



2024 D&A

# CNN 심화 1

(LeNet, AlexNet, VGG)

Deep Session 5차시



# CONTENTS.

---

## 01. CNN 모델 개요

---

- CNN
- CNN 모델 발전
- ILSVRC

## 02. 기초 모델 아키텍처

---

- LeNet
- VGG

## 03. 논문 리뷰

---

- AlexNet

# CNN 모델 개요

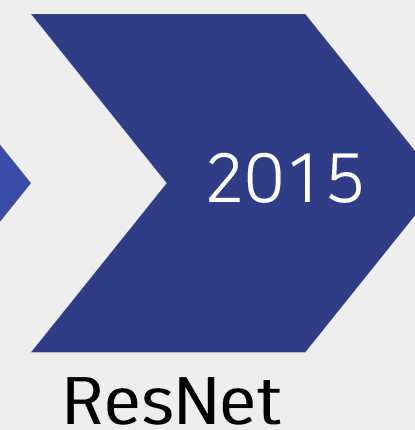
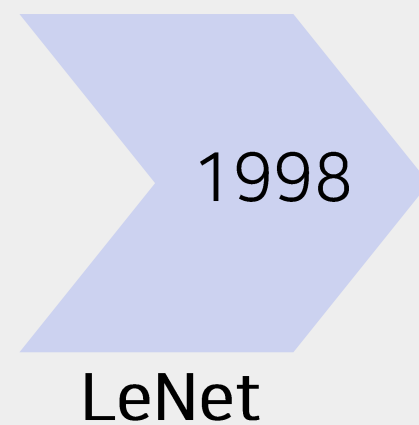
## CNN

### CNN이 각광받은 이유

- 이미지를 flat하게 펼쳐서 MLP로 학습 가능 → 이미지의 지역적인 정보가 날아가버린다는 단점 존재
- CNN은 이미지의 공간 정보를 유지한 상태로 학습이 가능한 모델

→ CNN은 Convolution과 Pooling을 반복적으로 사용하면서 불변하는 특징을 찾고  
그 특징을 입력 데이터로 Fully-connected 신경망에 보내 Classification을 수행하는 것

