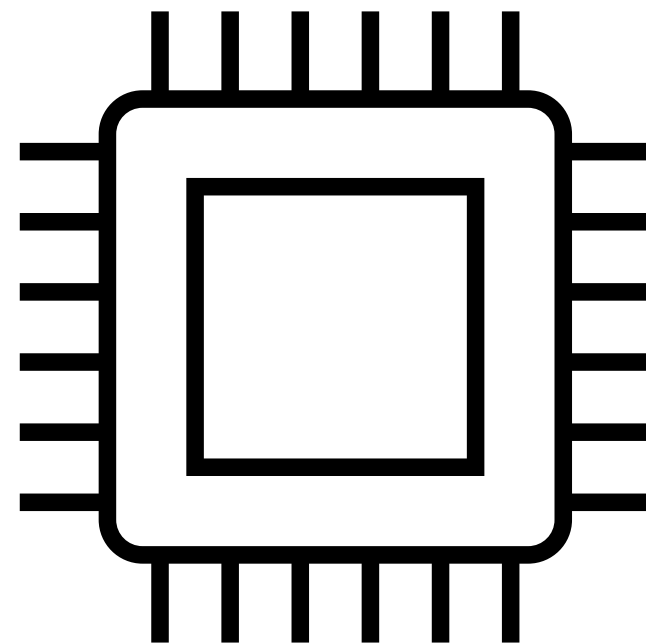


Image Classification

(이미지 분류)

2025.07.08.

Copyright©2025 by 고재균



High-Level Vision (이미지 분류) [1]

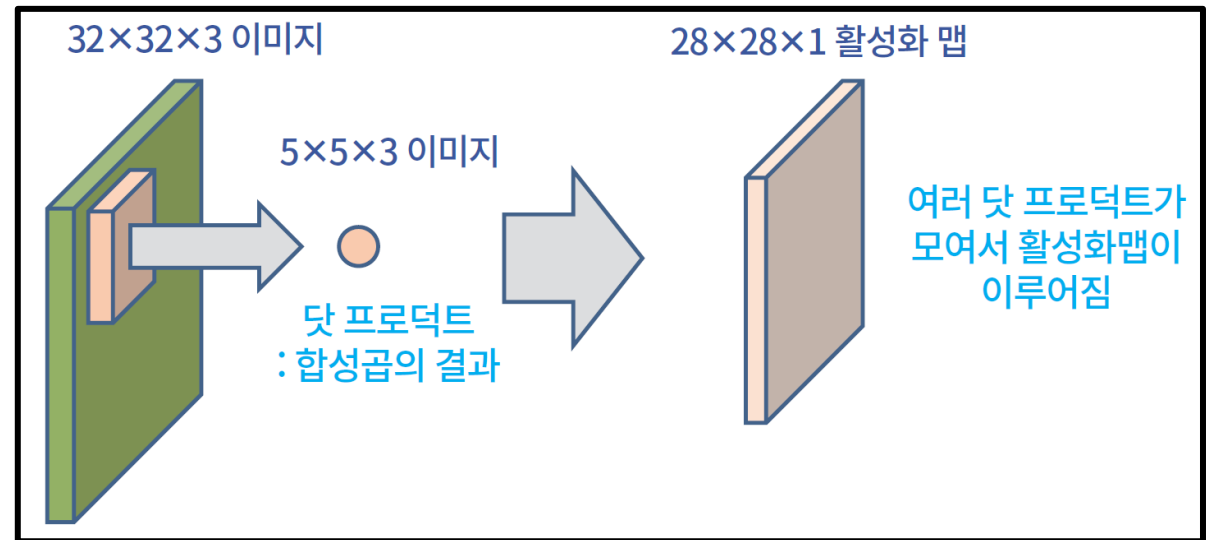
• Convolutional Operation

▪ 연산 과정

- ▶ 중심 및 그에 인접한 픽셀들과 2차원 또는 3차원 배열 형태의 가중치를 곱하고 더함
- ▶ 해당 연산을 통해 얻은 값을 토대로 중심 픽셀의 값을 변경

▪ 배열을 칭하는 용어

- ▶ 마스크 (Mask)
- ▶ 커널 (Kernel)
- ▶ 필터 (Filter)
- ▶ 템플릿 (Template)



High-Level Vision (이미지 분류) [2]

- Convolutional Operation

- 수식

$$x^{new}_{ij} = \sum_{a=0}^m \sum_{b=0}^m \omega_{ab} x^{old}_{(i+a)(j+a)}$$

where

$x^{old} \rightarrow \text{Input}$

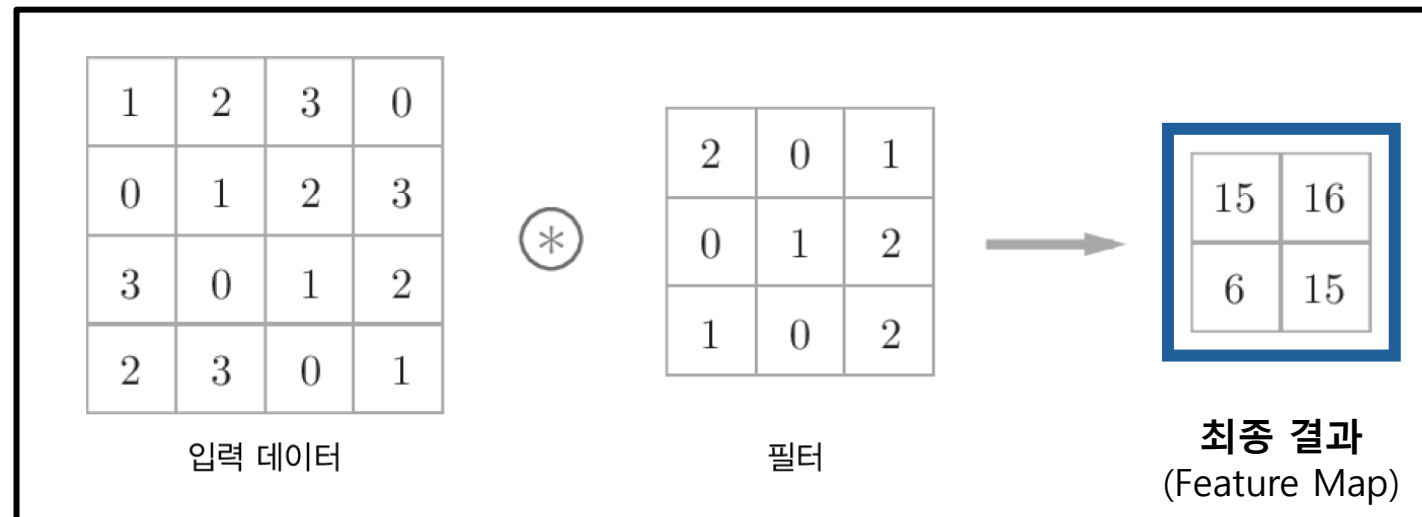
$x^{new} \rightarrow \text{output}$

$\text{Kernel } \omega \rightarrow m \times m \text{ Matrix}$

High-Level Vision (이미지 분류) [3]

- Convolutional Operation

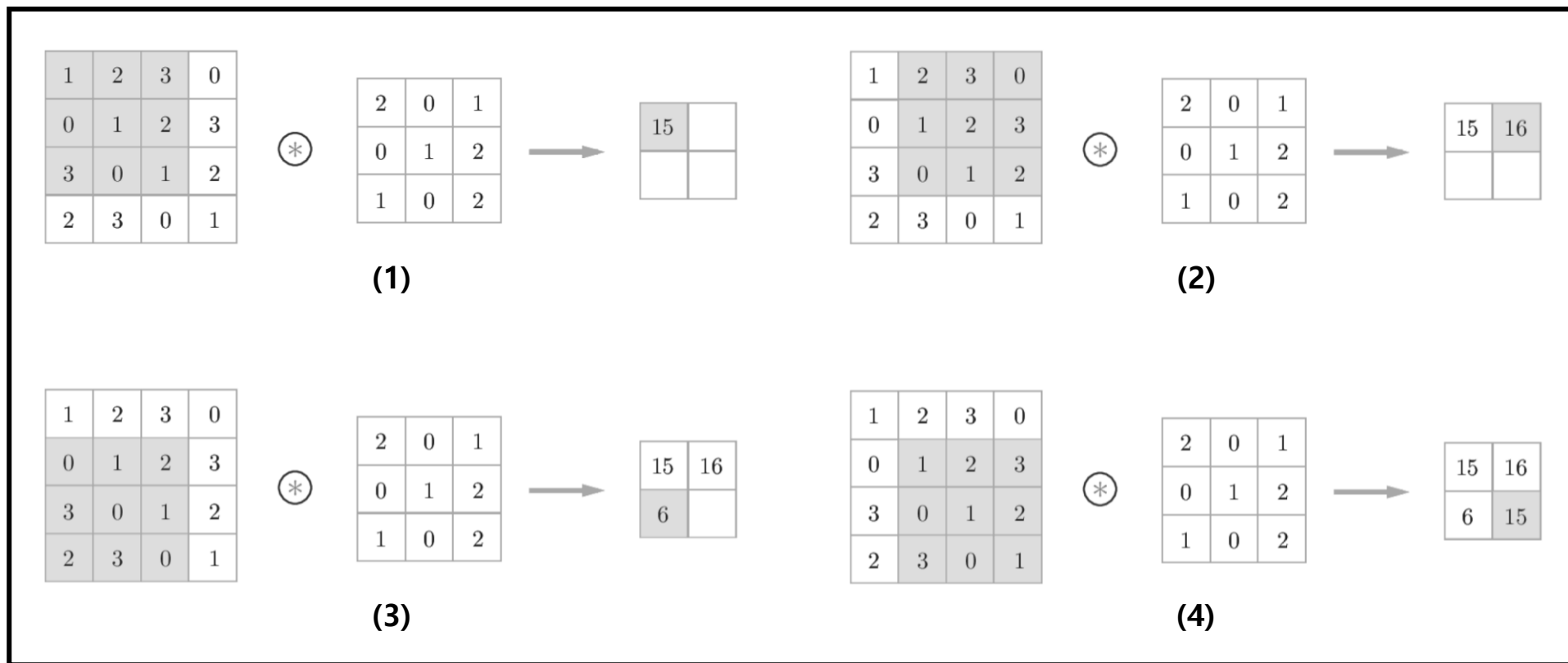
- 전반적인 연산 과정 [1]



High-Level Vision (이미지 분류) [4]

- Convolutional Operation

- 전반적인 연산 과정 [2]

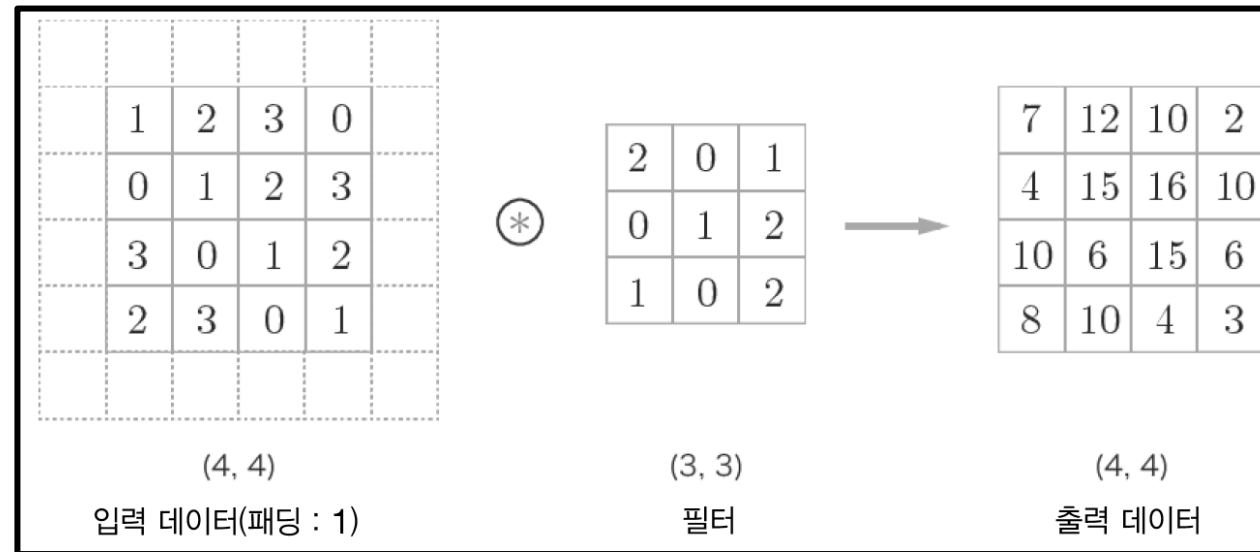


High-Level Vision (이미지 분류) [5]

• Convolutional Operation

▪ Padding

- ▶ 합성곱 연산을 수행하기 전에 입력 데이터 주변에 특정 값을 채우는 과정
- ▶ 출력 크기를 조절하기 위해 사용
- ▶ 꼭지점에 있는 Pixel이 한번만 Convolve 되는 현상을 방지
- ▶ 주로 Zero 또는 Reflect 패딩을 사용



High-Level Vision (이미지 분류) [6]

• Convolutional Operation

▪ Stride

- ▶ 필터를 적용하는 위치의 간격
- ▶ 출력 크기를 조절하기 위해 사용 가능

1	2	3	0	1	2	3
0	1	2	3	0	1	2
3	0	1	2	3	0	1
2	3	0	1	2	3	0
1	2	3	0	1	2	3
0	1	2	3	0	1	2
3	0	1	2	3	0	1

⊗

2	0	1
0	1	2
1	0	2



15		

(a) Stride = 1

스트라이드 : 2

1	2	3	0	1	2	3
0	1	2	3	0	1	2
3	0	1	2	3	0	1
2	3	0	1	2	3	0
1	2	3	0	1	2	3
0	1	2	3	0	1	2
3	0	1	2	3	0	1

⊗

2	0	1
0	1	2
1	0	2



15	17	

(b) Stride = 2

High-Level Vision (이미지 분류) [7]

- Convolutional Operation

- 입력 크기와 출력 크기에 대한 관계식
 - ▶ 입력 크기 : $H \times W$
 - ▶ 출력 크기 : $OH \times OW$
 - ▶ 필터 크기 : $FH \times FW$
 - ▶ 패딩 : P
 - ▶ 스트라이드 : S

$$OH = \frac{H + 2P - FH}{S} + 1$$

$$OW = \frac{W + 2P - FW}{S} + 1$$

High-Level Vision (이미지 분류) [8]



• Convolutional Operation (Rule-Based)

- 이미지 처리 (Image Processing)에서의 합성곱 연산

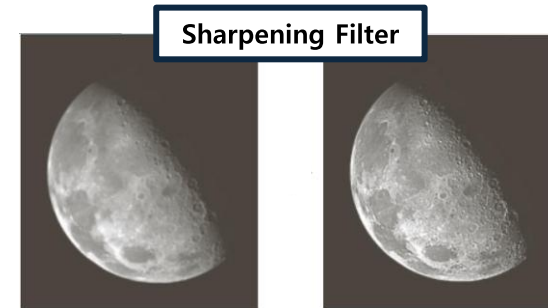
▶ Image Smoothing / Blurring

- I. 이미지를 부드럽게 표현하거나 노이즈 (Noise)를 제거하기 위해 사용
- II. Mean Filtering, Median Filtering 등 다양한 필터를 통해 Low-Pass Filter 구현



