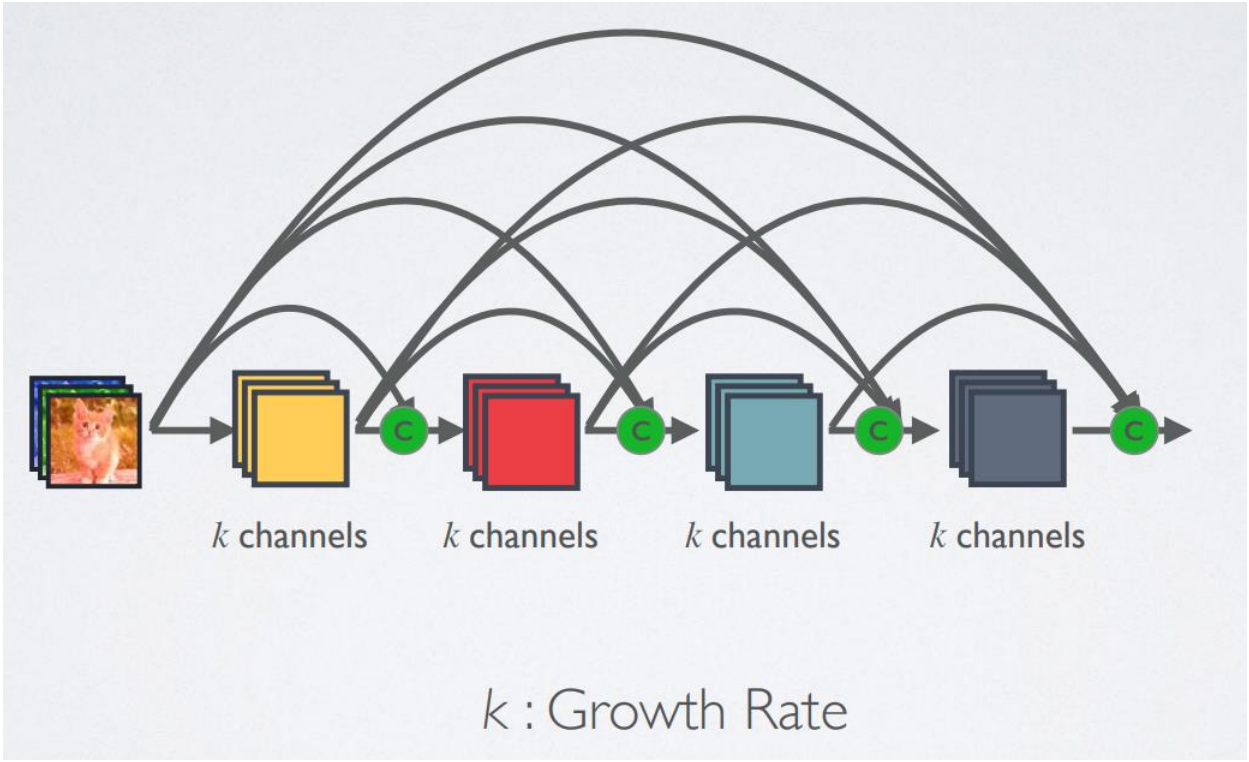
CNN Architecture 5: DenseNet

- Dense Connectivity



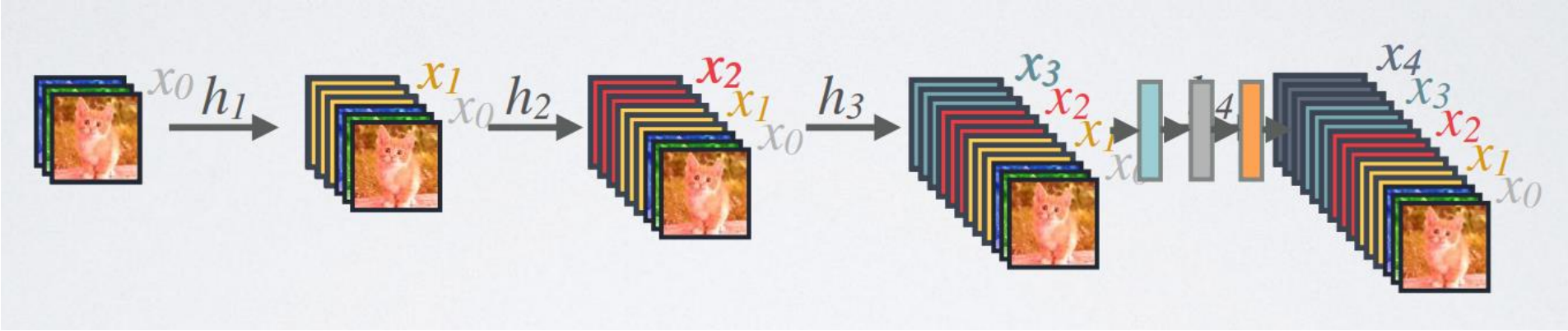
CNN Architecture 5: DenseNet

- Dense & Slim



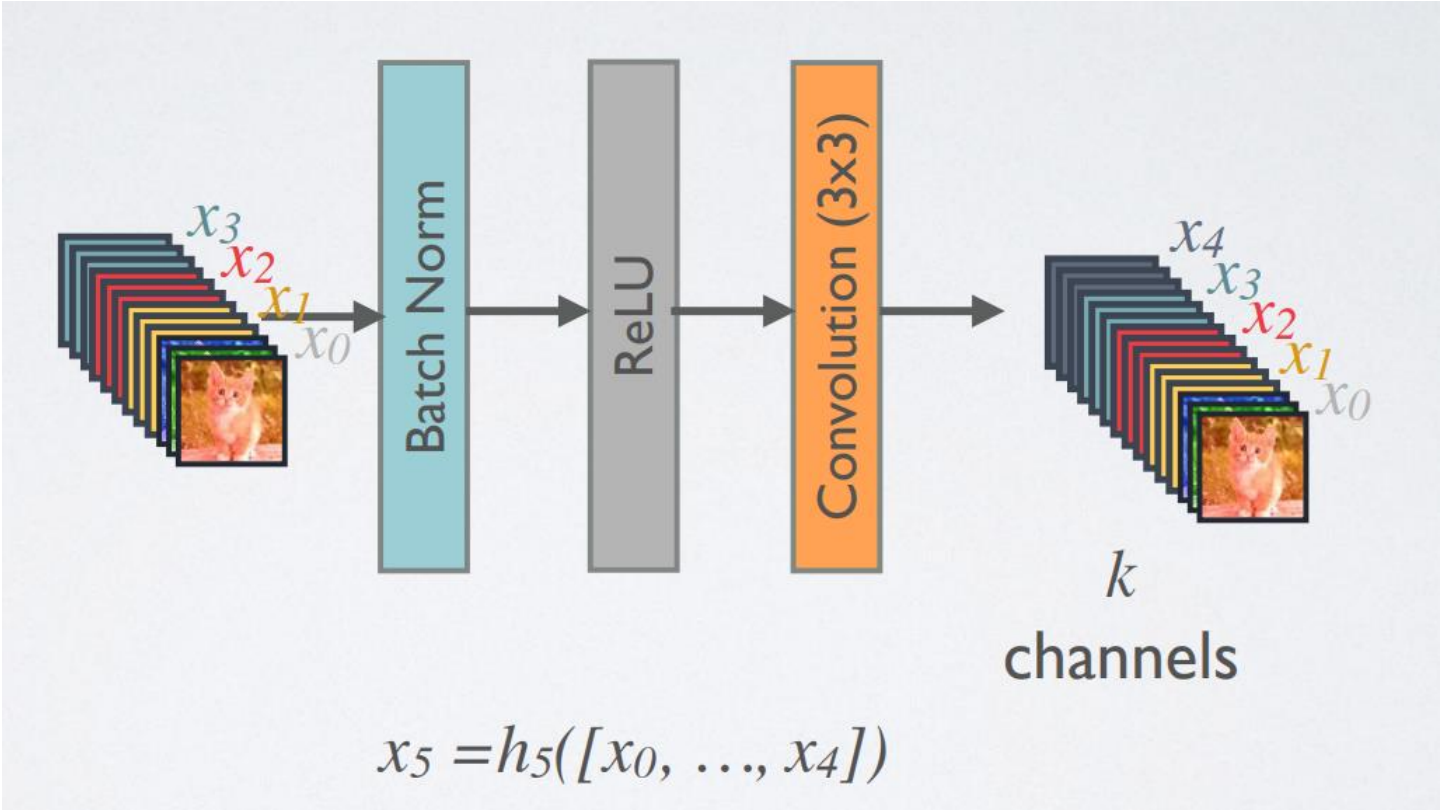
CNN Architecture 5: DenseNet

- Forward Propagation



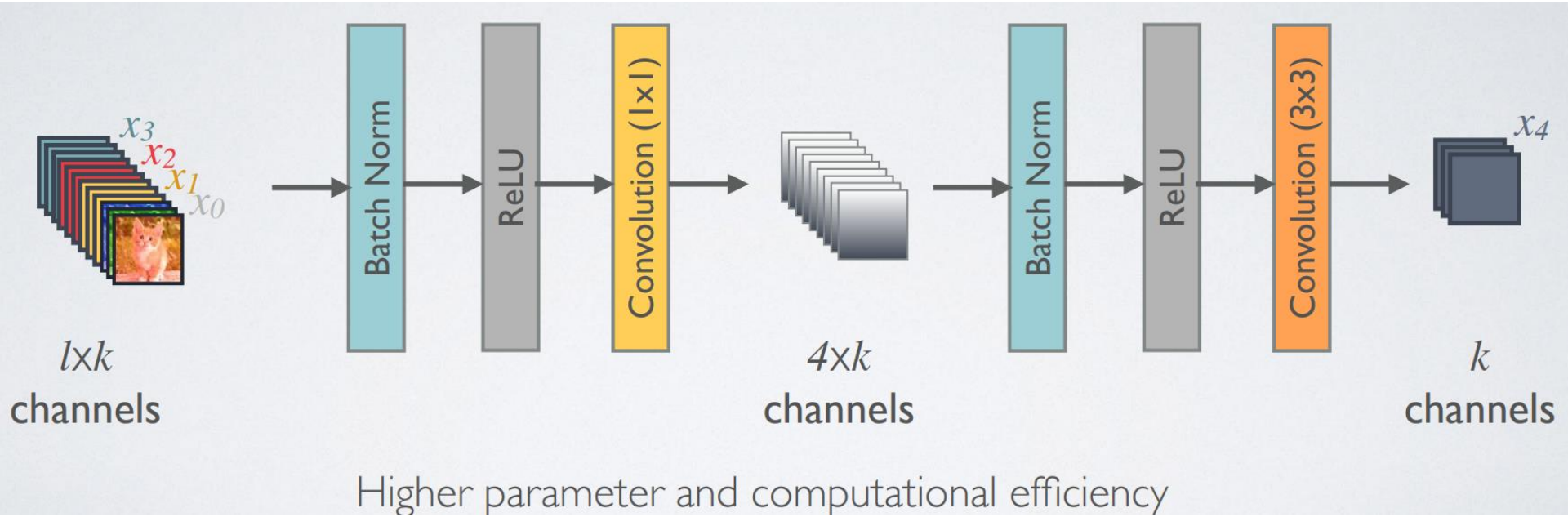