### CNN 모델의 발전

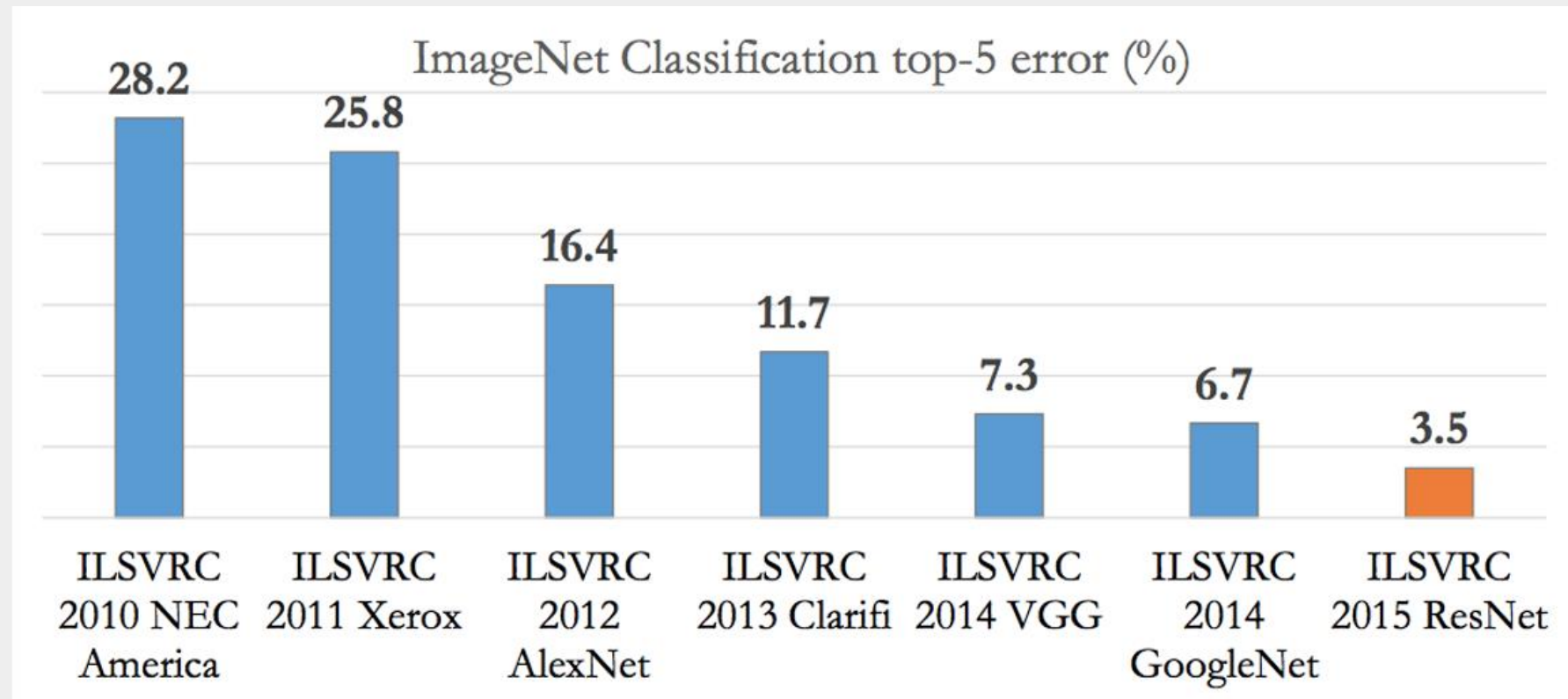


## ILSVRC (ImageNet Large Scale Visual Recognition Challenge)

### ILSVRC

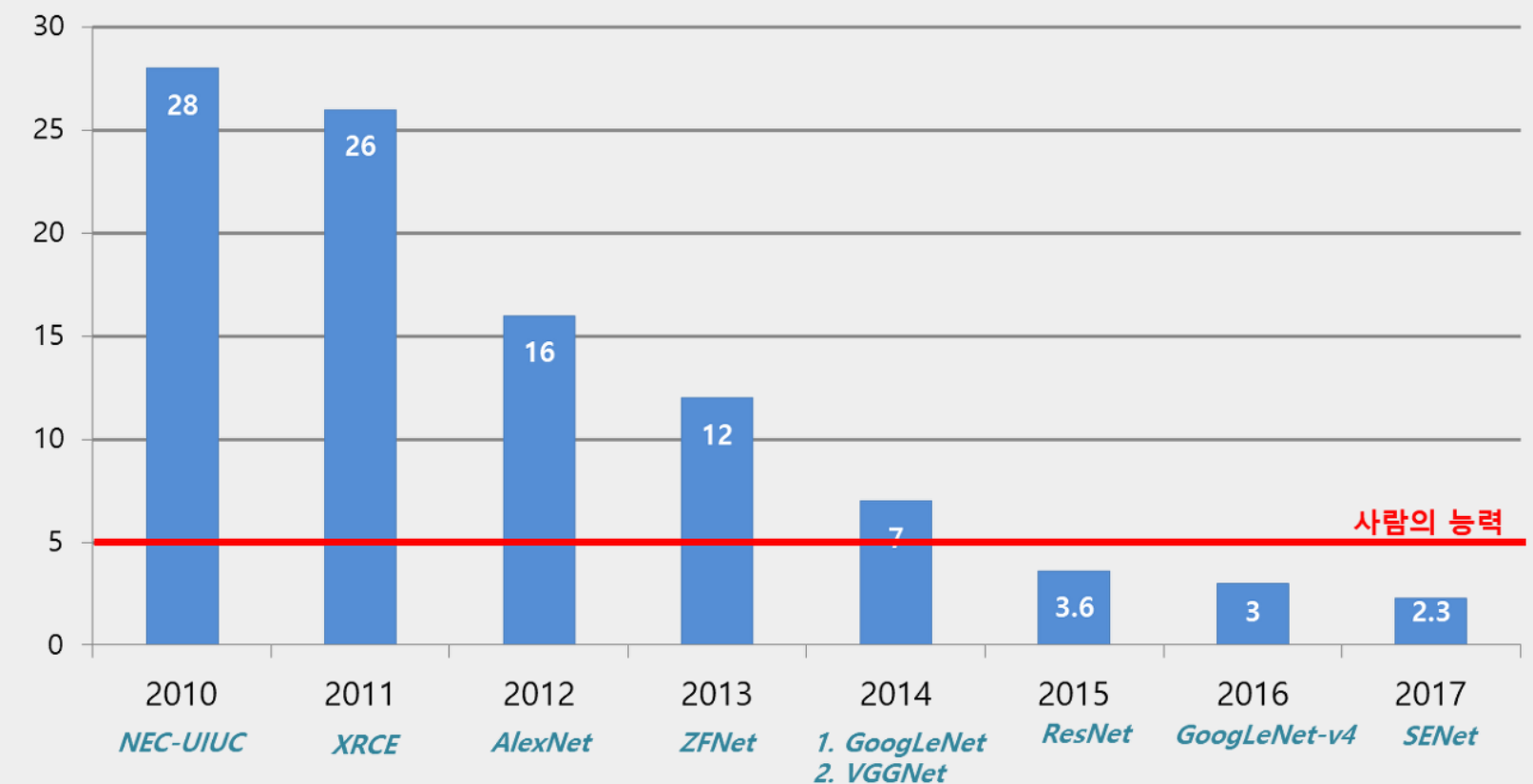
이미지 인식 경진대회

- 대용량의 이미지 데이터셋을 주고 이미지 분류 알고리즘의 성능을 평가



출처: [AlexNet에 대해 알아보자! | GDSC UOS \(gdsc-university-of-seoul.github.io\)](#)

### 우승 알고리즘의 분류 에러율(%)



출처: [ILSVRC 대회 \(이미지넷 이미지 인식 대회\) 역대 우승 알고리즘들 by bskyvision.com](#)