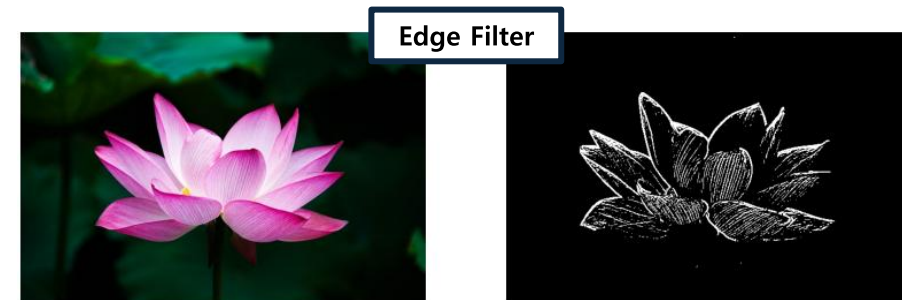
▶ Image Sharpening

- I. 이미지의 상세한 정보를 강화하기 위해 사용 (High-Pass Filter)
- II. 이미지의 상세한 정보 → 이미지 내부에 포함된 물체의 경계 혹은 질감 (Texture)의 경계 등을 의미



▶ Edge Detection

- I. 이미지 내에 포함된 경계선을 추출하기 위해 사용
- II. 경계선 → 짧은 범위에서 급격하게 밝기가 변하는 지점을 의미 (0 - 255)



High-Level Vision (이미지 분류) [9]

• Convolutional Operation (Rule-Based)

- 이미지 처리 (Image Processing)에서의 합성곱 연산

▶ Image Smoothing / Blurring

- Mean Filter → 모든 계수가 양수이고, 전체 합이 1인 필터를 사용

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \frac{1}{10} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \frac{1}{16} \begin{bmatrix} 2 & 1 & 2 \\ 1 & 4 & 1 \\ 2 & 1 & 2 \end{bmatrix} \quad \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

▶ Image Sharpening

- 자기 자신이 돋보여야 하는 경우임으로, 필터 주변의 값들과 대비되도록 신호를 변경
- 원래의 픽셀 값을 n배, 주변의 Pixel 값을 -1배 하여 합이 1이 되는 필터를 구현

-1	-1	-1
-1	9	-1
-1	-1	-1

Sharpening Mask
(n=9)

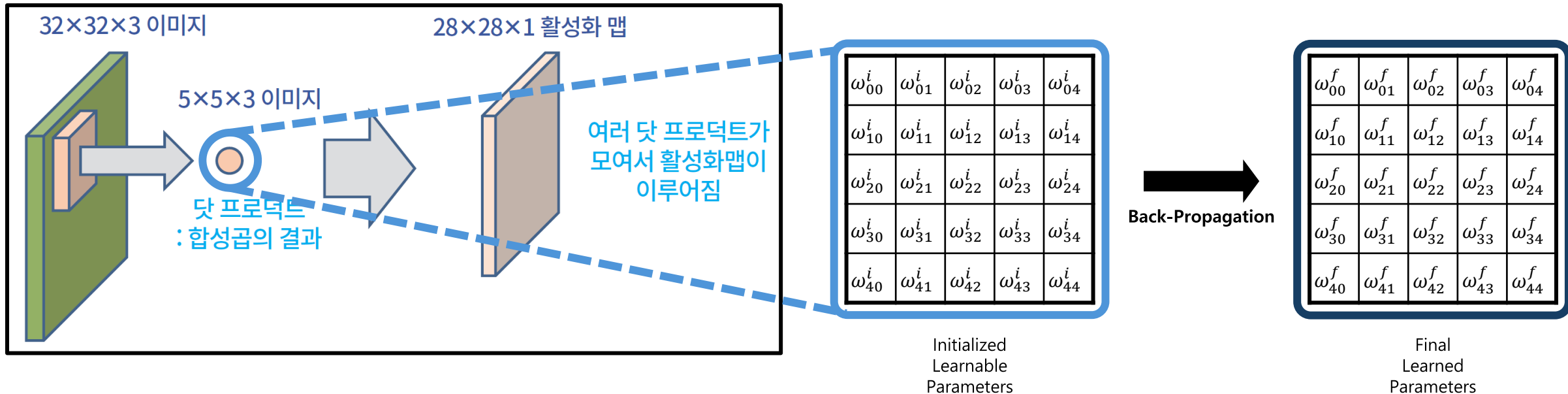
0	-1	0
-1	5	-1
0	-1	0

Sharpening Mask
(n=5)

High-Level Vision (이미지 분류) [10]

• Convolutional Neural Network (Learning-Based)

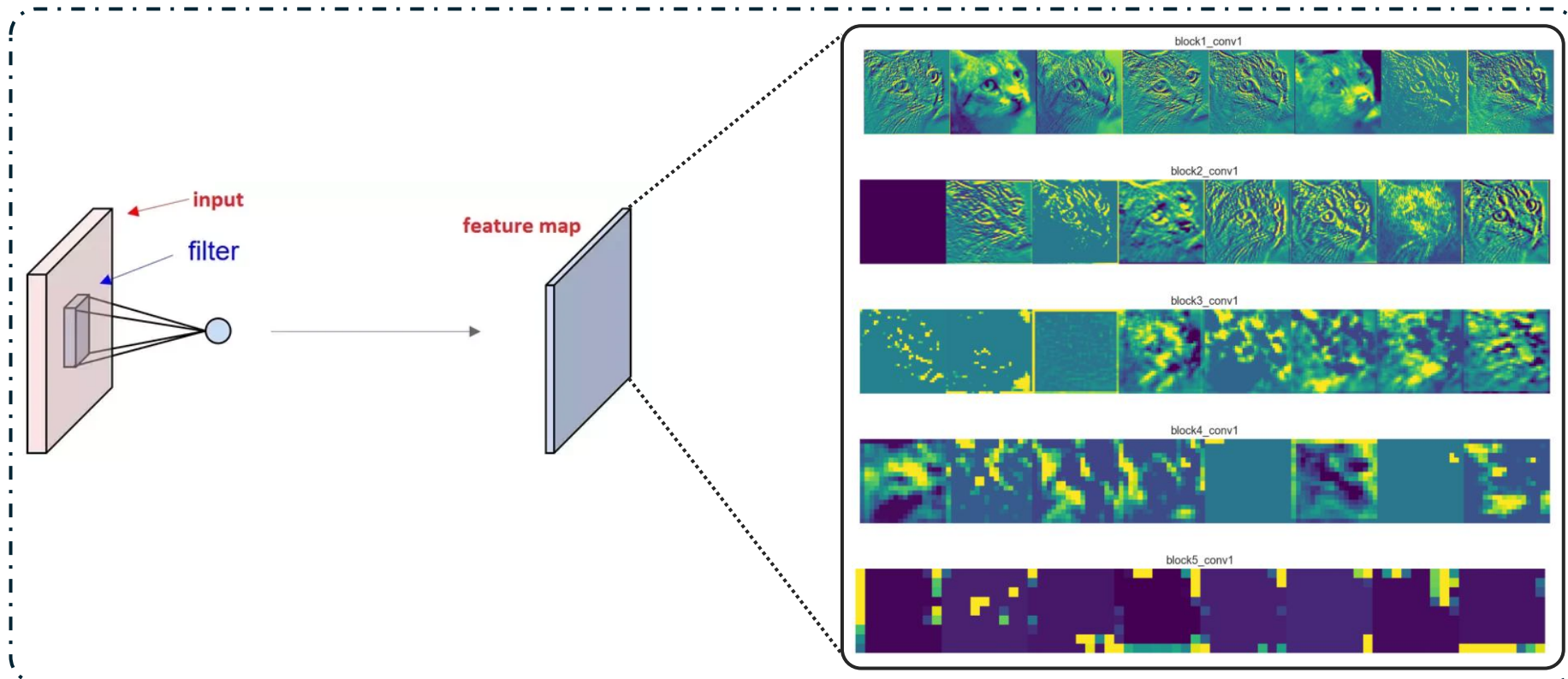
- Convolutional Neural Network (CNN)의 특징
 - 기존 이미지 필터와 다르게 커널 (Kernel)이 훈련 가능 함 (Trainable)
 - 초기화를 통해 커널 내 가중치 (Weight)의 초기값 설정
 - Back-Propagation을 통해 커널 내 가중치 (Weight) Update



High-Level Vision (이미지 분류) [11]

- Convolutional Neural Network (Learning-Based)

- Feature Map (특징 맵)
 - ▶ 입력 이미지와 필터(커널이라고도 함) 간의 합성곱 연산의 출력
 - ▶ 이미지에서 특정 기능이나 패턴의 존재를 나타내는 결과 값의 집합, 매트릭스

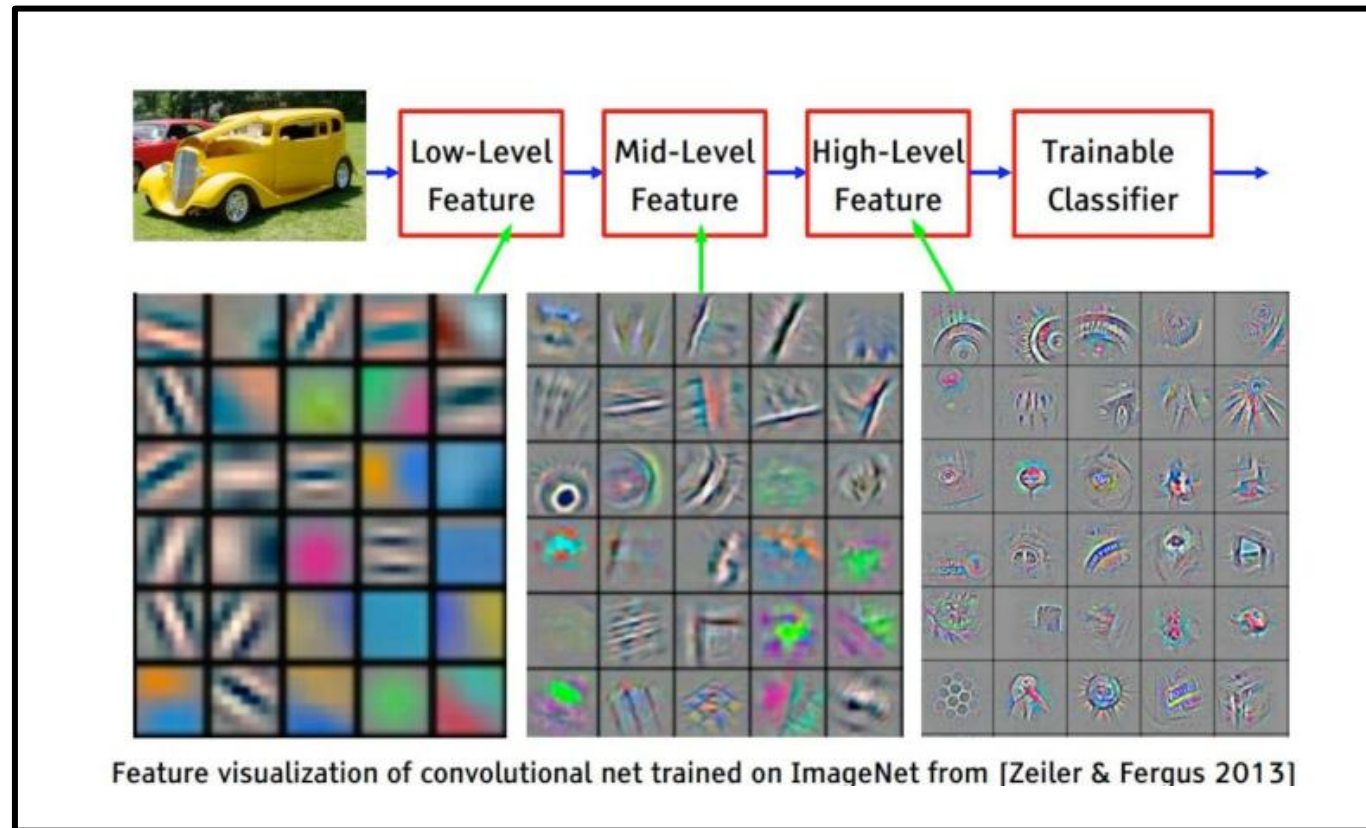


High-Level Vision (이미지 분류) [12]

- Convolutional Neural Network (Learning-Based)

- Feature Map Visualization

▶ CNN 모델의 Feature Map 시각화 → CNN 모델이 어떻게 이미지 분류 문제를 푸는가에 대한 분석

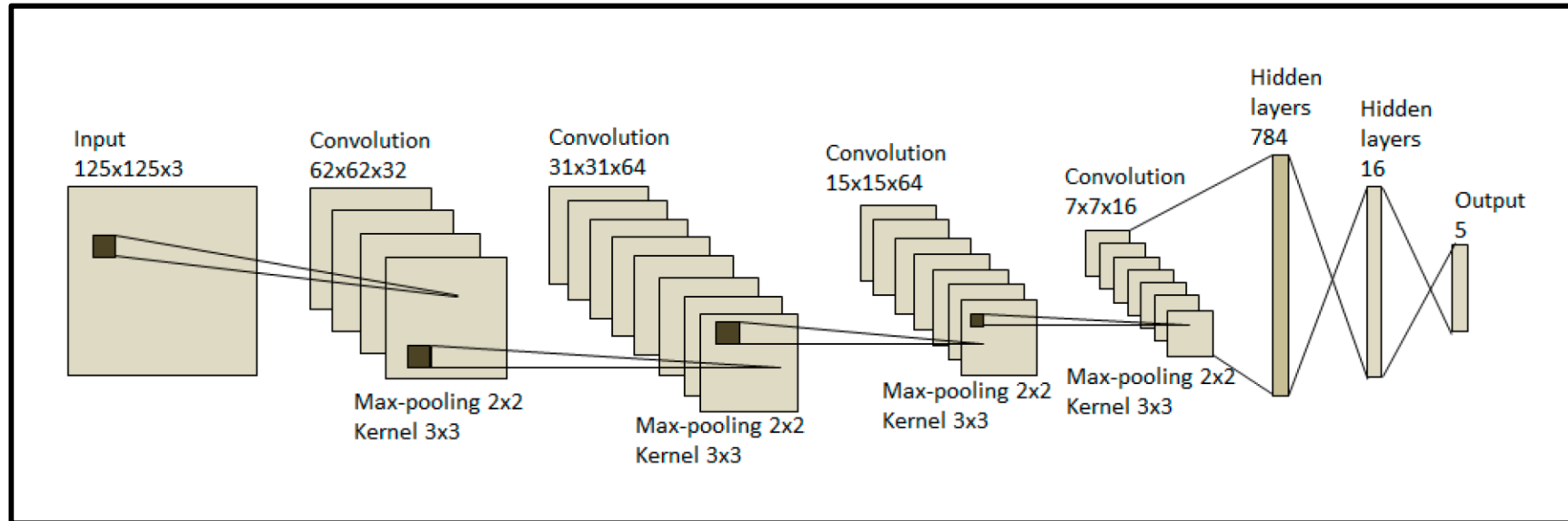


High-Level Vision (이미지 분류) [13]