CNN Architecture 5: DenseNet

- Composite layer in Densenet



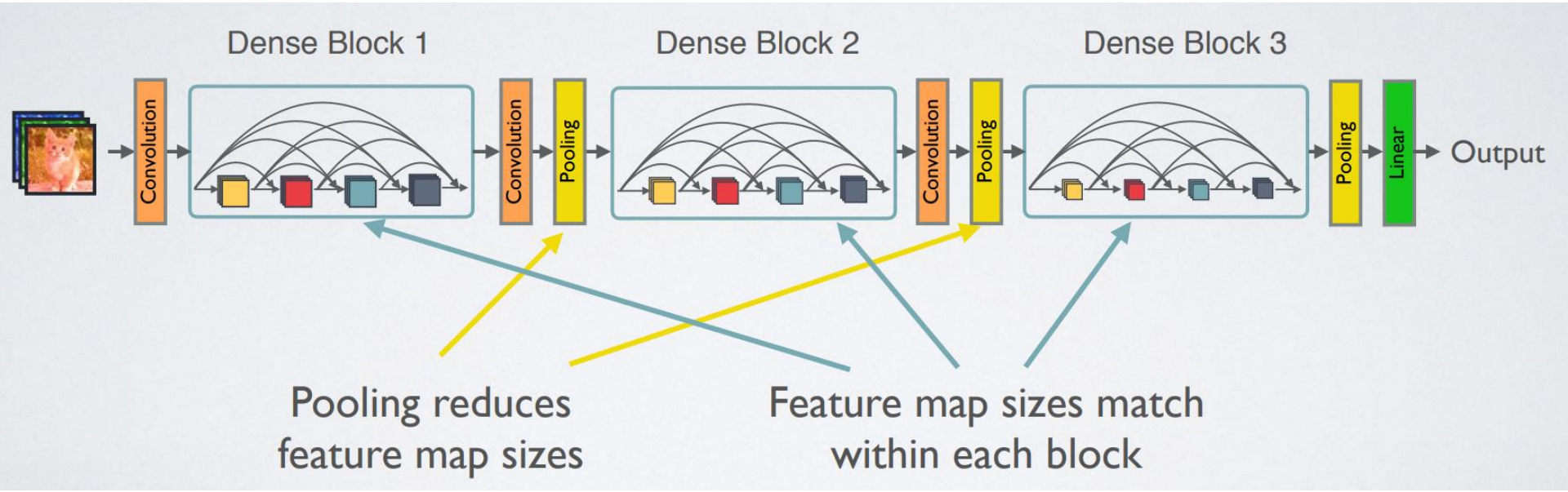
CNN Architecture 5: DenseNet

- Composite layer in Densenet with bottleneck layer



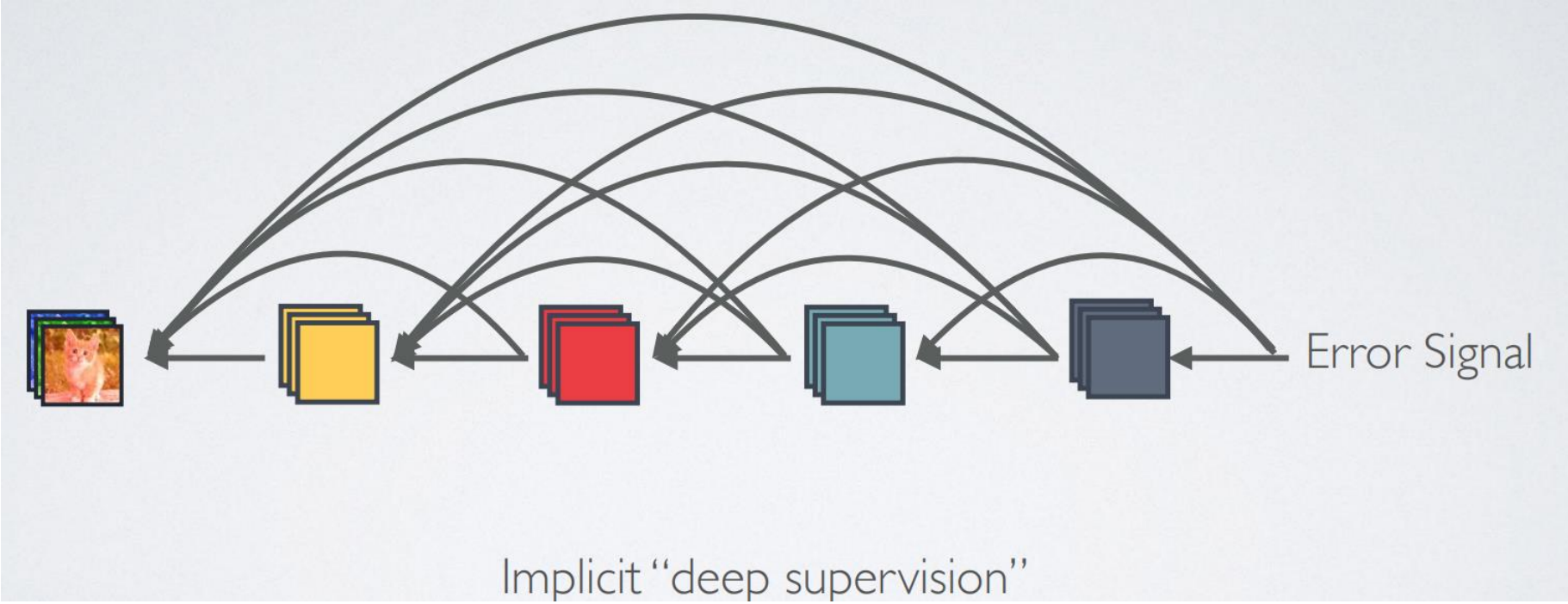
CNN Architecture 5: DenseNet

- Densenet



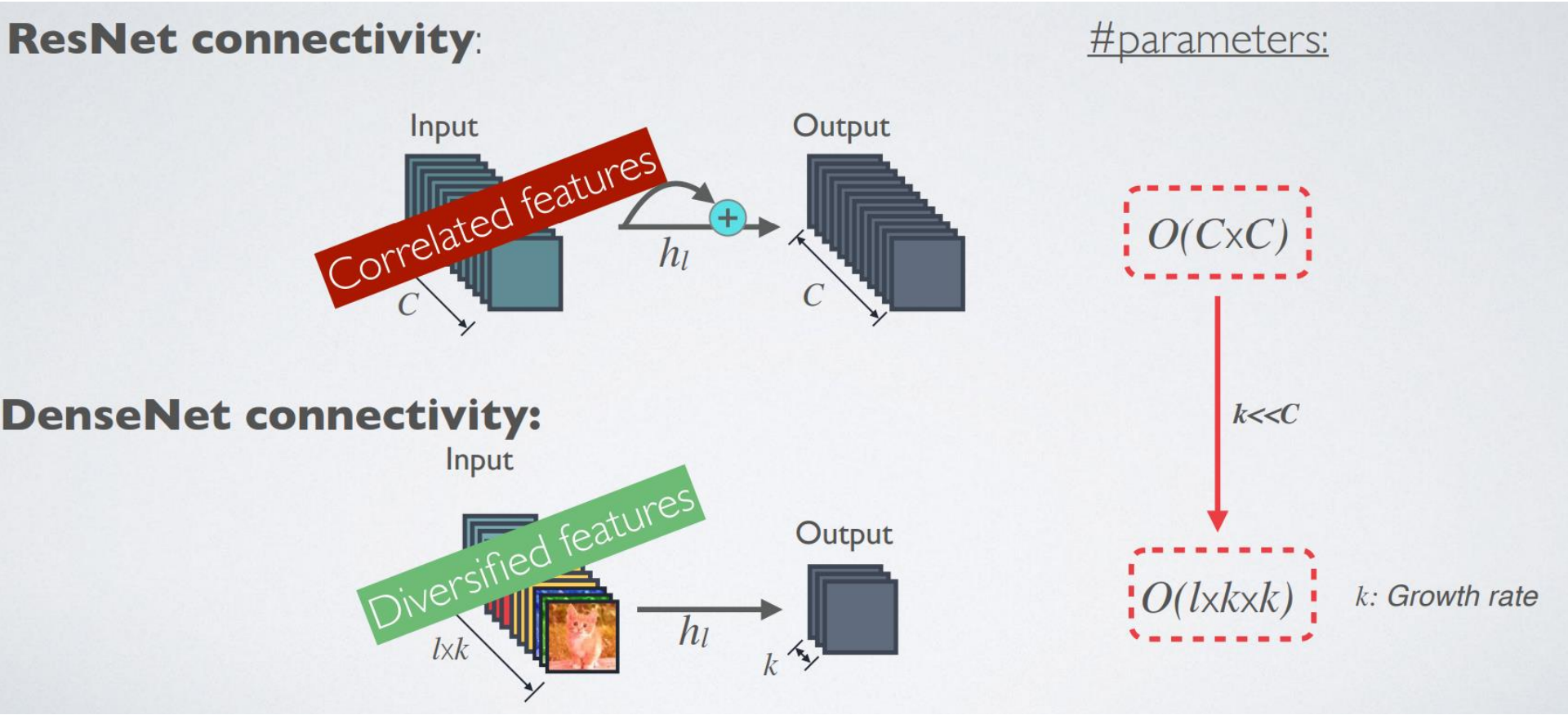
CNN Architecture 5: DenseNet