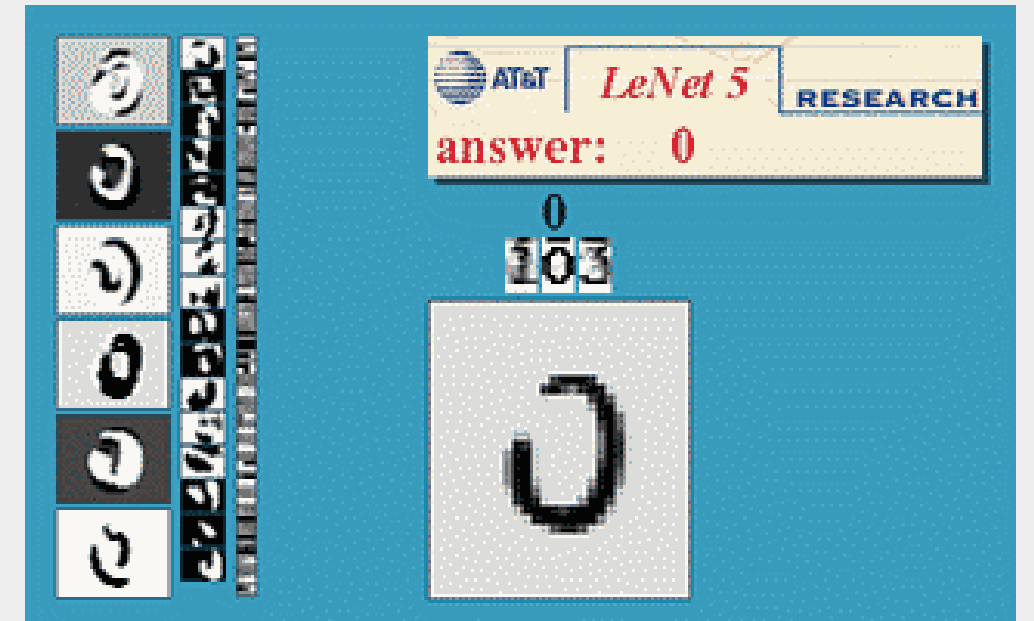
# 기초 모델 아키텍처

## LeNet

### 개요

- Yann LeCun이 개발한 CNN architecture
- 기존 DNN의 문제점을 해결하기 위해 만들어짐
- 손글씨 숫자를 인식하는 네트워크 → MNIST 데이터셋 사용 (0~9 손글씨)

### 구조



ChatGPT 4 ▾

EX

You

LeNet을 대표하는 그림을 그려줘

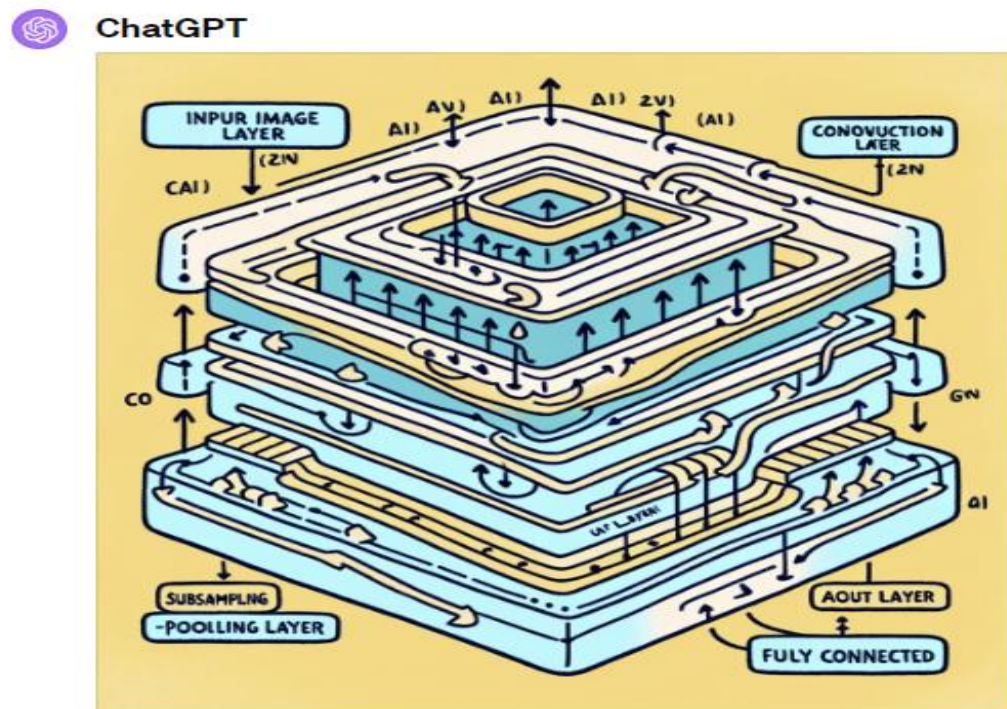


# 기초 모델 아키텍처

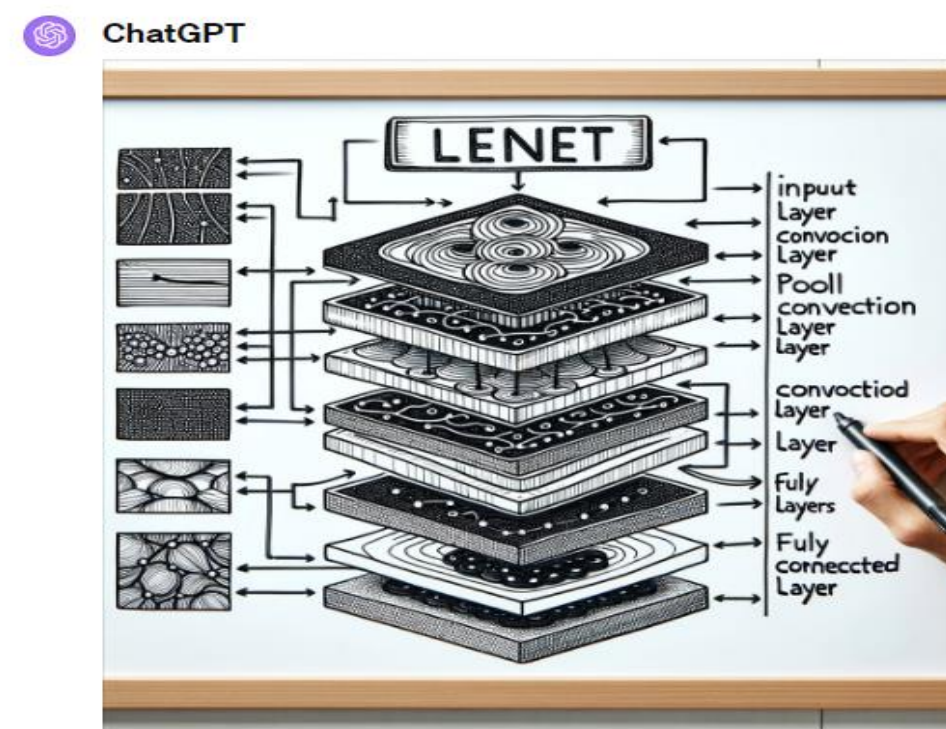
## LeNet

ChatGPT 4 ▾

EX You  
LeNet을 대표하는 그림을 그려줘



Here is an illustration of the LeNet architecture. This classic deep learning model for image recognition is depicted with the sequence of layers from the input image layer through convolutional, subsampling (pooling), fully connected layers, and finally to the output layer. The flow from one layer to the next is indicated by arrows, providing a clear understanding of how data moves through the network.