- Convolutional Neural Network (Learning-Based)

- 이미지 분류 모델의 보편적 구조

- ▶ Convolutional Layer + Pooling + Flatten + MLP의 조합으로 구성

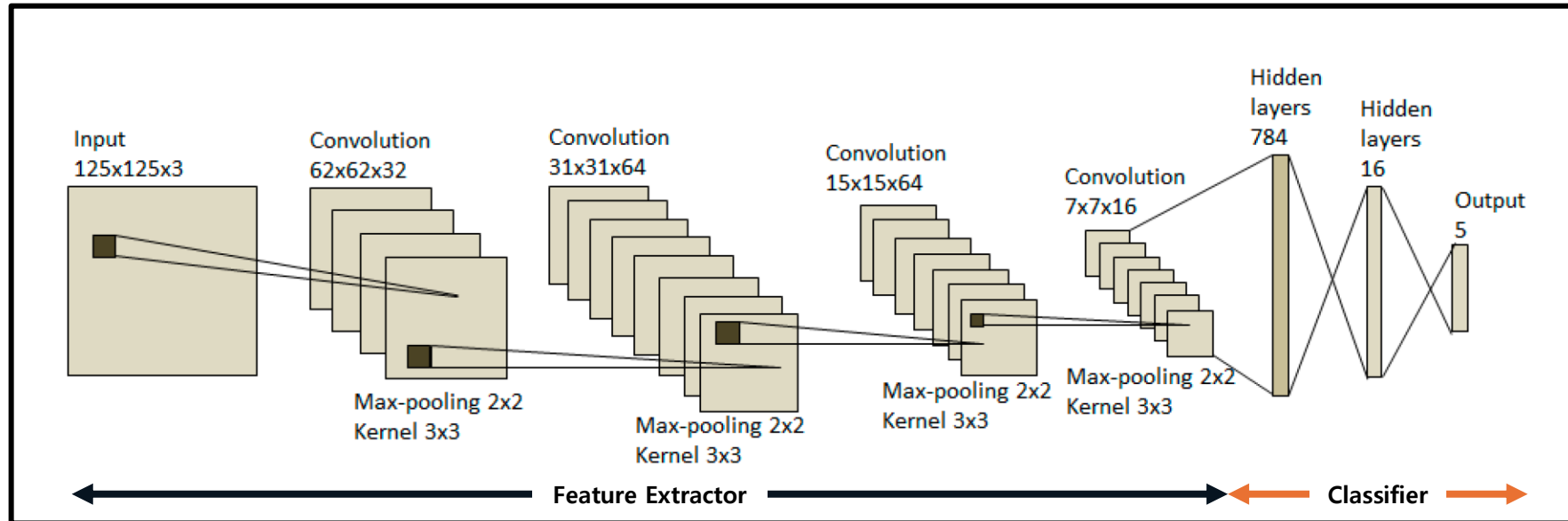


High-Level Vision (이미지 분류) [14]

• Convolutional Neural Network (Learning-Based)

▪ 이미지 분류 모델의 보편적 구조

▶ 특징 추출기 (Feature Extractor)와 분류기 (Classifier)로 구분 가능

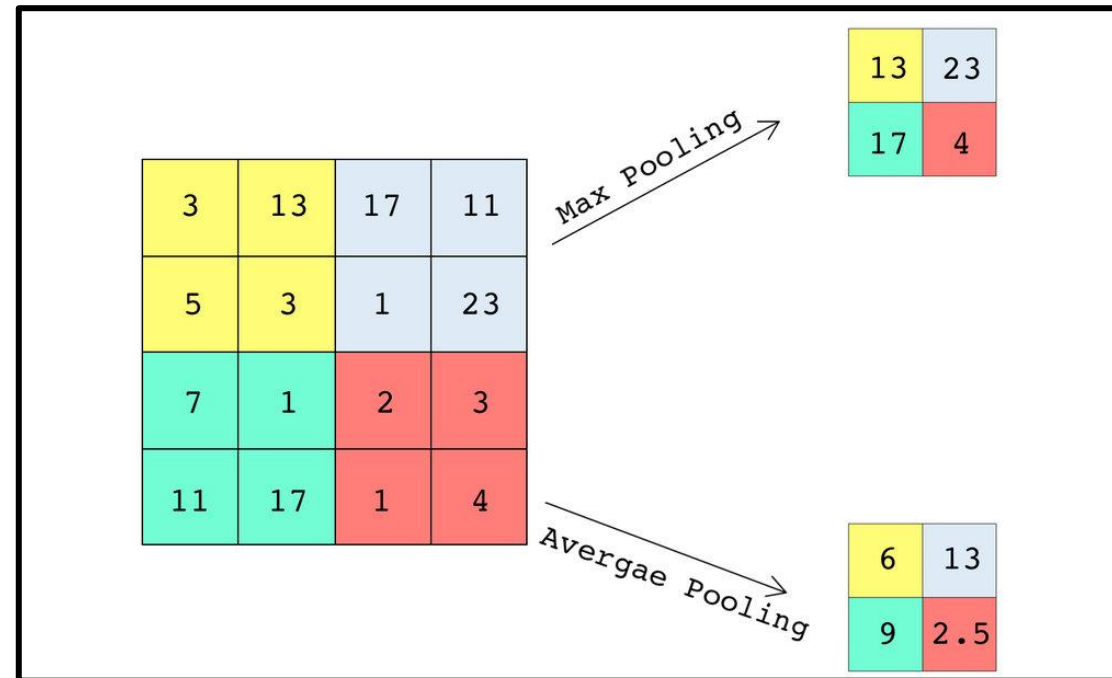


High-Level Vision (이미지 분류) [15]

- Convolutional Neural Network (Learning-Based)

- Pooling Layer

- ▶ 차원 축소를 위한 연산 (Feature Map의 크기를 조절)
 - ▶ 상대적으로 더 중요한 정보를 모으는 과정

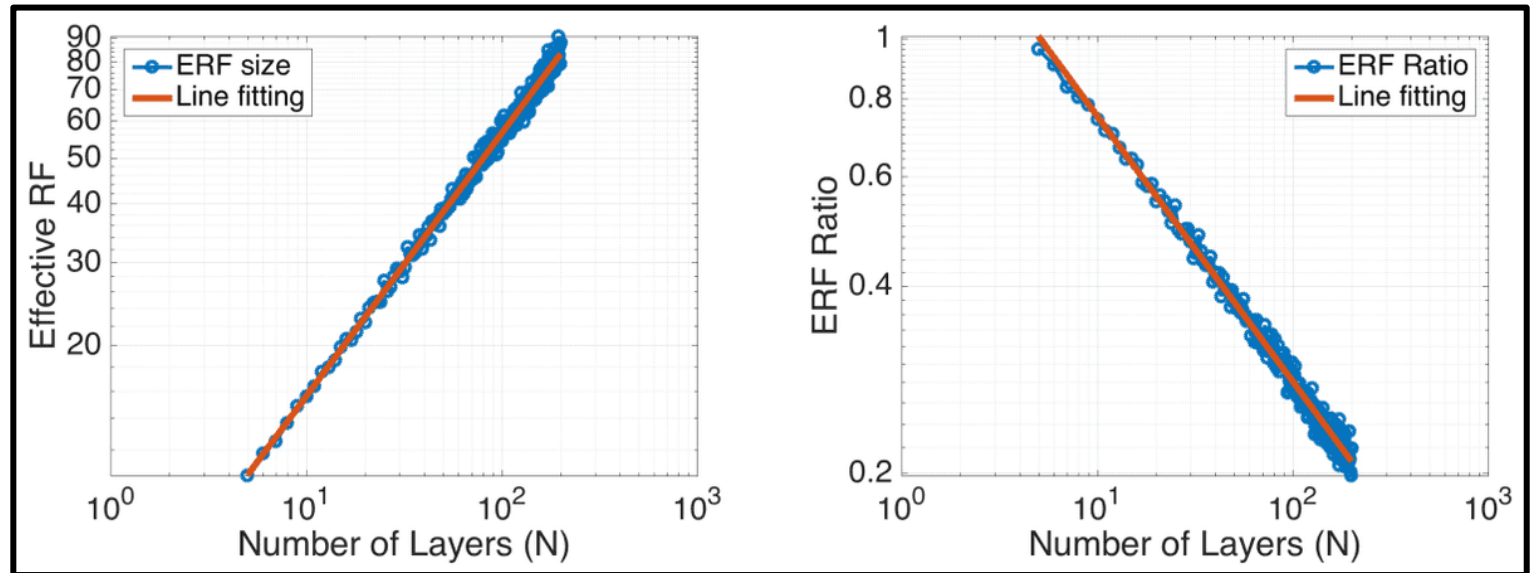
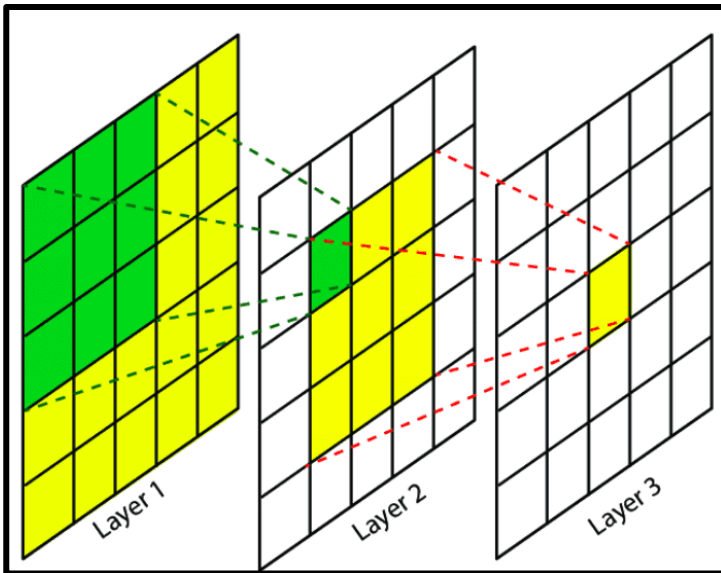


High-Level Vision (이미지 분류) [16]

• Convolutional Neural Network (Learning-Based)

▪ Receptive Field

- ▶ 제한된 Kernel Size (e.g., 3x3) 에 의해 전체 이미지를 한 번에 보는 것은 불가능
- ▶ 이러한 문제를 해결하고자 Convolution 및 Pooling 레이어를 연속적으로 사용
- ▶ Receptive Field가 확보된 모델은 이미지 전체의 정보를 누락 없이 학습 가능

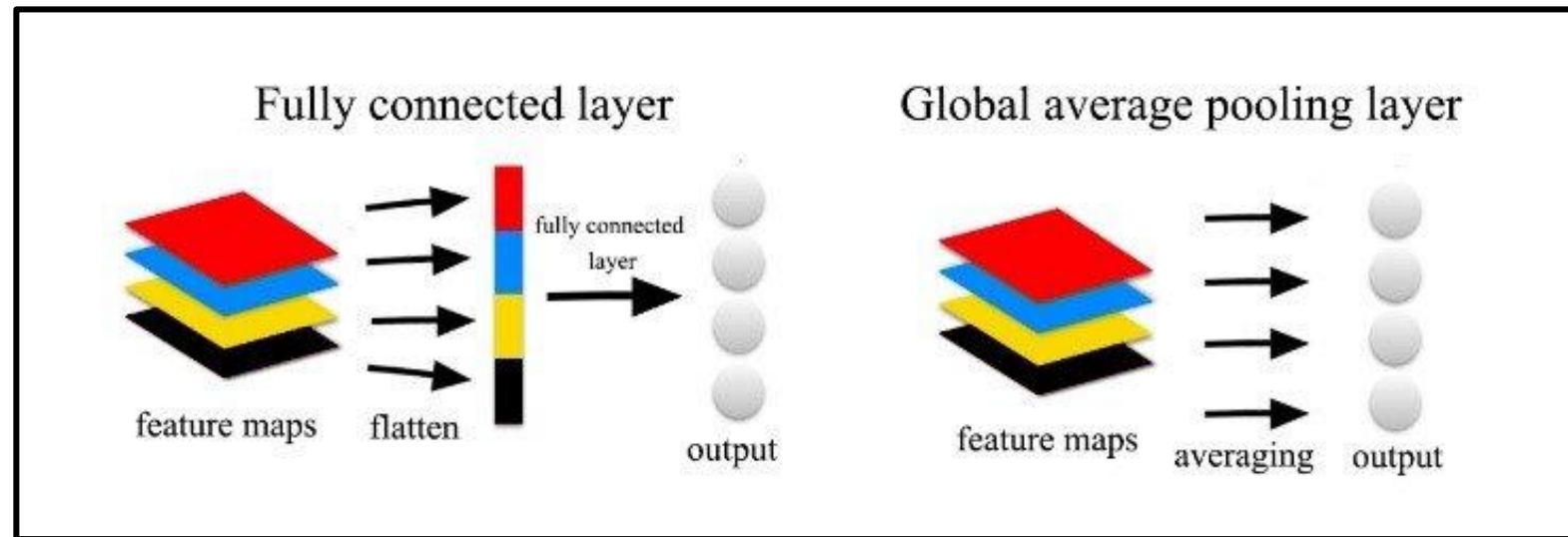


High-Level Vision (이미지 분류) [17]

- Convolutional Neural Network (Learning-Based)

- Convolutional Layer → MLP Layer

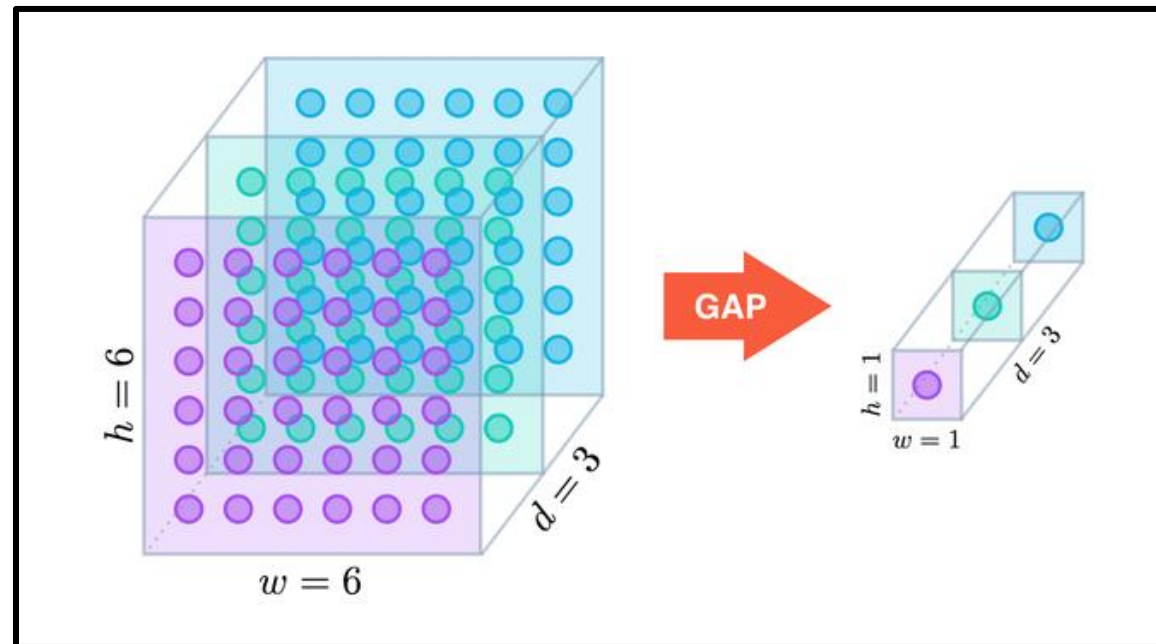
- ▶ 분류를 위해 Convolution 레이어에서 얻은 특징 맵을 MLP 레이어에서 사용
 - ▶ 하지만 두 레이어 간 사용하는 입력 데이터의 차원은 불일치 (3D → 1D)
 - ▶ 입력 데이터를 1차원으로 변환하기 위해 **Flattening** 또는 **Global Average Pooling** (GAP)를 사용