- Densenet의 장점 1: Gradient flow가 잘 된다



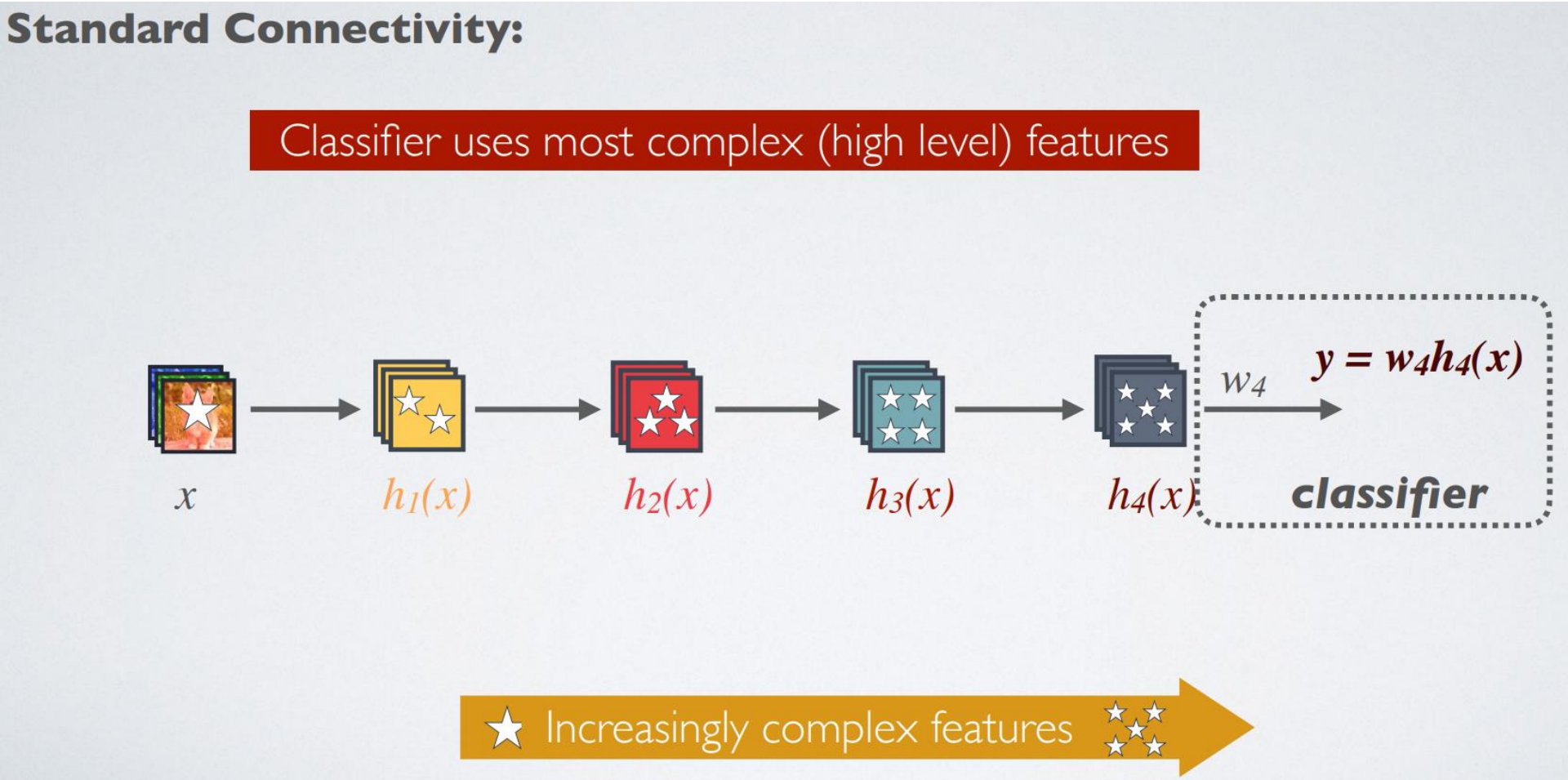
CNN Architecture 5: DenseNet

- Densenet의 장점 2: 파라미터의 수가 적다



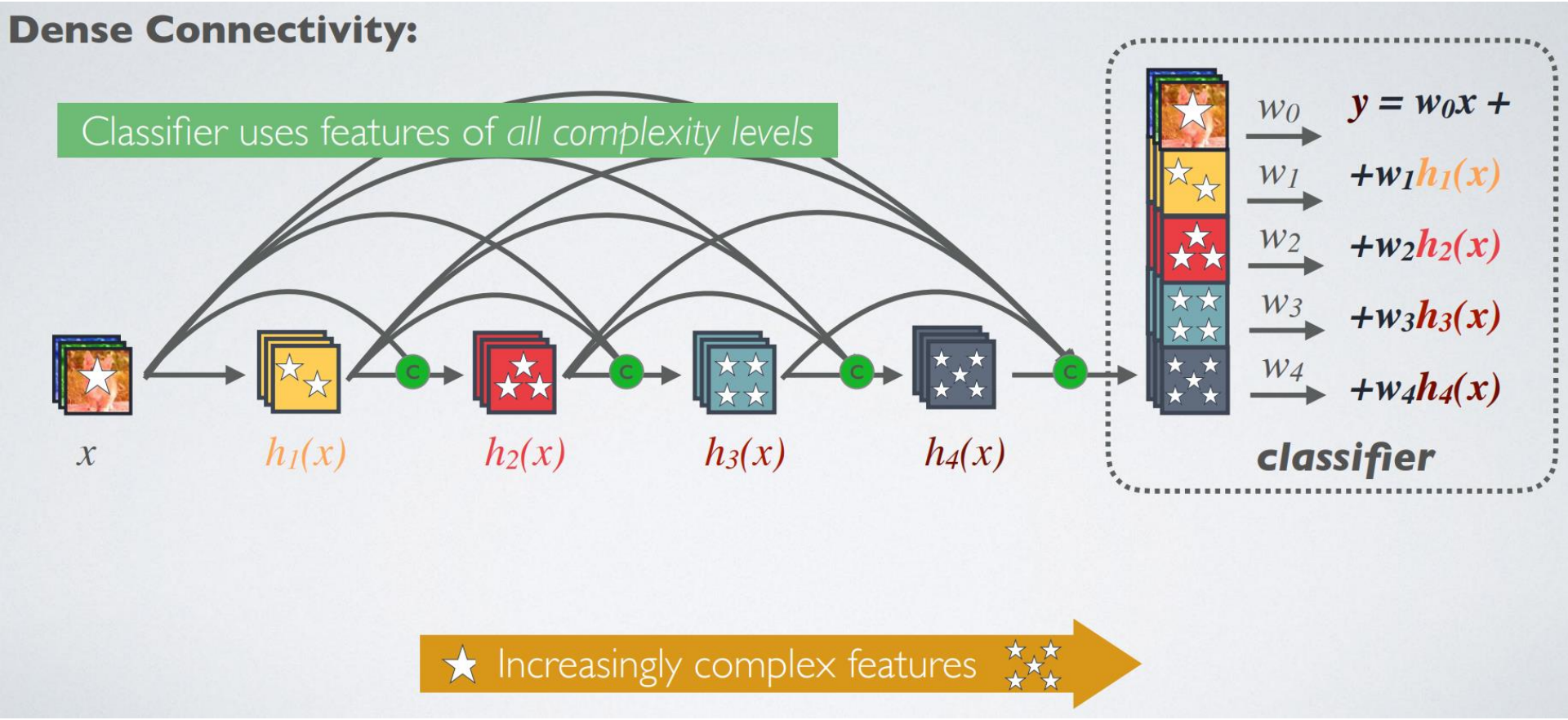
CNN Architecture 5: DenseNet

- Densenet의 장점 3: Feature 수준의 다양성이 보장된다



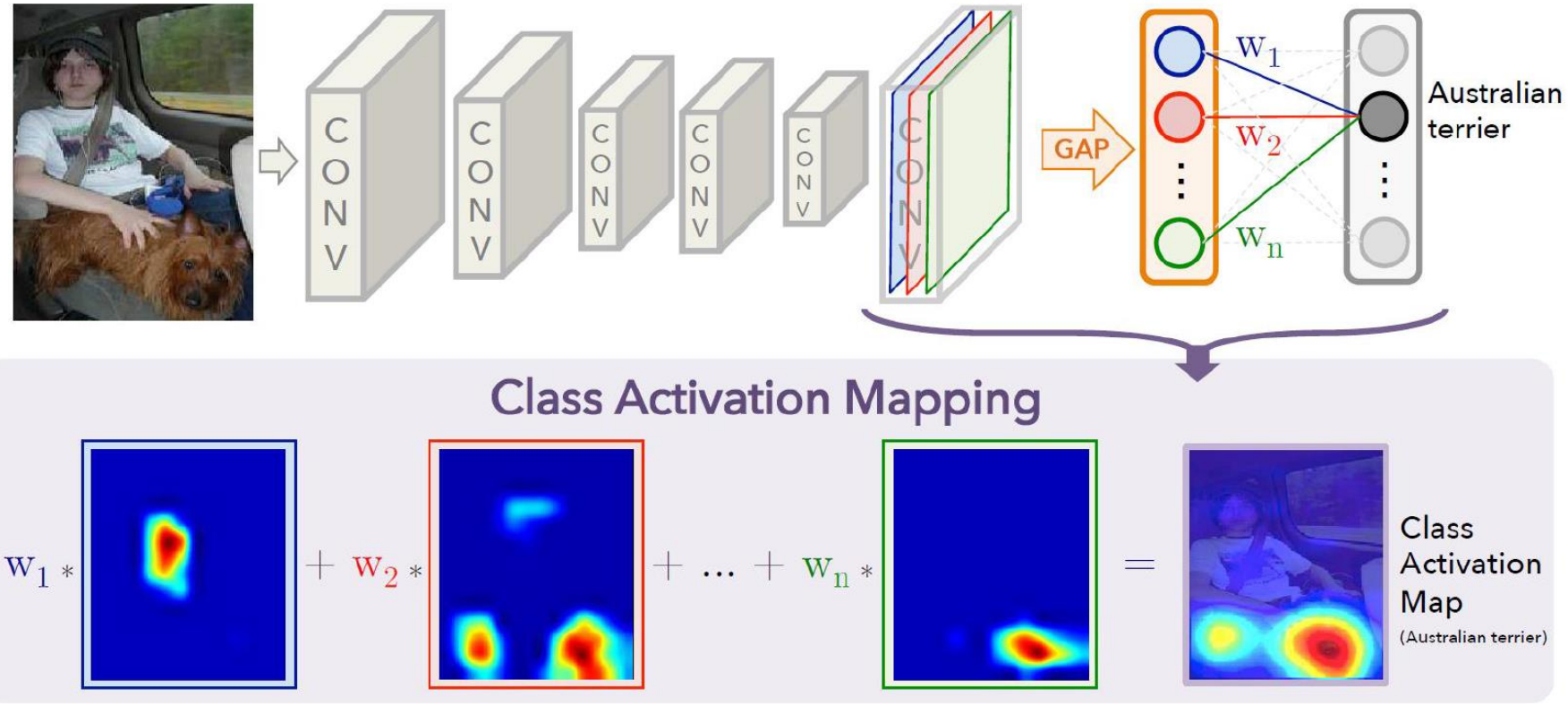
CNN Architecture 5: DenseNet