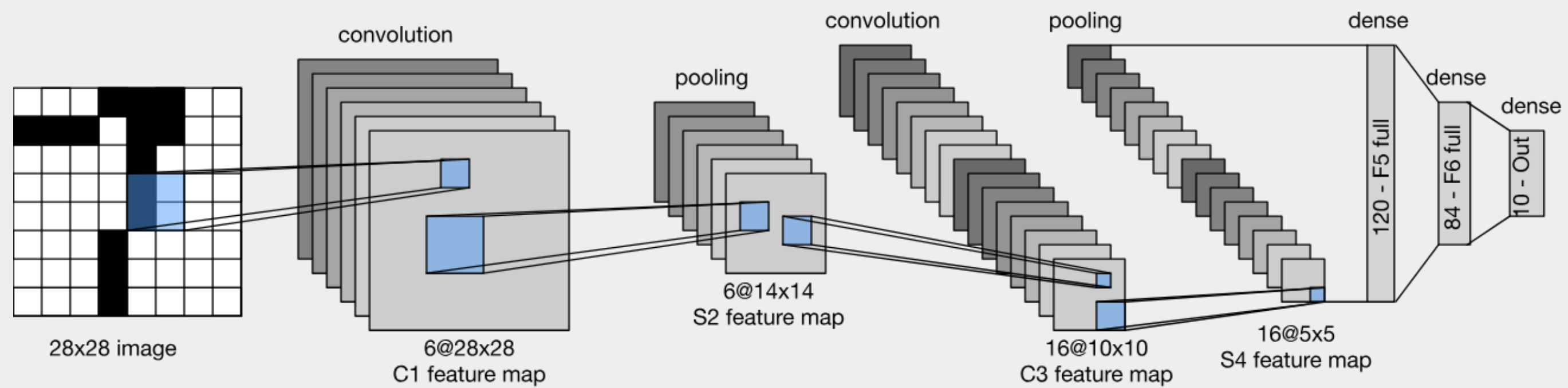


Here is a hand-drawn style diagram illustrating the LeNet architecture. This visual captures the typical layout and flow of a LeNet neural network, from the input layer through the various types of layers to the output.

# 기초 모델 아키텍처

## LeNet

### 배경

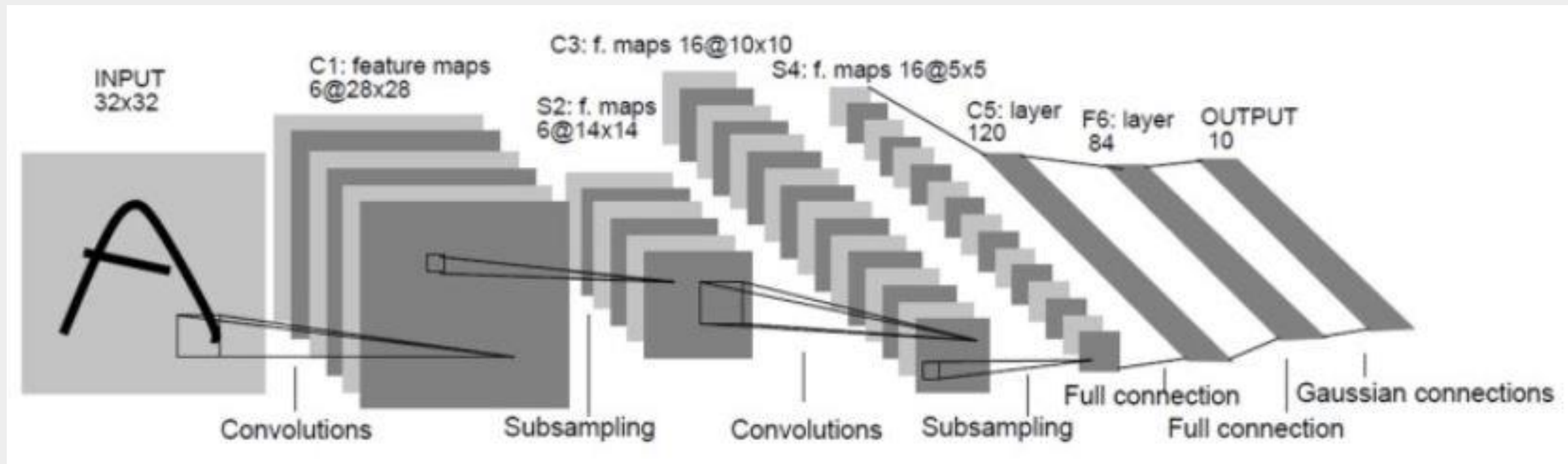


# 기초 모델 아키텍처

## LeNet

### 구조

- 32 x 32 크기의 흑백 이미지에서 학습된 7 layer CNN (Input 층 제외)
- Input → Conv (C1) → Subsampling (S2) → Conv (C3) → Subsampling (S4) → Conv (C5) → FC6 → FC7 (Output)