High-Level Vision (이미지 분류) [18]

- Convolutional Neural Network (Learning-Based)

- Why Use Global Average Pooling (GAP)?
 - ▶ Flattening와 달리 Convolution 연산과 흡사
 - ▶ Flattening 대비 출력 노드 개수 감소 ($H \times W \times C$ vs. C)
 - ▶ 입력 이미지의 Translation에 더 견고 (Robustness)



High-Level Vision (이미지 분류) [19]

- **Convolutional Neural Network (Learning-Based)**
 - 이미지 분류 과정 시각화

<https://poloclub.github.io/cnn-explainer/>

High-Level Vision (이미지 분류) [20]

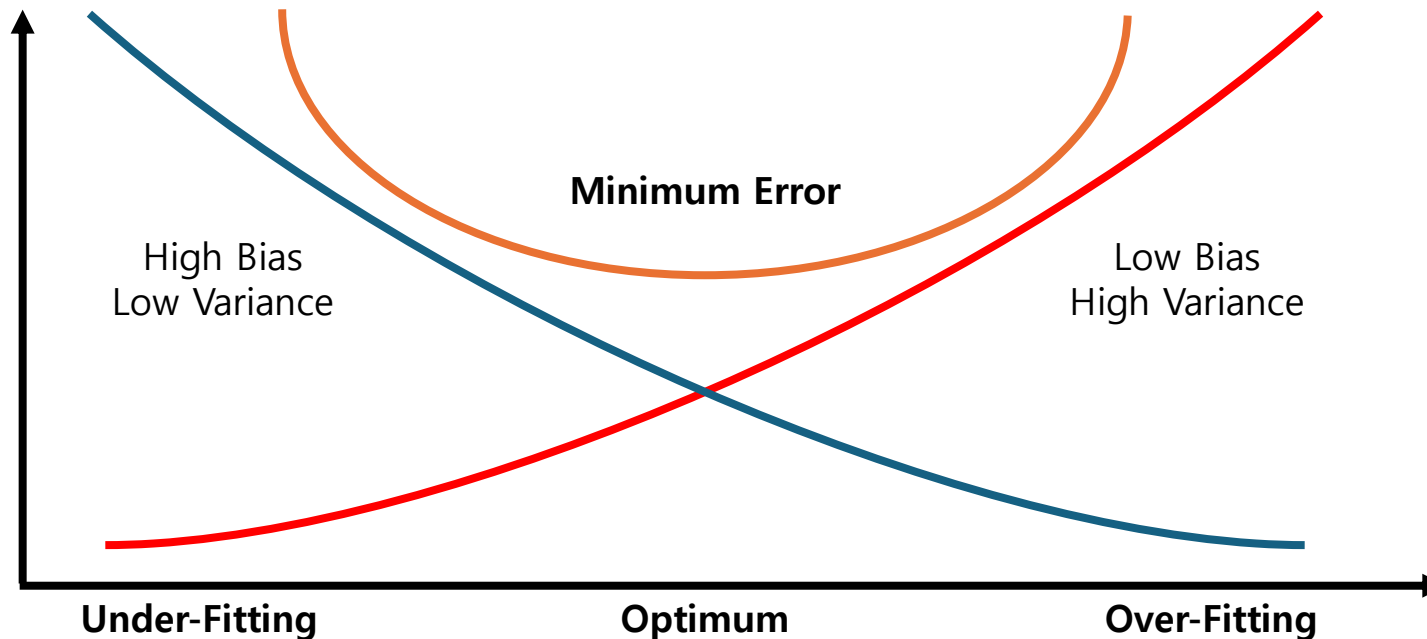
• Convolutional Neural Network (Learning-Based)

▪ Bias-Variance Tradeoff

▶ **Model의 크기 (Capacity)**는 **Bias**와 **Variance**에 의해 결정

▶ **Bayes Estimator의 Mean Squared Error (MSE)**는 다음 수식을 통해 표현 가능

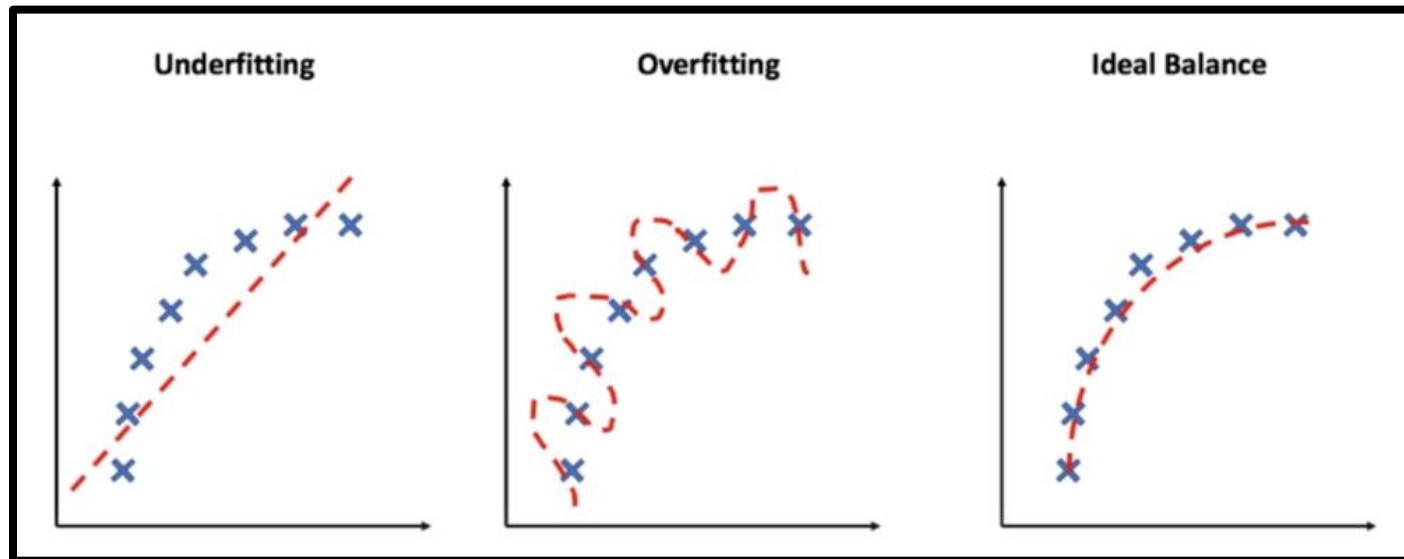
$$MSE = bias^2 + variance$$



High-Level Vision (이미지 분류) [21]

• Convolutional Neural Network (Learning-Based)

- 과적합 (Overfitting)
 - ▶ 인공 신경망이 훈련 데이터셋에 과하게 훈련 된 상태
 - ▶ 훈련 데이터셋에 포함되어 있지 않은 새로운 데이터를 입력 받을 시 성능 감소 (일반성 감소)
 - ▶ 다른 말로 훈련 데이터셋의 분포를 따르지 않는 데이터를 분석하는 능력 감소
 - ▶ 정칙화 (Regularization), Dropout, Batch Normalization 등을 사용하여 해결 가능

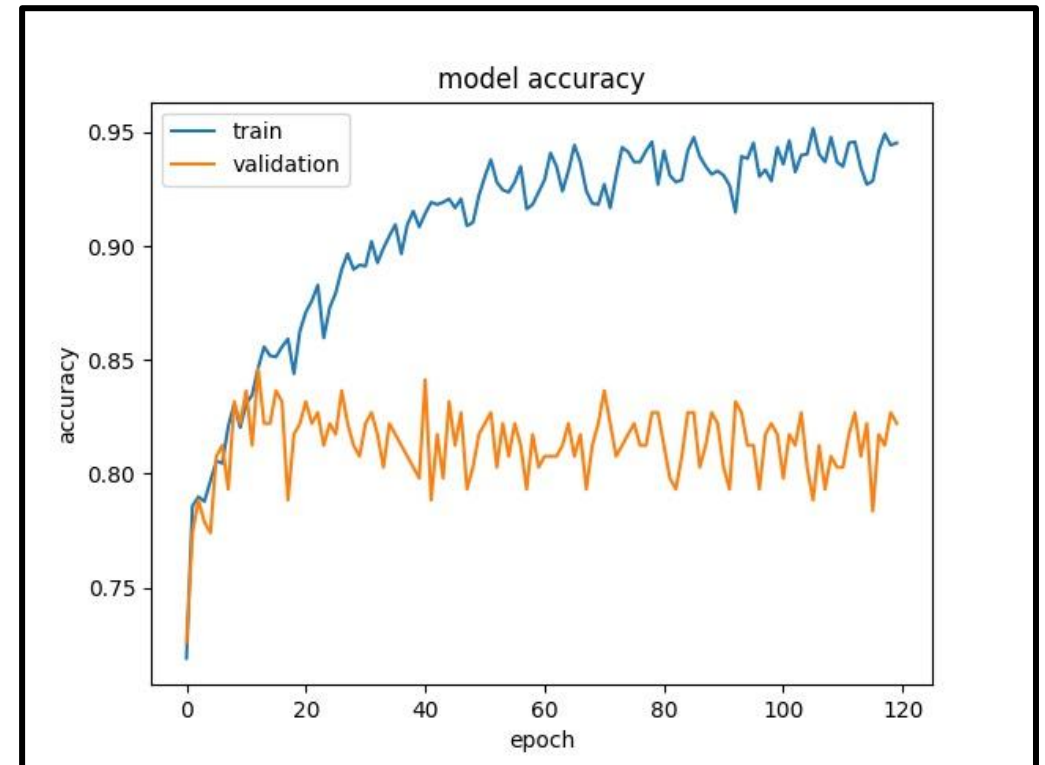
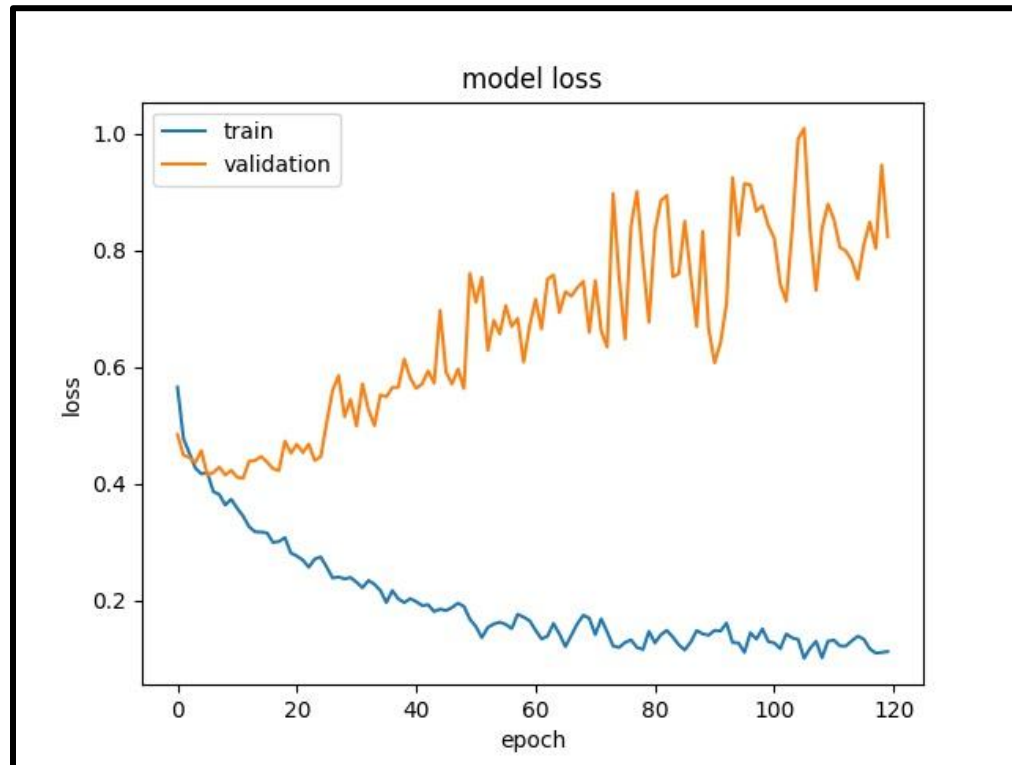


High-Level Vision (이미지 분류) [22]

- Convolutional Neural Network (Learning-Based)

- 과적합 (Overfitting)

▶ 훈련 데이터셋 vs. 검증 데이터셋 (손실 함수 및 지표 비교)



High-Level Vision (이미지 분류) [23]

- Convolutional Neural Network (Learning-Based)

- 정칙화 (Regularization)

- ▶ 특정 가중치 값이 다른 가중치 값 대비 매우 큰 값을 갖지 않도록 **페널티 부여**
 - ▶ 딥 러닝에는 손실 함수에 **L2 Regularization (Norm)**을 부여하여 훈련 진행 (L1 Regularization도 존재)
 - ▶ **L2 Regularization** → 가중치를 제곱하여 모두 더한 값에 **중요도 λ** 를 곱한 값을 사용