- Densenet의 장점 3: Feature 수준의 다양성이 보장된다



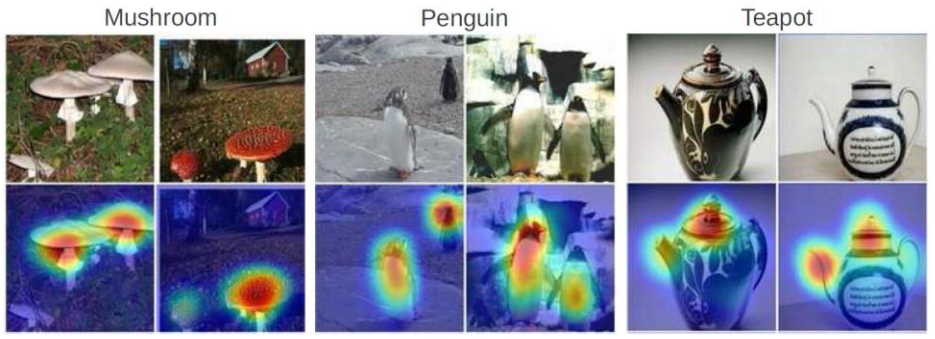
CNN Localization

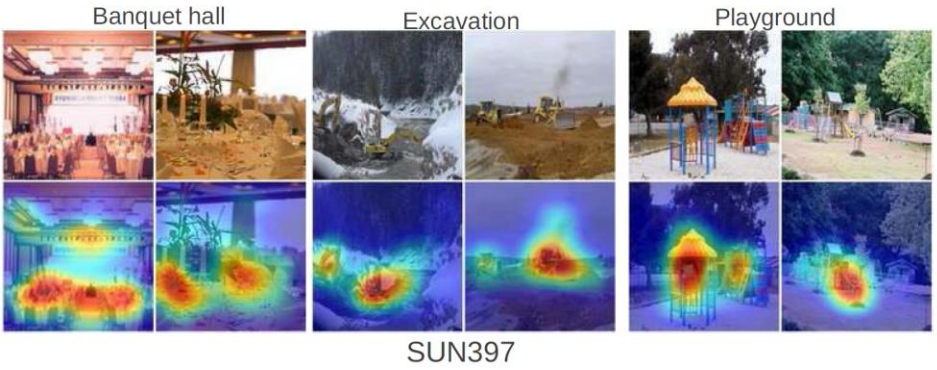
- Image Localization
  - ✓ Class Activation Map (CAM) (Zhou et al., 2016)
    - Localize significant areas based only on class label information

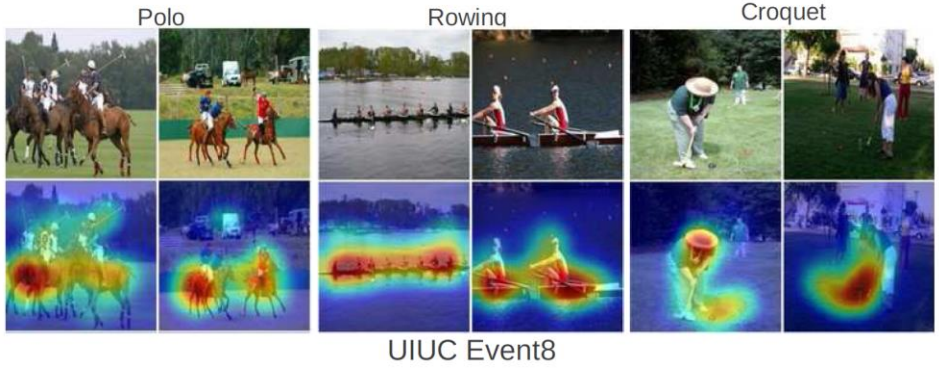