# 기초 모델 아키텍처

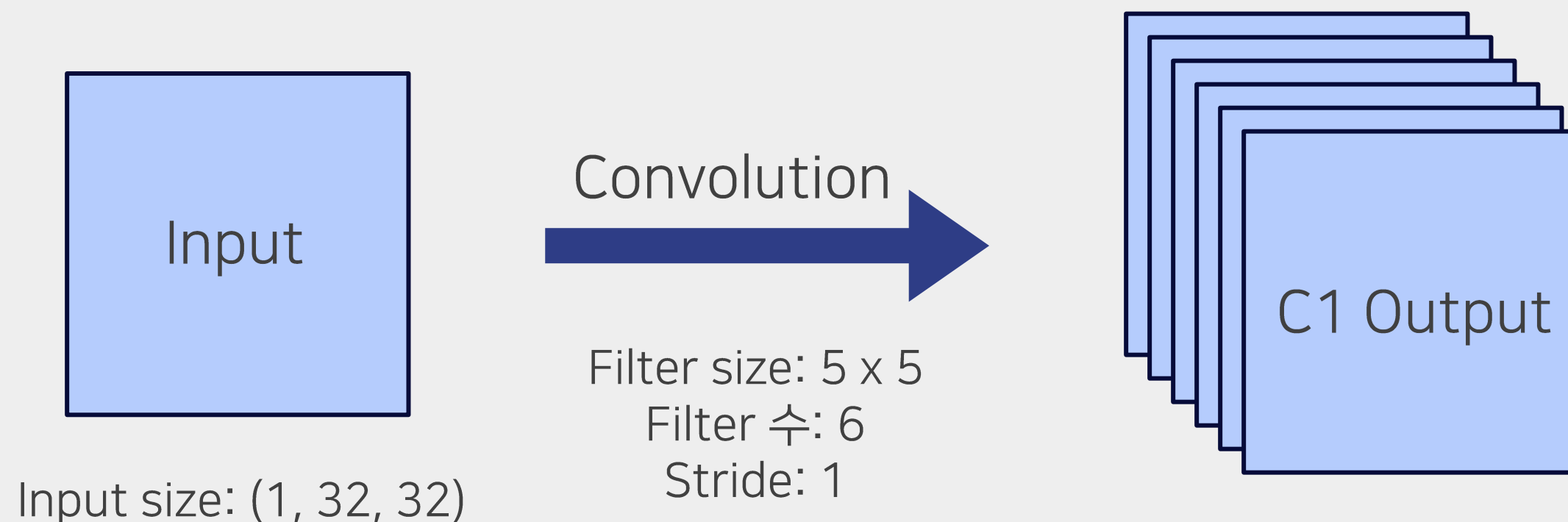
## LeNet

### Conv1 layer

32 x 32 사이즈의 입력 이미지를 6개 (= 6 채널)의 5 x 5 필터와 컨볼루션 연산을 진행

→ 6장의 28 x 28 피쳐 맵을 얻게 됨

→ 훈련해야 할 파라미터 수 = (가중치 x 입력 맵 개수 + 편향) x 피쳐 맵 개수 =  $(5 \times 5 \times 1 + 1) \times 6 = 156$   
(여기서 가중치 = 필터 사이즈인 5 x 5)