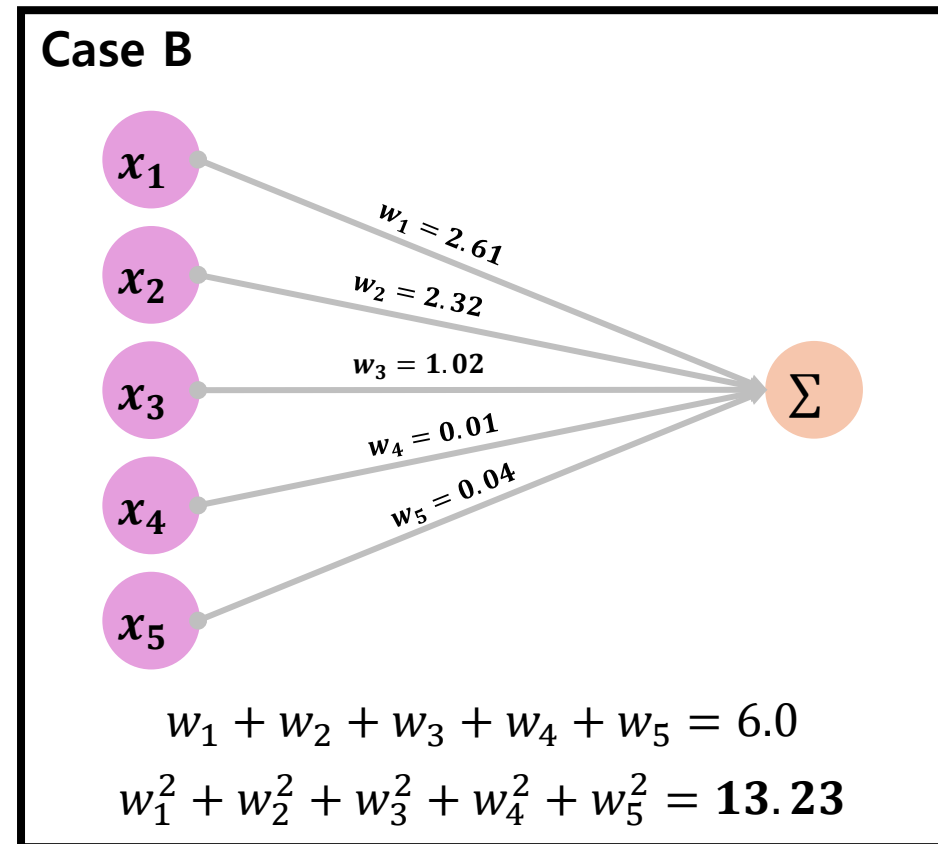
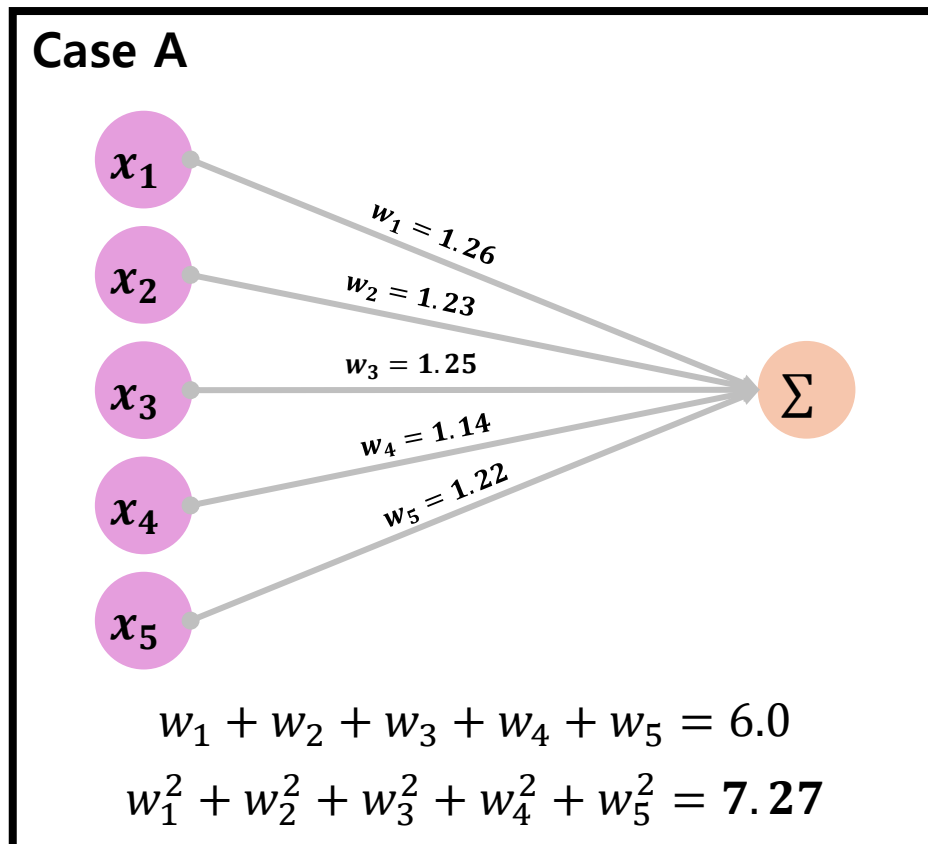
$$E(W) = \frac{1}{2n} \sum_{k=1}^n (y - \hat{y}_k)^2 + \frac{\lambda}{2} \sum w_i^2$$

- ▶ 해당 함수를 사용하여 **손실 함수와 L2-Norm**을 만족하는 **W^*** 를 계산
 - ▶ 개별의 w_i 가 작을수록 Decision Boundary가 부드러운 곡선을 이루어 과적합을 해결
 - ▶ **중요도 λ** 을 사용하여 손실 함수 감소의 비율과 인공 신경망을 단순화의 비율을 결정

High-Level Vision (이미지 분류) [24]

- Convolutional Neural Network (Learning-Based)

- 정칙화 (Regularization)



High-Level Vision (이미지 분류) [25]

- Convolutional Neural Network (Learning-Based)
 - **Dropout:** A Simple Way to Prevent Neural Networks from Overfitting (JMLR 2014)

Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Nitish Srivastava
Geoffrey Hinton
Alex Krizhevsky
Ilya Sutskever
Ruslan Salakhutdinov
*Department of Computer Science
University of Toronto
10 Kings College Road, Rm 3302
Toronto, Ontario, M5S 3G4, Canada.*

NITISH@CS.TORONTO.EDU
HINTON@CS.TORONTO.EDU
KRIZ@CS.TORONTO.EDU
ILYA@CS.TORONTO.EDU
RSALAKHU@CS.TORONTO.EDU

Editor: Yoshua Bengio

Abstract

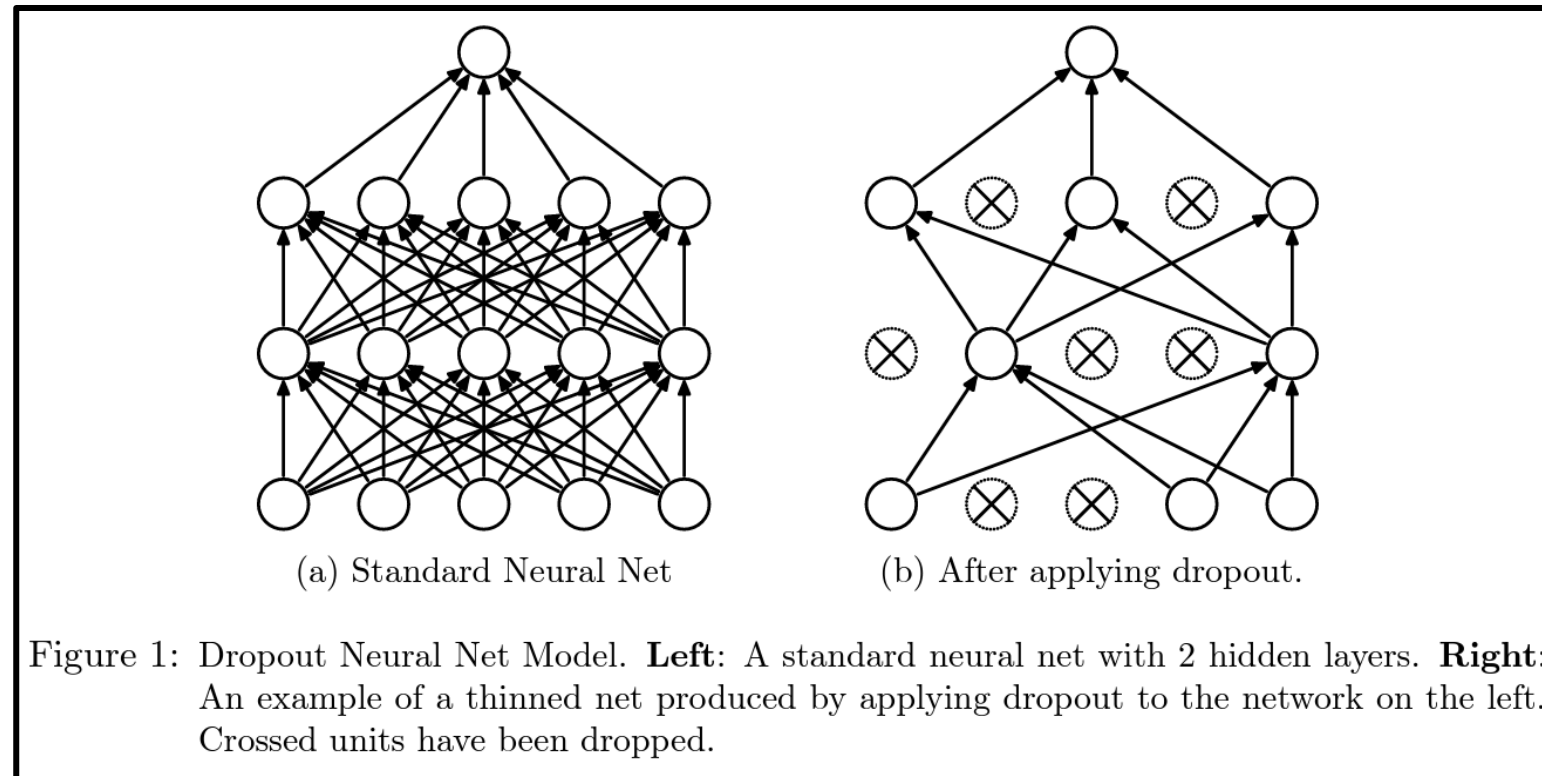
Deep neural nets with a large number of parameters are very powerful machine learning systems. However, overfitting is a serious problem in such networks. Large networks are also slow to use, making it difficult to deal with overfitting by combining the predictions of many different large neural nets at test time. Dropout is a technique for addressing this problem. The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. During training, dropout samples from an exponential number of different “thinned” networks. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights. This significantly reduces overfitting and gives major improvements over other regularization methods. We show that dropout improves the performance of neural networks on supervised learning tasks in vision, speech recognition, document classification and computational biology, obtaining state-of-the-art results on many benchmark data sets.

Keywords: neural networks, regularization, model combination, deep learning

High-Level Vision (이미지 분류) [26]

- Convolutional Neural Network (Learning-Based)

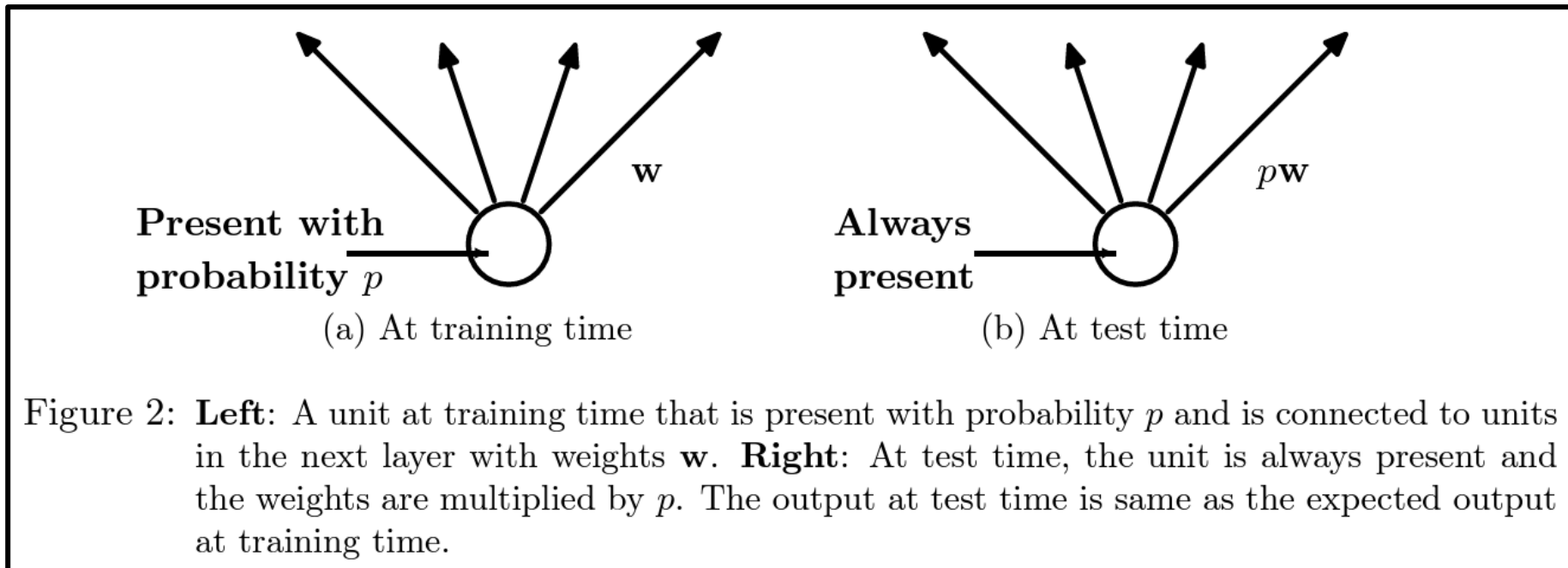
- **Dropout:** A Simple Way to Prevent Neural Networks from Overfitting (JMLR 2014)



High-Level Vision (이미지 분류) [27]

- Convolutional Neural Network (Learning-Based)

- **Dropout:** A Simple Way to Prevent Neural Networks from Overfitting (JMLR 2014)



High-Level Vision (이미지 분류) [28]

- Convolutional Neural Network (Learning-Based)

- **Dropout:** A Simple Way to Prevent Neural Networks from Overfitting (JMLR 2014)

4. Model Description

This section describes the dropout neural network model. Consider a neural network with L hidden layers. Let $l \in \{1, \dots, L\}$ index the hidden layers of the network. Let $\mathbf{z}^{(l)}$ denote the vector of inputs into layer l , $\mathbf{y}^{(l)}$ denote the vector of outputs from layer l ($\mathbf{y}^{(0)} = \mathbf{x}$ is the input). $W^{(l)}$ and $\mathbf{b}^{(l)}$ are the weights and biases at layer l . The feed-forward operation of a standard neural network (Figure 3a) can be described as (for $l \in \{0, \dots, L-1\}$ and any hidden unit i)

$$\begin{aligned} z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \mathbf{y}^l + b_i^{(l+1)}, \\ y_i^{(l+1)} &= f(z_i^{(l+1)}), \end{aligned}$$

where f is any activation function, for example, $f(x) = 1 / (1 + \exp(-x))$.