CNN Localization

- Image Localization
  - ✓ Class Activation Map (CAM)
    - 예측한 분류 범주에 가장 큰 영향을 끼치는 이미지의 영역을 추정





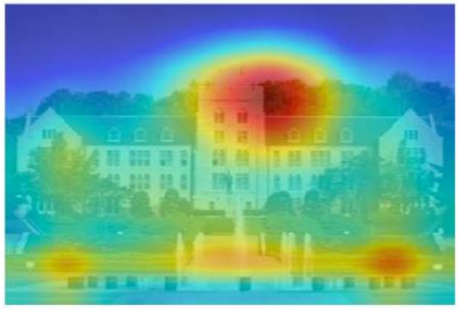


CNN Localization

- Image Localization
  - ✓ Class Activation Map (CAM)
    - 예측한 분류 범주에 가장 큰 영향을 끼치는 이미지의 영역을 추정

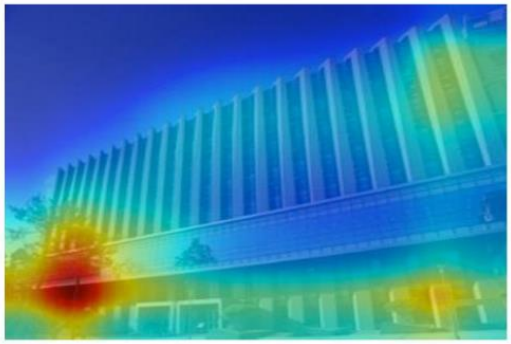
Predictions:

- **Type of environment:** outdoor
- **Semantic categories:** palace:0.23, formal_garden:0.20, mansion:0.15, castle:0.07, courthouse:0.06
- **SUN scene attributes:** man-made, openarea, naturallight, grass, vegetation, foliage, leaves, directsunsunny, trees, vacationingtouring
- **Informative region for the category *palace* is:**