# 기초 모델 아키텍처

## LeNet

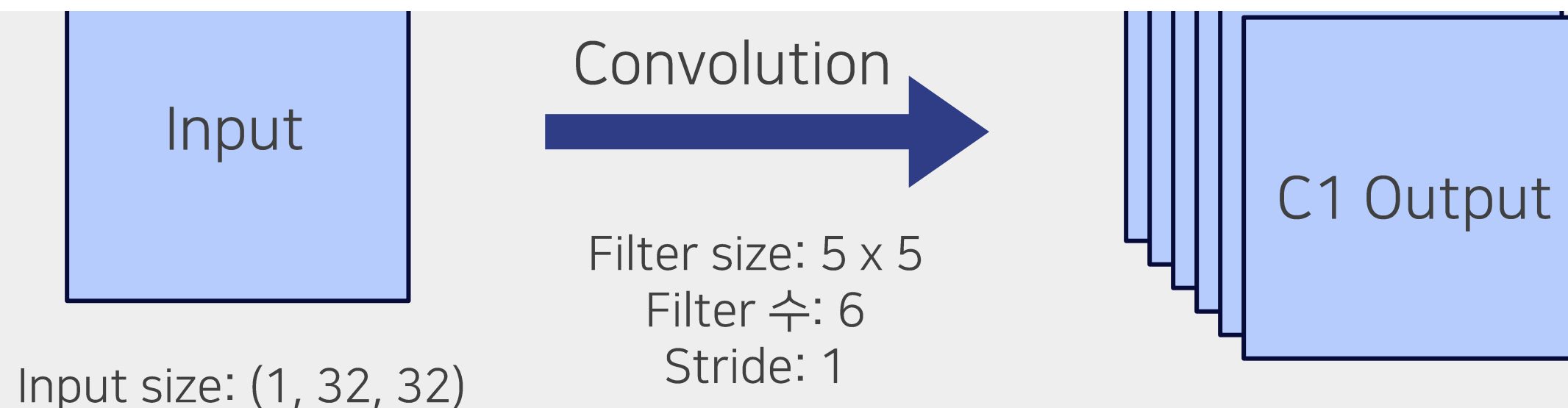
### Conv1 layer

32 x 32 사이즈의 입력 이미지를 6개 (= 6 채널)의 5 x 5 필터와 컨볼루션 연산을 진행

→ 6개의 28 x 28 피쳐 맵이 생성되

→ 후  
(여

$$Output\ size = \frac{input\ size - filter\ size + (2 * padding)}{Stride} + 1$$



# 기초 모델 아키텍처

## LeNet

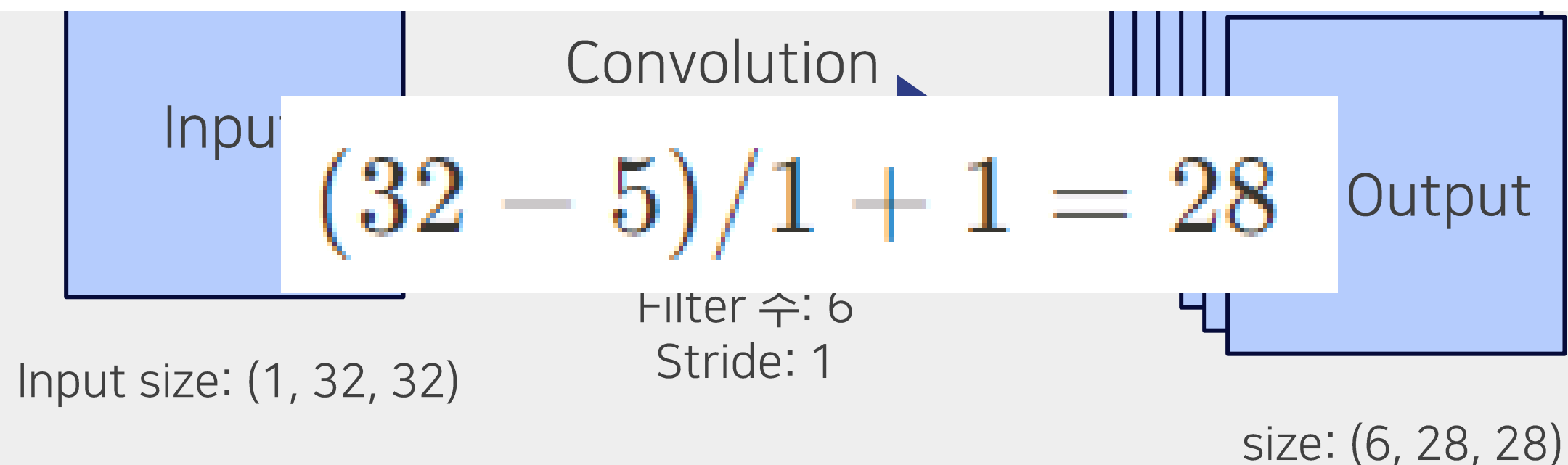
### Conv1 layer

32 x 32 사이즈의 입력 이미지를 6개 (= 6 채널)의 5 x 5 필터와 컨볼루션 연산을 진행

→ 6개의 28 x 28 피쳐 맵이 생성되

→ 후  
(여

$$Output\ size = \frac{input\ size - filter\ size + (2 * padding)}{Stride} + 1$$



# 기초 모델 아키텍처

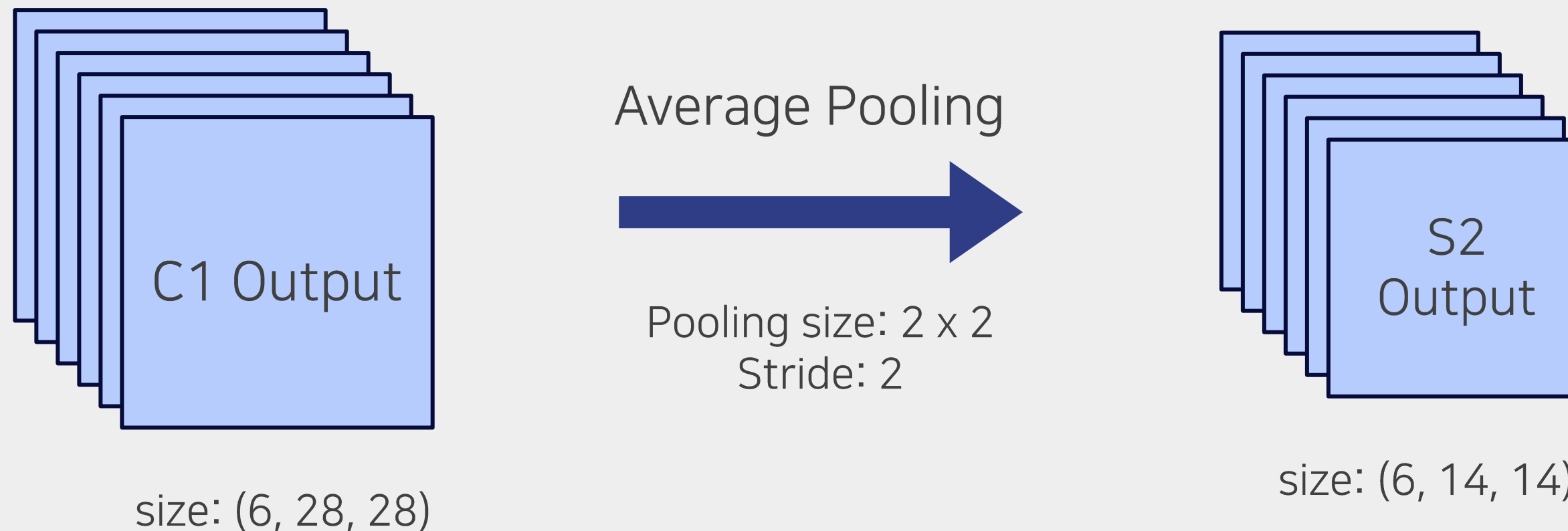
## LeNet

### S2 layer

6장의  $28 \times 28$  피쳐 맵에 대해 Subsampling 진행해서 결과적으로 6장의  $14 \times 14$  피쳐 맵으로 축소됨

→ Subsampling = Pooling (여기서는 평균 풀링을 사용)