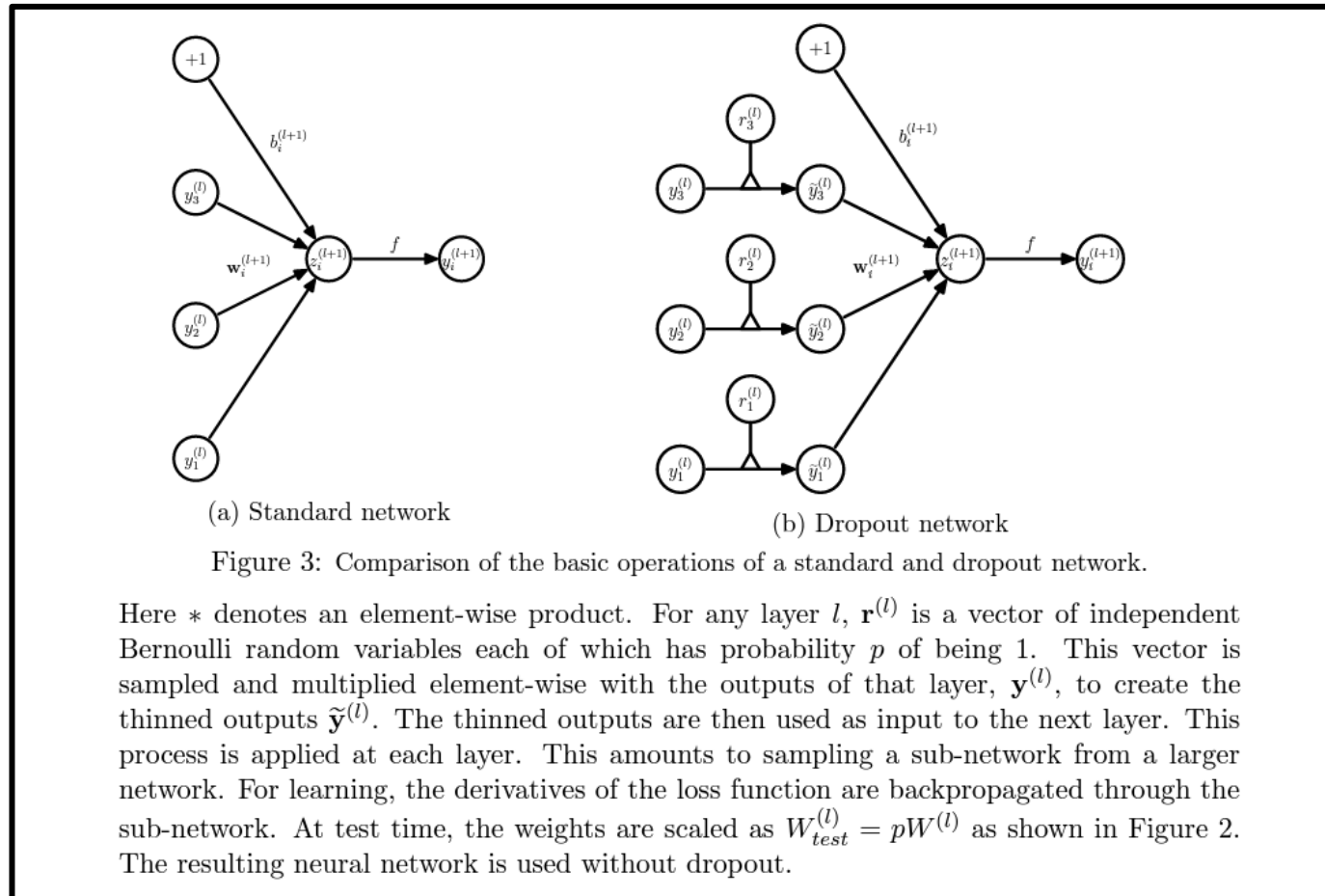
With dropout, the feed-forward operation becomes (Figure 3b)

$$\begin{aligned} r_j^{(l)} &\sim \text{Bernoulli}(p), \\ \tilde{\mathbf{y}}^{(l)} &= \mathbf{r}^{(l)} * \mathbf{y}^{(l)}, \\ z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \tilde{\mathbf{y}}^l + b_i^{(l+1)}, \\ y_i^{(l+1)} &= f(z_i^{(l+1)}). \end{aligned}$$

High-Level Vision (이미지 분류) [29]

• Convolutional Neural Network (Learning-Based)

- **Dropout:** A Simple Way to Prevent Neural Networks from Overfitting (JMLR 2014)



High-Level Vision (이미지 분류) [30]

• Convolutional Neural Network (Learning-Based)

- **Dropout:** A Simple Way to Prevent Neural Networks from Overfitting (JMLR 2014)

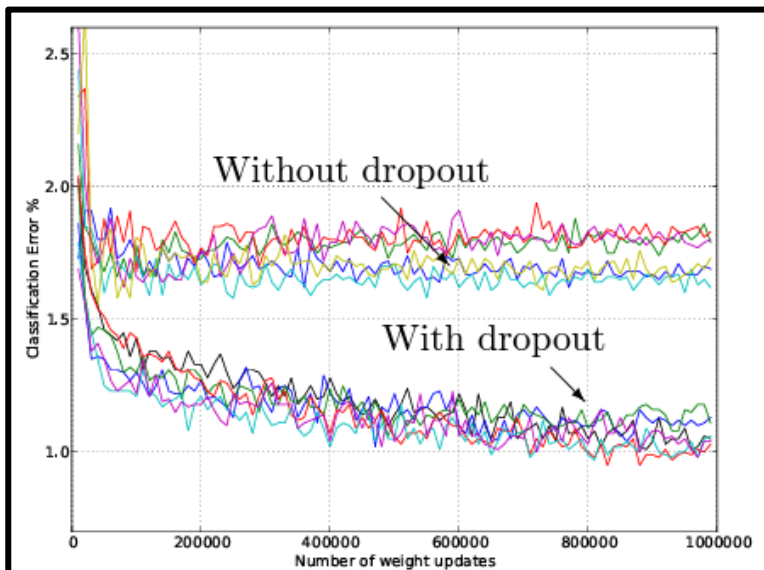


Figure 4: Test error for different architectures with and without dropout. The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

Method	CIFAR-10	CIFAR-100
Conv Net + max pooling (hand tuned)	15.60	43.48
Conv Net + stochastic pooling (Zeiler and Fergus, 2013)	15.13	42.51
Conv Net + max pooling (Snoek et al., 2012)	14.98	-
Conv Net + max pooling + dropout fully connected layers	14.32	41.26
Conv Net + max pooling + dropout in all layers	12.61	37.20
Conv Net + maxout (Goodfellow et al., 2013)	11.68	38.57

Table 4: Error rates on CIFAR-10 and CIFAR-100.

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SVM on Fisher Vectors of Dense SIFT and Color Statistics	-	-	27.3
Avg of classifiers over FVs of SIFT, LBP, GIST and CSIFT	-	-	26.2
Conv Net + dropout (Krizhevsky et al., 2012)	40.7	18.2	-
Avg of 5 Conv Nets + dropout (Krizhevsky et al., 2012)	38.1	16.4	16.4

Table 6: Results on the ILSVRC-2012 validation/test set.

High-Level Vision (이미지 분류) [31]

• Convolutional Neural Network (Learning-Based)

- **Batch Normalization:** Accelerating Deep Network Training by Reducing Internal Covariate Shift (ICML 2015)

Abstract

Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities. We refer to this phenomenon as *internal covariate shift*, and address the problem by normalizing layer inputs. Our method draws its strength from making normalization a part of the model architecture and performing the normalization *for each training mini-batch*. Batch Normalization allows us to use much higher learning rates and be less careful about initialization, and in some cases eliminates the need for Dropout. Applied to a state-of-the-art image classification model, Batch Normalization achieves the same accuracy with 14 times fewer training steps, and beats the original model by a significant margin. Using an ensemble of batch-normalized networks, we improve upon the best published result on ImageNet classification: reaching 4.82% top-5 test error, exceeding the accuracy of human raters.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

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

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

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

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

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

$$\frac{\partial \ell}{\partial \hat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$

$$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^m \frac{\partial \ell}{\partial \hat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2} (\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2}$$

$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = \sum_{i=1}^m \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^m \frac{\partial \ell}{\partial y_i} \cdot \hat{x}_i$$

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^m \frac{\partial \ell}{\partial y_i}$$

3.2. Batch-Normalized Convolutional Networks

Batch Normalization can be applied to any set of activations in the network. Here, we focus on transforms that consist of an affine transformation followed by an element-wise nonlinearity:

$$z = g(Wu + b)$$

where W and b are learned parameters of the model, and $g(\cdot)$ is the nonlinearity such as sigmoid or ReLU. This formulation covers both fully-connected and convolutional layers. We add the BN transform immediately before the nonlinearity, by normalizing $x = Wu + b$. We could have also normalized the layer inputs u , but since u is likely the output of another nonlinearity, the shape of its distribution is likely to change during training, and constraining its first and second moments would not eliminate the covariate shift. In contrast, $Wu + b$ is more likely to have a symmetric, non-sparse distribution, that is “more Gaussian” (Hyvärinen & Oja, 2000); normalizing it is likely to produce activations with a stable distribution.

Note that, since we normalize $Wu + b$, the bias b can be ignored since its effect will be canceled by the subsequent mean subtraction (the role of the bias is subsumed by β in Alg. 1). Thus, $z = g(Wu + b)$ is replaced with

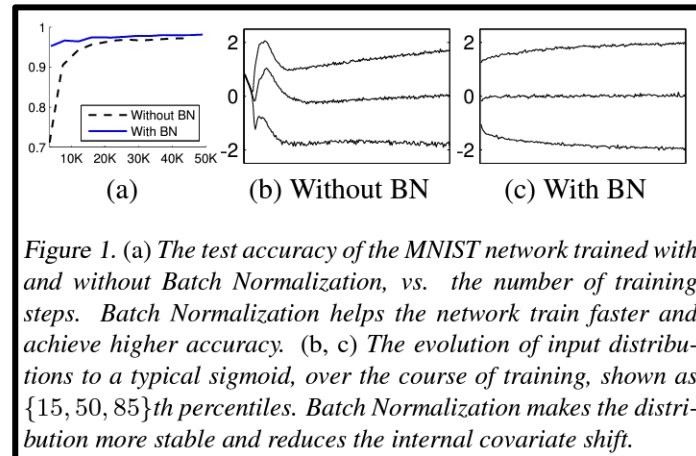
$$z = g(\text{BN}(Wu))$$

where the BN transform is applied independently to each dimension of $x = Wu$, with a separate pair of learned parameters $\gamma^{(k)}, \beta^{(k)}$ per dimension.

High-Level Vision (이미지 분류) [32]

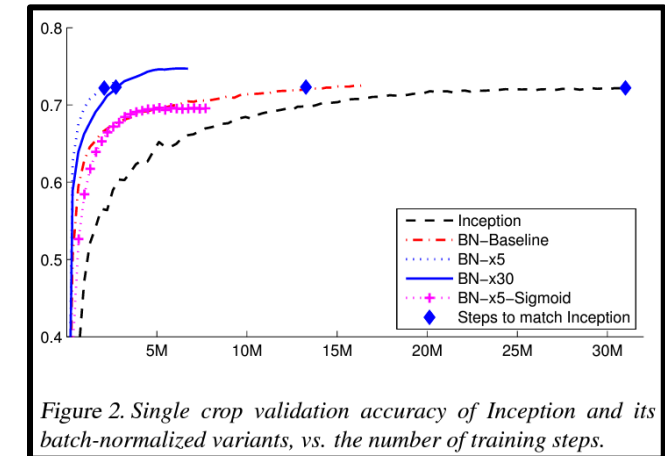
• Convolutional Neural Network (Learning-Based)

- **Batch Normalization:** Accelerating Deep Network Training by Reducing Internal Covariate Shift (ICML 2015)



Model	Steps to 72.2%	Max accuracy
Inception	$31.0 \cdot 10^6$	72.2%
BN-Baseline	$13.3 \cdot 10^6$	72.7%
BN-x5	$2.1 \cdot 10^6$	73.0%
BN-x30	$2.7 \cdot 10^6$	74.8%
BN-x5-Sigmoid		69.8%

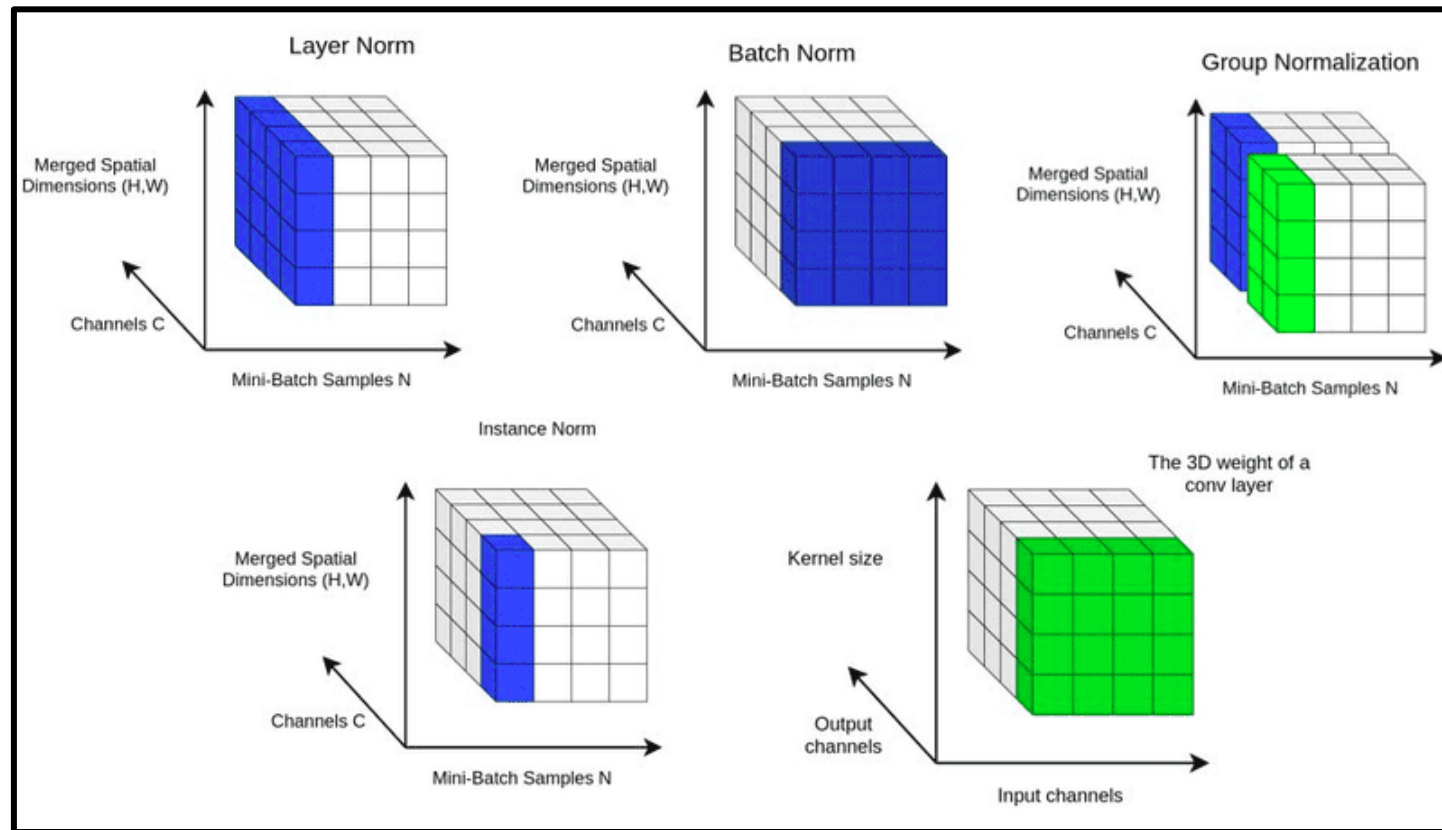
Figure 3. For Inception and the batch-normalized variants, the number of training steps required to reach the maximum accuracy of Inception (72.2%), and the maximum accuracy achieved by the network.