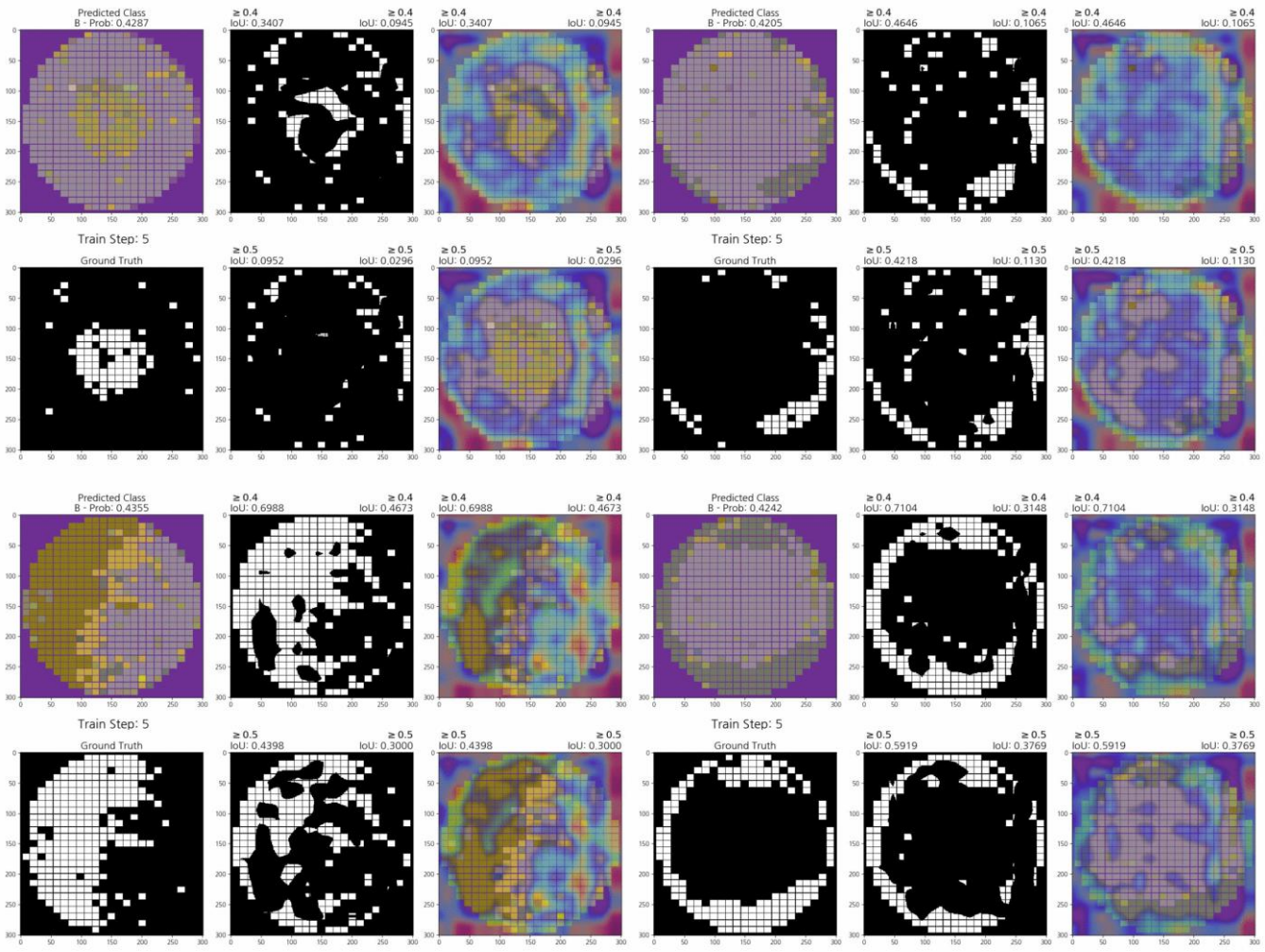
Predictions:

- **Type of environment:** outdoor
- **Semantic categories:** office_building:0.33, building_facade:0.24, hospital:0.17, skyscraper:0.06,
- **SUN scene attributes:** man-made, naturallight, openarea, mostlyverticalcomponents, directsunsunny, clouds, glass, mostlyhorizontalcomponents, semi-enclosedarea, metal
- **Informative region for the category *office_building* is:**



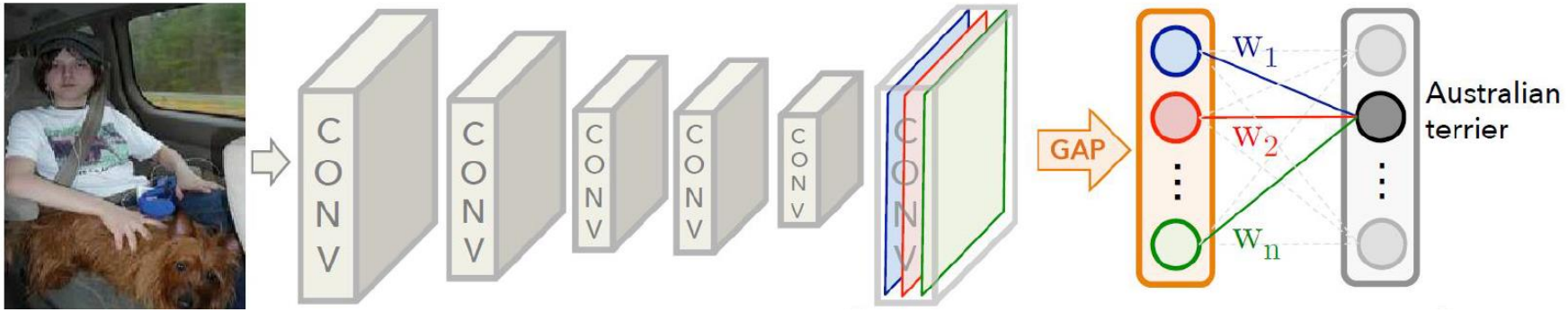
CNN Localization

- Image Localization:WBM에서의 불량 원인 영역 추정



Class Activation Map

- CAM 작동 원리



✓ $f_k(x, y)$: 마지막 convolution layer unit k의 activation (x, y)는 공간적 좌표

✓ Global average pooling 수행:

$$F_k = \sum_{x,y} f_k(x, y)$$

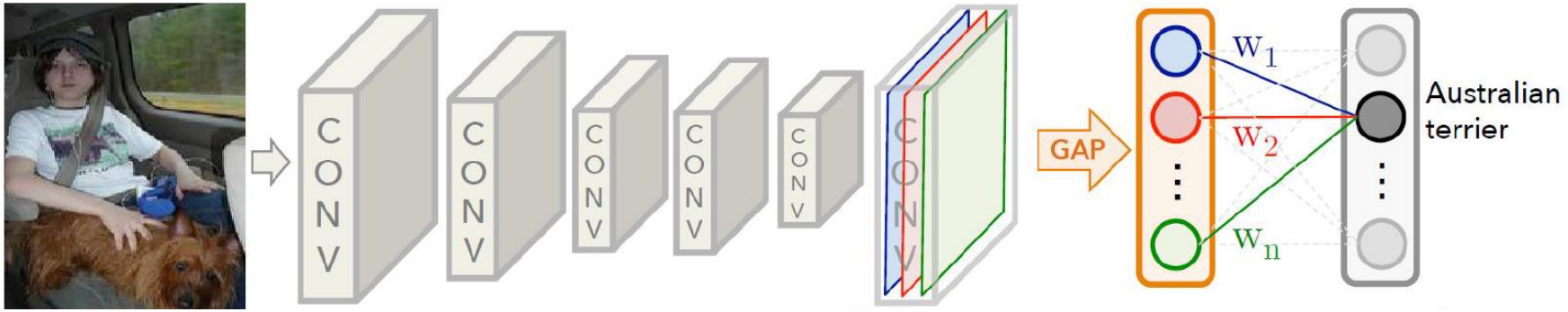
✓ 범주 c에 대한 softmax 함수 계산 전 입력값

$$S_c = \sum_k w_k^c F_k$$

- w_k^c : Unit k의 범주 c에 대한 가중치

Class Activation Map

- CAM 작동 원리



✓ 범주 c에 대한 softmax 함수의 출력

$$\frac{\exp(S_c)}{\sum_c \exp(S_c)}$$

✓ S_c는 다음과 같이 표현될 수 있고, M_c를 c범주에 대한 class activation map으로 정의할 수 있음

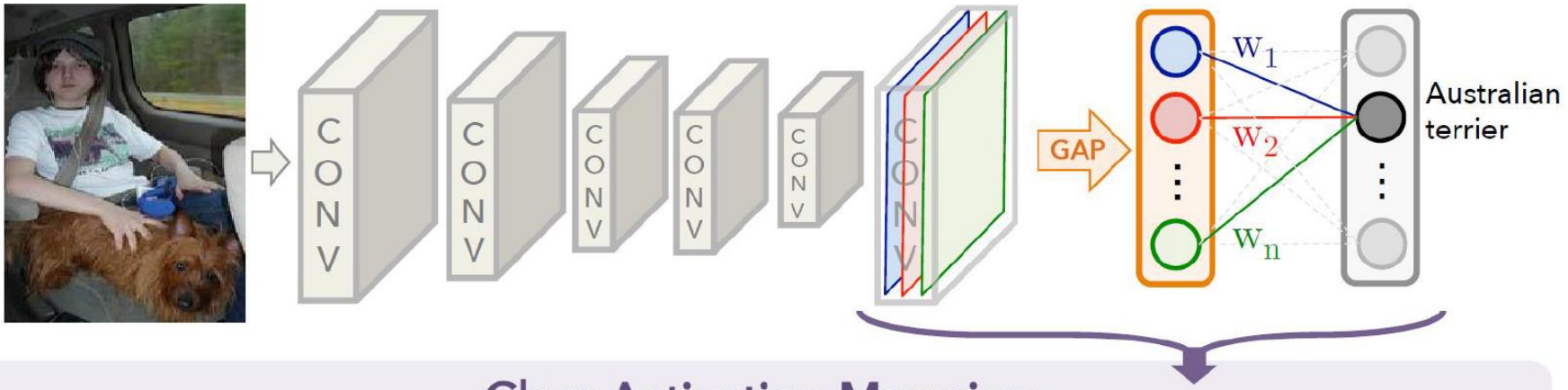
$$S_c = \sum_k w_k^c \sum_{x,y} f_k(x,y) = \sum_{x,y} \sum_k w_k^c f_k(x,y)$$

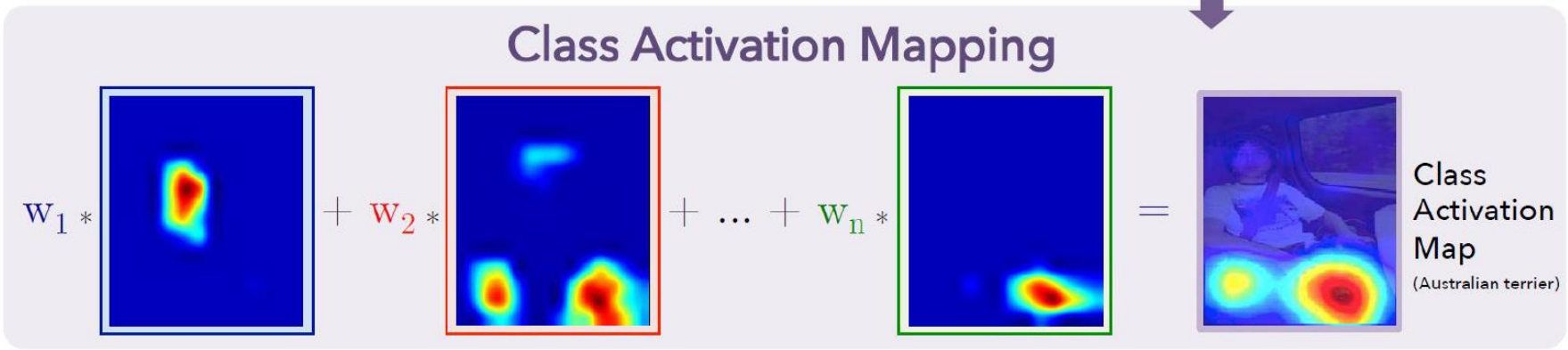
$$M_c(x,y) = \sum_k w_k^c f_k(x,y)$$

Class Activation Map

- CAM 작동 원리

원본 이미지의 크기(I)와 feature map(F)의 크기가
다르기 때문에 (일반적으로 I > F) upsampling을 수행






ANY
questions?