→ 훈련해야 할 파라미터 수 = (가중치 + 편향)  $\times$  피쳐 맵 개수 =  $(1 + 1) \times 6 = 12$

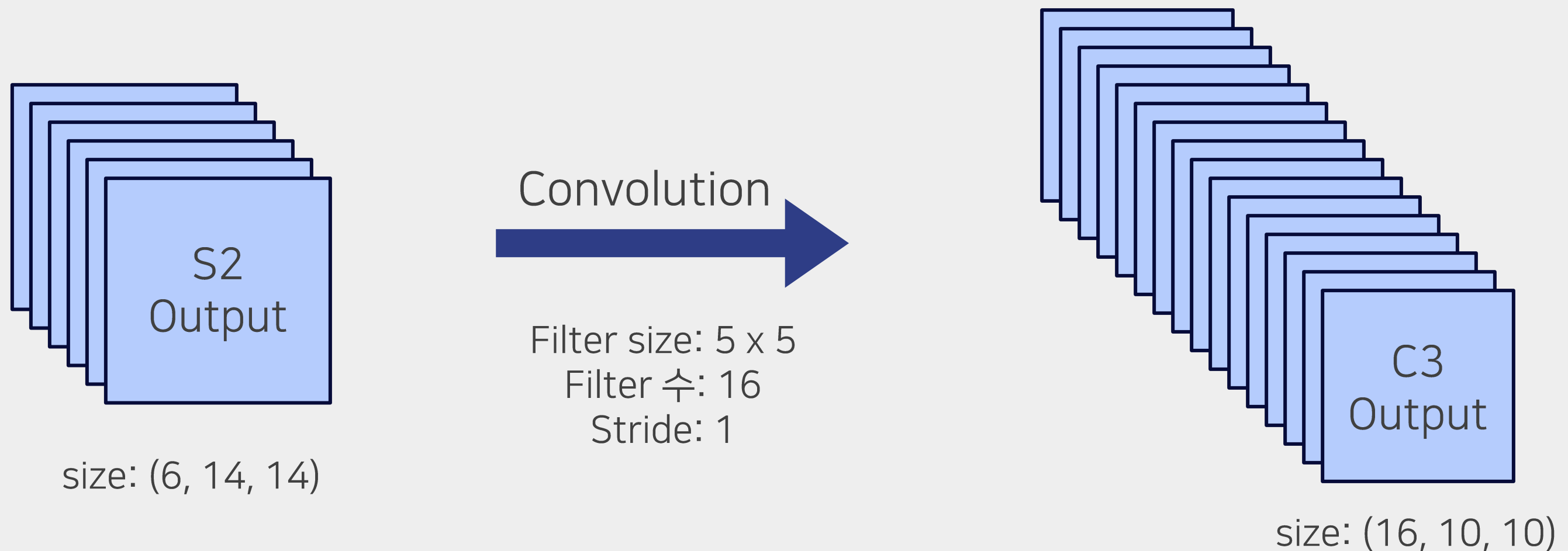


# 기초 모델 아키텍처

## LeNet

### Conv3 layer

6장의  $14 \times 14$  피쳐 맵에  $5 \times 5$  커널의 컨볼루션 연산을 수행해서 16장의  $10 \times 10$  피쳐 맵을 만들어냄