Model	Resolution	Crops	Models	Top-1 error	Top-5 error
GoogLeNet ensemble	224	144	7	-	6.67%
Deep Image low-res	256	-	1	-	7.96%
Deep Image high-res	512	-	1	24.88	7.42%
Deep Image ensemble	up to 512	-	-	-	5.98%
MSRA multicrop	up to 480	-	-	-	5.71%
MSRA ensemble	up to 480	-	-	-	4.94%*
BN-Inception single crop	224	1	1	25.2%	7.82%
BN-Inception multicrop	224	144	1	21.99%	5.82%
BN-Inception ensemble	224	144	6	20.1%	4.82%*

High-Level Vision (이미지 분류) [33]




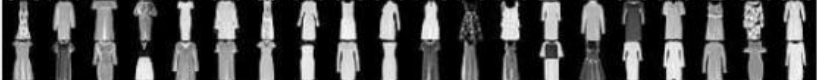






- Convolutional Neural Network (Learning-Based)
 - Types of Normalization



High-Level Vision (이미지 분류) [34]

- Convolutional Neural Network (Learning-Based)

- Fashion MNIST Dataset 분류 (실습) [1]

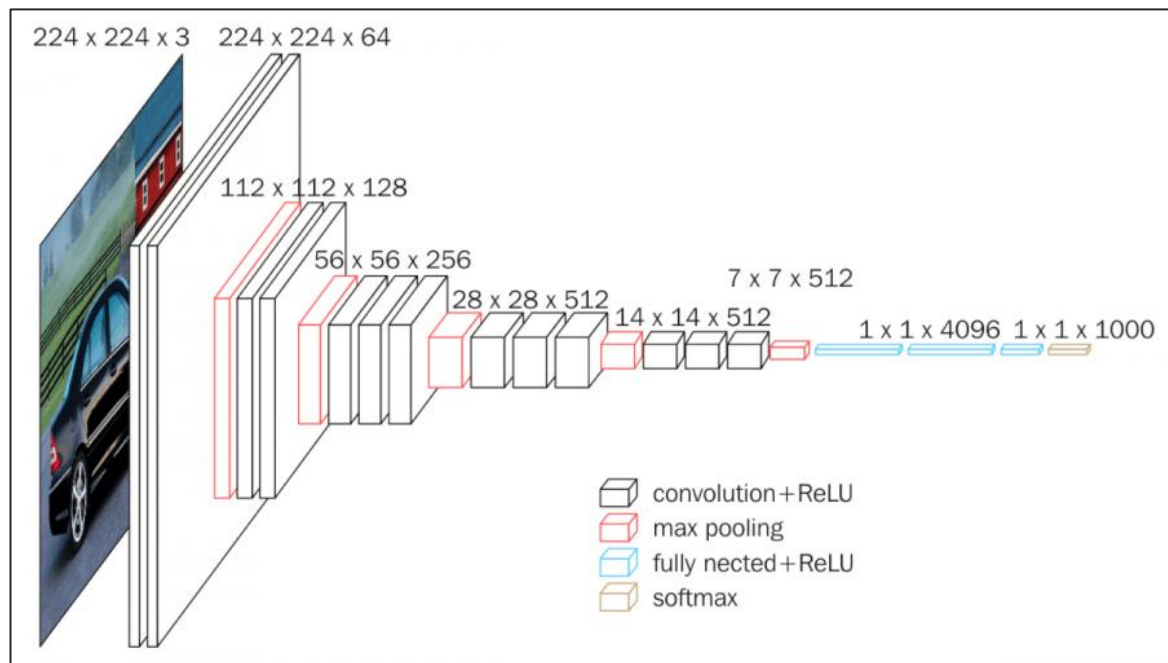
Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

High-Level Vision (이미지 분류) [35]

• Convolutional Neural Network (Learning-Based)

- Fashion MNIST Dataset 분류 (실습) [2]

▶ VGG Network



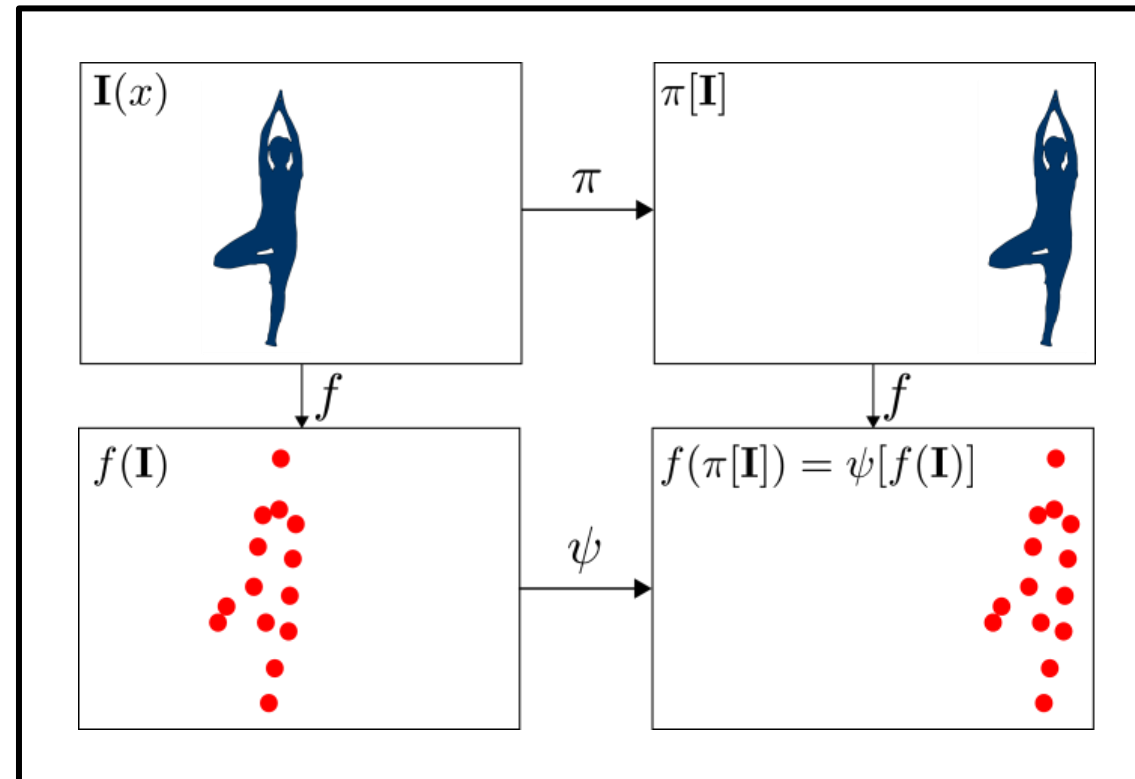
ConvNet 구성					
A	A-LRN	B	C	D	E
11 layers	11 layers	13 layers	16 layers	16 layers	19 layers
input (224 x 224 RGB)					
conv3-64	conv3-64	conv3-64	conv3-64	conv3-64	conv3-64
	LRN	conv3-65	conv3-65	conv3-65	conv3-65
maxpool					
conv3-128	conv3-128	conv3-128	conv3-128	conv3-128	conv3-128
		conv3-128	conv3-128	conv3-128	conv3-128
maxpool					
conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256
conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256
			conv1-256	conv3-256	conv3-256
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	conv3-512	conv3-512
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	conv3-512	conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
softmax					

High-Level Vision (이미지 분류) [36]

- Convolutional Neural Network (Learning-Based)

- Shift Invariant

- ▶ Convolutional Operation의 특징

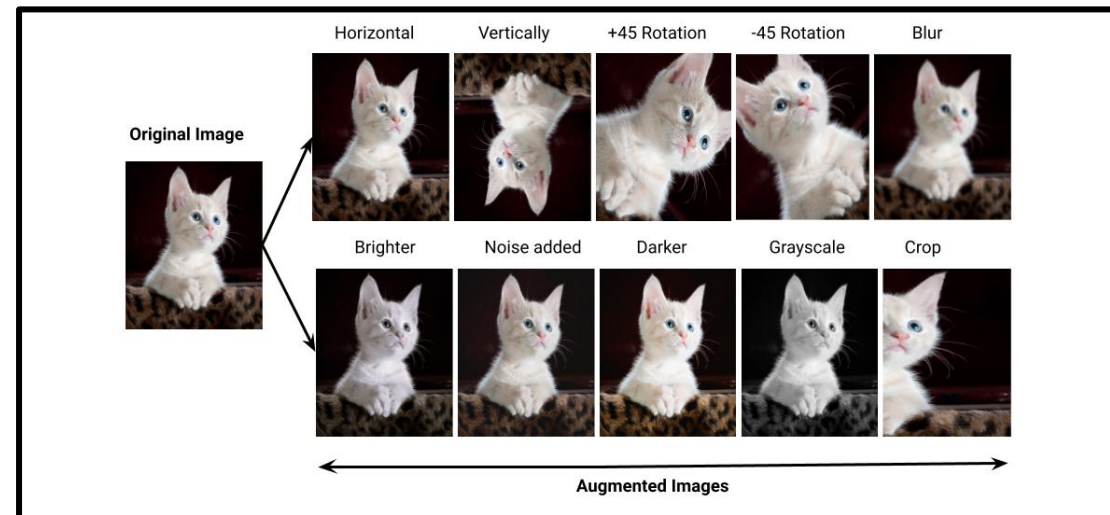


High-Level Vision (이미지 분류) [37]

• Convolutional Neural Network (Learning-Based)

▪ Data Augmentation

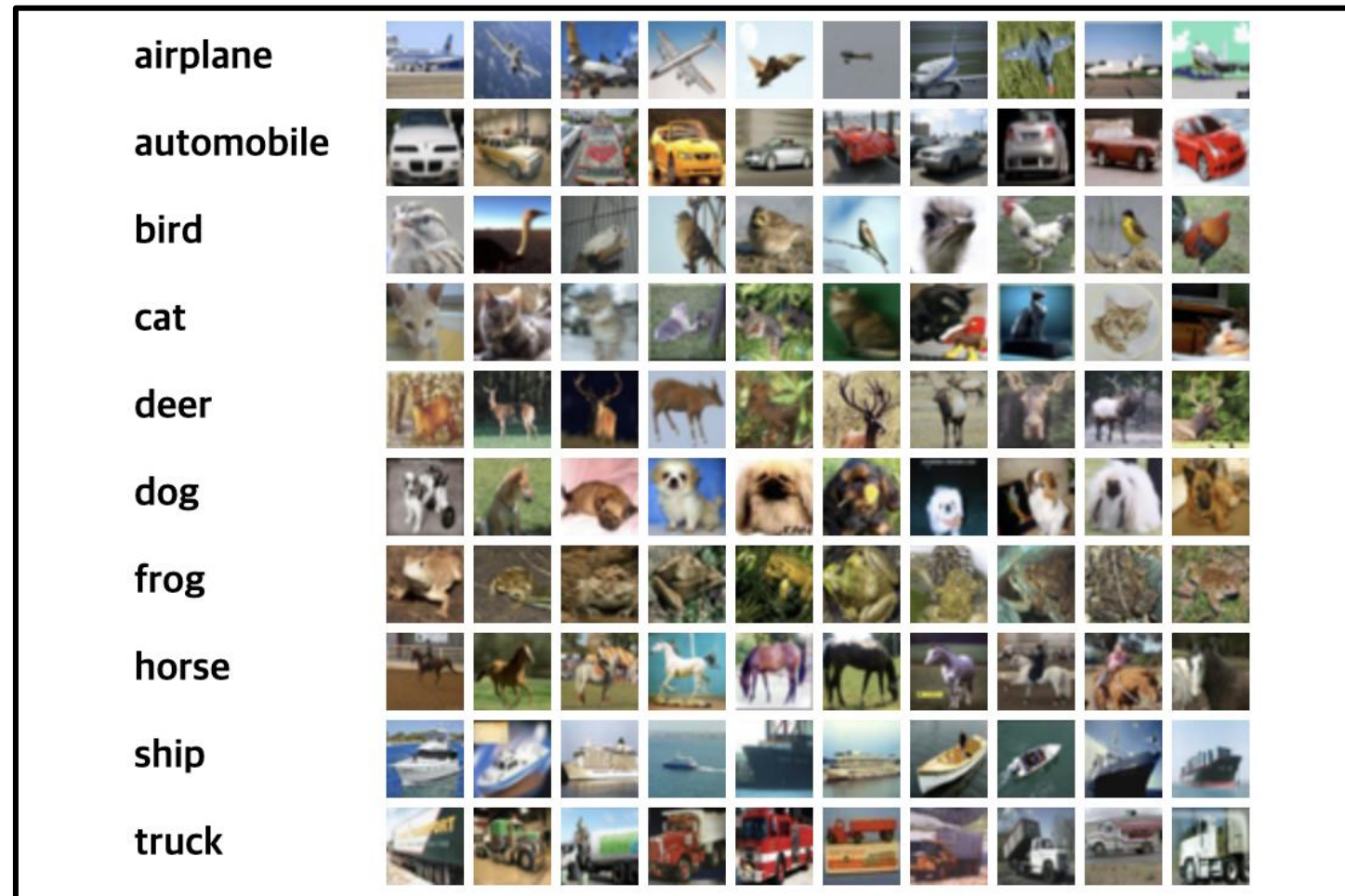
- ▶ 딥 러닝 모델의 과적합 (Overfitting) 문제를 해결하기 위한 또 하나의 방법
- ▶ 데이터 Pair의 경우의 수를 늘려 입력 데이터의 범위를 늘리는 과정
- ▶ CNN의 경우 Translational Invariance 성질을 이용하여 입력 이미지를 다양한 방식으로 확장
 - ▷ Flipping
 - ▷ Color Space
 - ▷ Cropping
 - ▷ Rotation
 - ▷ Translation
 - ▷ Noise Injection
 - ▷ Blurring



High-Level Vision (이미지 분류) [38]

- Convolutional Neural Network (Learning-Based)

- CIFAR-10 Dataset 분류 (실습) [1]



High-Level Vision (이미지 분류) [39]

• Convolutional Neural Network (Learning-Based)

- CIFAR-10 Dataset 분류 (실습) [2]

▶ CNN Training From-Scratch & Data-Augmentation

ConvNet 구성					
A	A-LRN	B	C	D	E
11 layers	11 layers	13 layers	16 layers	16 layers	19 layers
input (224 x 224 RGB)					
conv3-64	conv3-64	conv3-64	conv3-64	conv3-64	conv3-64
	LRN	conv3-65	conv3-65	conv3-65	conv3-65
maxpool					
conv3-128	conv3-128	conv3-128	conv3-128	conv3-128	conv3-128
		conv3-128	conv3-128	conv3-128	conv3-128
maxpool					
conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256
conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256
			conv1-256	conv3-256	conv3-256
					conv3-256
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	conv3-512	conv3-512
					conv3-512
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	conv3-512	conv3-512
					conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
softmax					

High-Level Vision (이미지 분류) [40]

- Convolutional Neural Network (Learning-Based)

- 사전 학습된 이미지 분류 모델

- ▶ ImageNet Dataset

- ▷ ImageNet은 시각적 객체 인식 연구에 사용하도록 설계된 주석이 달린 이미지의 대규모 데이터베이스
 - ▷ 1,400만 개 이상의 이미지가 포함되어 있음
 - ▷ 각 이미지에는 WordNet 동기화 세트를 사용하여 주석을 달아 컴퓨터 비전 작업에서 딥러닝 모델을 훈련하는 데 사용할 수 있는 가장 광범위한 리소스



High-Level Vision (이미지 분류) [41]

• Convolutional Neural Network (Learning-Based)

▪ 사전 학습된 이미지 분류 모델

▶ PyTorch Pretrained Models

- | | |
|------------------|---------------------|
| ▷ AlexNet | ▷ RegNet |
| ▷ ConvNeXt | ▷ ResNet |
| ▷ DenseNet | ▷ ResNeXt |
| ▷ EfficientNet | ▷ ShuffleNetV2 |
| ▷ EfficientNetV2 | ▷ SqueezeNet |
| ▷ GoogLeNet | ▷ SwinTransformer |
| ▷ Inception V3 | ▷ VGG |
| ▷ MaxVit | ▷ VisionTransformer |
| ▷ MNASNet | ▷ Wide ResNet |
| ▷ MobileNetV2 | |
| ▷ MobileNetV3 | |

```
from torchvision.io import decode_image
from torchvision.models import resnet50, ResNet50_Weights

img = decode_image("test/assets/encode_jpeg/grace_hopper_517x606.jpg")

# Step 1: Initialize model with the best available weights
weights = ResNet50_Weights.DEFAULT
model = resnet50(weights=weights)
model.eval()

# Step 2: Initialize the inference transforms
preprocess = weights.transforms()

# Step 3: Apply inference preprocessing transforms
batch = preprocess(img).unsqueeze(0)

# Step 4: Use the model and print the predicted category
prediction = model(batch).squeeze(0).softmax(0)
class_id = prediction.argmax().item()
score = prediction[class_id].item()
category_name = weights.meta["categories"][class_id]
print(f"{category_name}: {100 * score:.1f}%")
```

Torchvision ver. 0.20.0

High-Level Vision (이미지 분류) [42]

- **Convolutional Neural Network (Learning-Based)**
 - List of Class Indices

<https://deeplearning.cms.waikato.ac.nz/user-guide/class-maps/IMAGENET/>

High-Level Vision (이미지 분류) [43]

• Convolutional Neural Network (Learning-Based)

▪ 사전 학습된 이미지 분류 모델 (실습)

▶ PyTorch Pretrained Models

▷ **AlexNet**

▷ ConvNeXt

▷ DenseNet

▷ EfficientNet

▷ EfficientNetV2

▷ GoogLeNet

▷ Inception V3

▷ MaxVit

▷ MNASNet

▷ MobileNetV2

▷ MobileNetV3

▷ RegNet

▷ **ResNet**

▷ ResNeXt

▷ ShuffleNetV2

▷ SqueezeNet

▷ SwinTransformer

▷ **VGG**

▷ VisionTransformer

▷ Wide ResNet



High-Level Vision (이미지 분류) [44]

- Convolutional Neural Network (Learning-Based)

- Transfer Learning (전이학습)

딥러닝 모델 훈련을 위한 전이학습

전이학습은 하나의 작업을 위해 훈련된 모델을 유사 작업 수행 모델의 시작점으로 활용하는 딥러닝 접근법입니다. 일반적으로 신경망은 처음부터 새로 훈련하는 것보다 전이학습을 통해 업데이트하고 재훈련하는 편이 더 빠르고 간편합니다. 이 접근법은 여러 응용 분야 중에서도 특히 객체 검출, 영상 인식, 음성 인식 분야에 흔히 사용됩니다.

전이학습은 다음과 같은 이유로 널리 쓰입니다.

- 대용량 데이터셋으로 이미 훈련된 널리 쓰이는 모델을 재사용함으로써 더 적은 레이블 지정 데이터를 사용하여 모델을 훈련할 수 있습니다.
- 훈련 시간과 연산 리소스를 줄일 수 있습니다. 전이학습에서는 사전 훈련된 모델이 이전의 학습을 기반으로 이미 가중치를 학습했으므로 가중치가 처음부터 새로 학습되지 않습니다.
- GoogLeNet과 ResNet처럼 널리 쓰이는 아키텍처를 포함해 딥러닝 연구 커뮤니티에서 개발되는 모델 아키텍처를 활용할 수 있습니다.

High-Level Vision (이미지 분류) [45]

• Convolutional Neural Network (Learning-Based)

()

- Transfer Learning (전이학습)

전이학습을 위한 사전 훈련된 모델

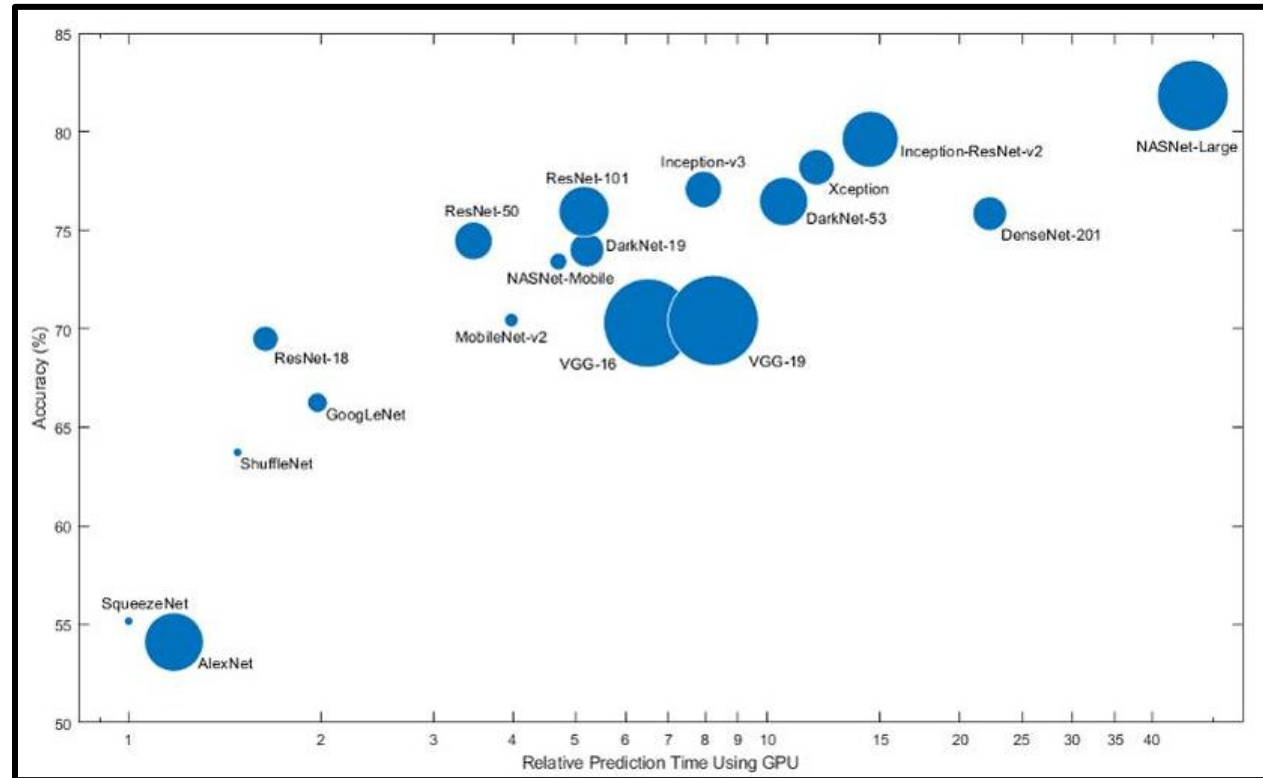
전이학습에서는 딥러닝 연구자들이 구축한 *사전 훈련된 딥러닝 모델*이 핵심이라 할 수 있는데, 이러한 모델은 수천 또는 수백만 개의 샘플 훈련 영상을 통해 훈련되었습니다.

다수의 사전 훈련된 모델을 사용할 수 있는데, 각각 다음과 같은 부문에서 장단점이 있습니다.

- **크기:** 어느 정도의 메모리 사용량이 모델에 적합한가? 모델 크기의 중요성은 모델을 배포할 위치와 방법에 따라 달라집니다. 임베디드 하드웨어와 데스크탑 중 어디에서 모델이 실행되니까? 신경망의 크기는 메모리 용량이 작은 시스템에 배포할 때 특히 중요합니다.
- **정확도:** 재훈련 전의 모델 성능은 어느 정도인가? 널리 쓰이는 데이터셋으로서 백만 개의 영상과 천 개의 영상 클래스가 포함된 ImageNet에서 성능이 우수한 모델은 일반적으로 그와 비슷한 새로운 작업에서도 잘 작동할 가능성이 큼니다. 그러나 ImageNet에서 정확도 점수가 낮다고 해서 모든 작업에서 모델의 성능이 떨어진다는 의미는 아닙니다.
- **예측 속도:** 새로운 입력에 대한 모델의 예측 속도가 얼마나 빠른가? 예측 속도는 하드웨어 및 배치 크기와 같은 다른 딥러닝 요소는 물론이고 선택된 모델의 아키텍처와 모델 크기에 따라서도 달라집니다.

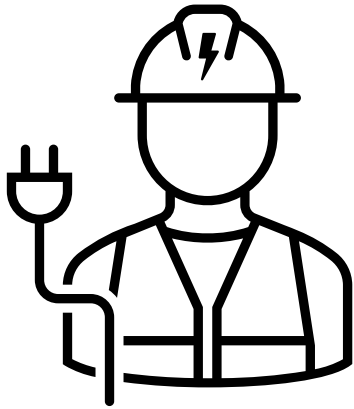
High-Level Vision (이미지 분류) [46]

- Convolutional Neural Network (Learning-Based)
 - Transfer Learning (전이학습)



High-Level Vision (이미지 분류) [47]

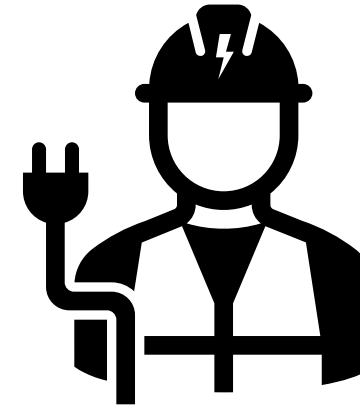
- Convolutional Neural Network (Learning-Based)
 - Transfer Learning (전이학습)



유사 직종 경력 사원
(Pretrained Model)



사내 교육
(Transfer Learning)



해당 분야 전문가
(Trained Model)

High-Level Vision (이미지 분류) [48]

- Convolutional Neural Network (Learning-Based)

- 전이학습 (실습)

- ▶ PyTorch Pretrained Models

- | | |
|----------------------|---------------------|
| ▷ AlexNet | ▷ RegNet |
| ▷ ConvNeXt | ▷ ResNet |
| ▷ DenseNet | ▷ ResNeXt |
| ▷ EfficientNet | ▷ ShuffleNetV2 |
| ▷ EfficientNetV2 | ▷ SqueezeNet |
| ▷ GoogLeNet | ▷ SwinTransformer |
| ▷ Inception V3 | ▷ VGG |
| ▷ MaxVit | ▷ VisionTransformer |
| ▷ MNASNet | ▷ Wide ResNet |
| ▷ MobileNetV2 | |
| ▷ MobileNetV3 | |

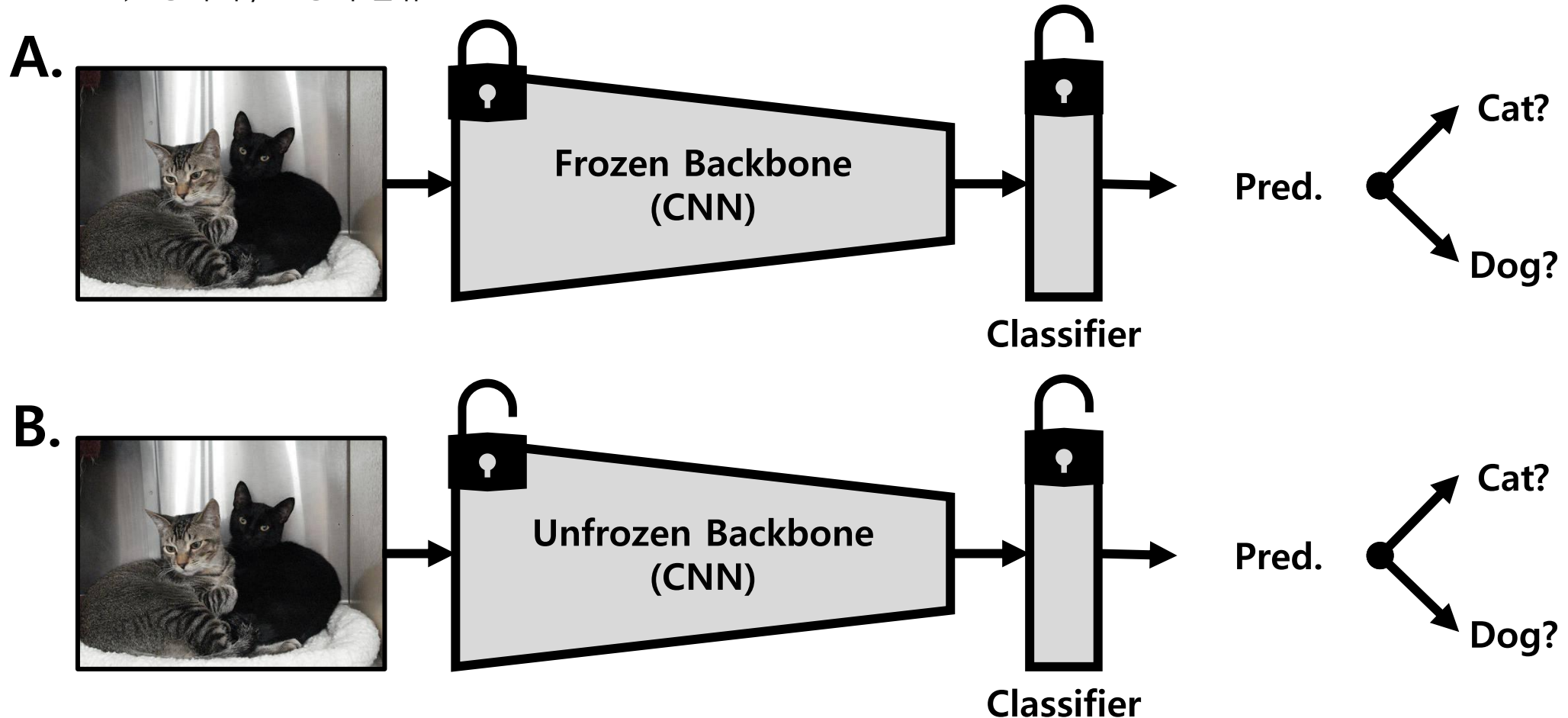


High-Level Vision (이미지 분류) [49]

- Convolutional Neural Network (Learning-Based)

- 전이학습 (실습)

- ▶ 강아지 / 고양이 분류



High-Level Vision (이미지 분류) [50]

- **Convolutional Neural Network (Learning-Based)**

- 전이학습 (개인 실습)

- ▶ 과일 분류 (1시간)

1. 데이터셋/img/fruit.zip에 맞는 Custom Dataloader를 제작하세요.
2. 해당 데이터셋을 활용하여 Transfer Learning을 진행하세요.
 - I. MobileNetV3 Unfrozen Backbone
 - II. MobileNetV3 Frozen Backbone
3. 두 가지 방식의 성능을 비교하세요.
4. Hyperparameter를 변경하며 Image Classification의 성능을 높여보세요.