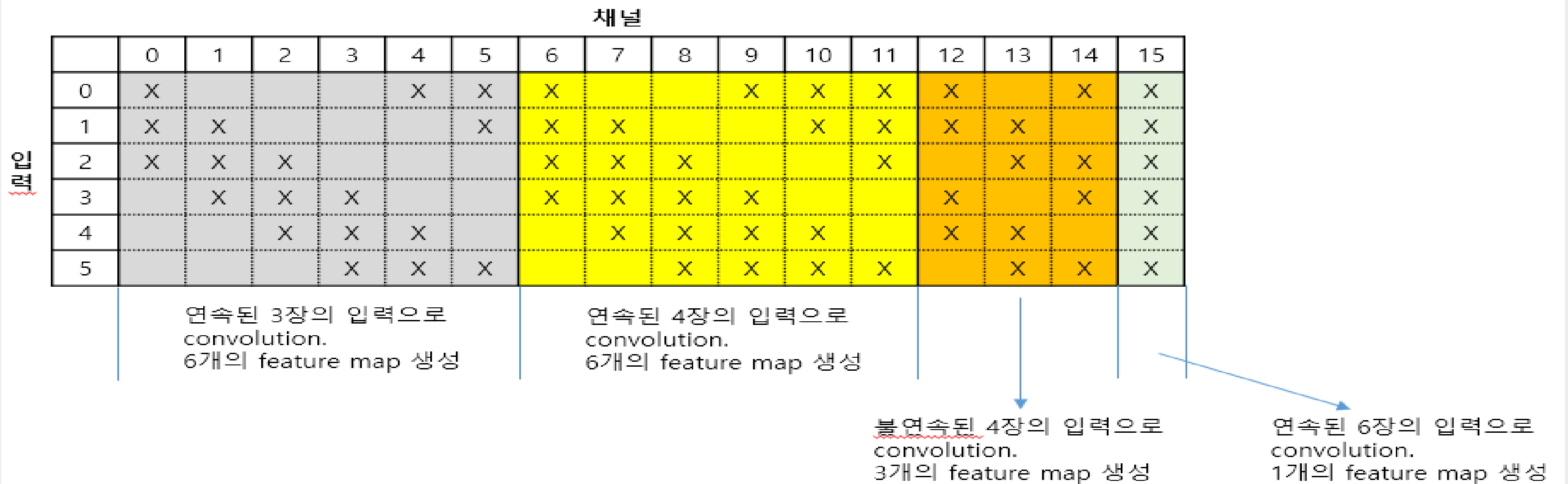


# 기초 모델 아키텍처

## LeNet

### Conv3 layer

네트워크의 대칭성을 깨뜨리고 네트워크 간의 연결 파라미터 수를 합리적으로 하기 위해서 이처럼 구성함



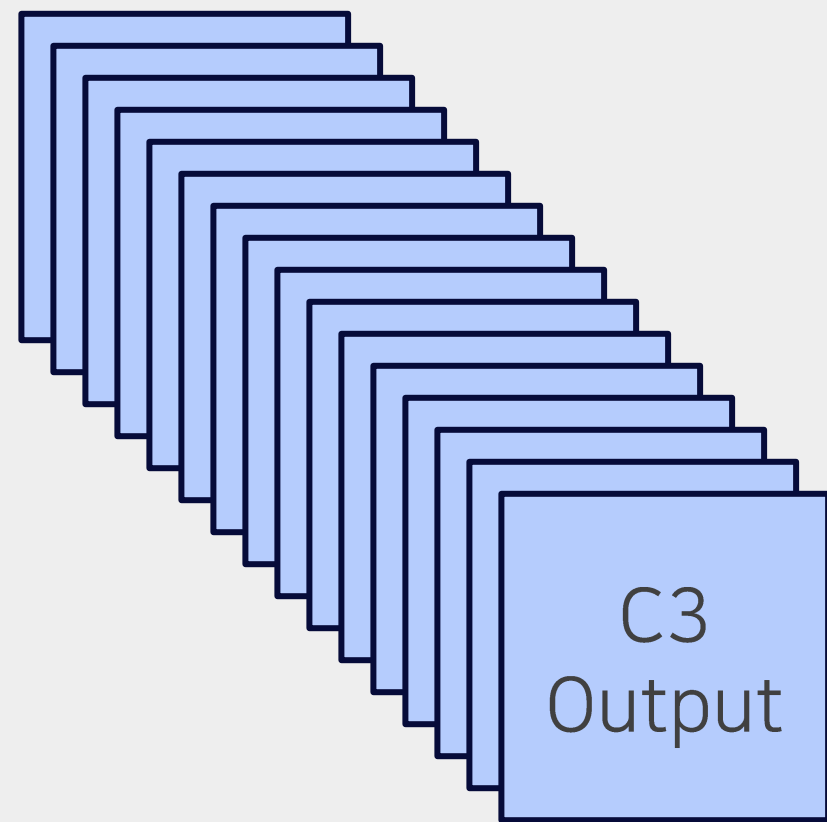


# 기초 모델 아키텍처

## LeNet

### S4 layer

16장의 10 x 10 피쳐 맵에 대해 Subsampling 진행해서 결과적으로 16장의 5 x 5 피쳐 맵으로 축소됨  
→ 훈련해야 할 파라미터 수 = (가중치 + 편향) x 피쳐 맵 개수 =  $(1 + 1) \times 16 = 32$

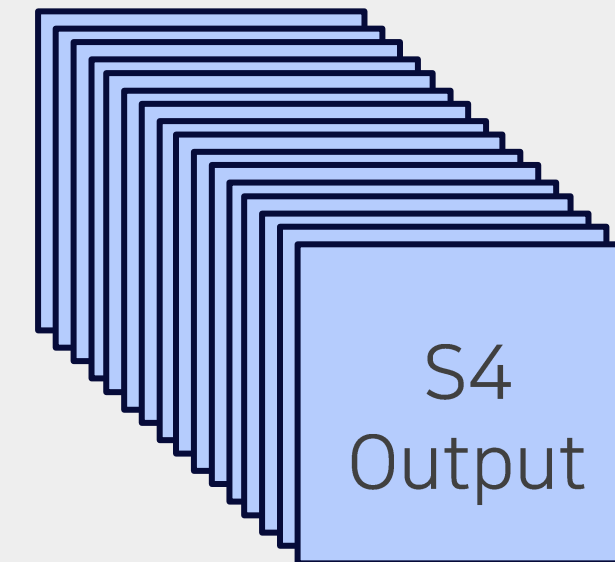


size: (16, 10, 10)

Average Pooling



Pooling size: 2 x 2  
Stride: 2



size: (16, 5, 5)

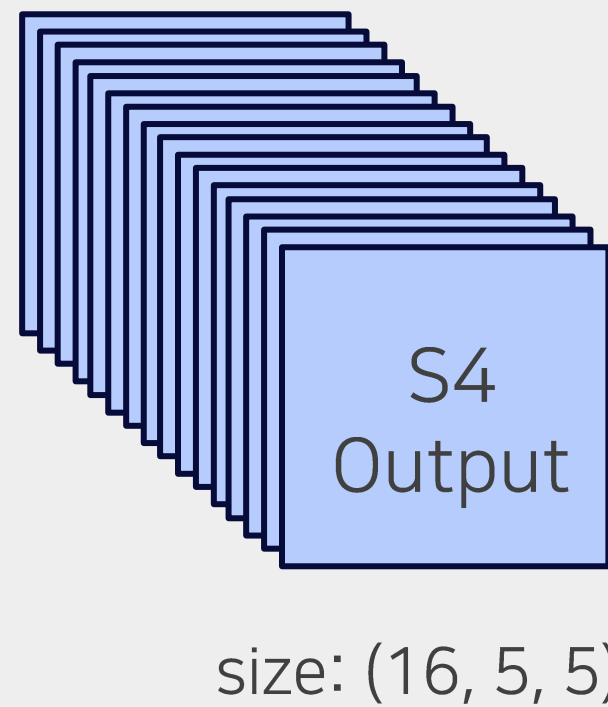
# 기초 모델 아키텍처

## LeNet

### Conv5 layer

16장의  $5 \times 5$  피쳐 맵을 120개의  $5 \times 5 \times 16$  사이즈의 필터와 컨볼루션하고 120개의  $1 \times 1$  피쳐 맵을 만들어냄

→ 훈련해야 할 파라미터 수 = (가중치  $\times$  입력 맵 개수 + 편향)  $\times$  피쳐 맵 개수 =  $(5 \times 5 \times 16 + 1) \times 120 = 48120$



Convolution



Filter size:  $5 \times 5$   
Filter 수: 120  
Stride: 1

C  
5  
O  
u  
t  
p  
u  
t

size: (120, 1, 1)

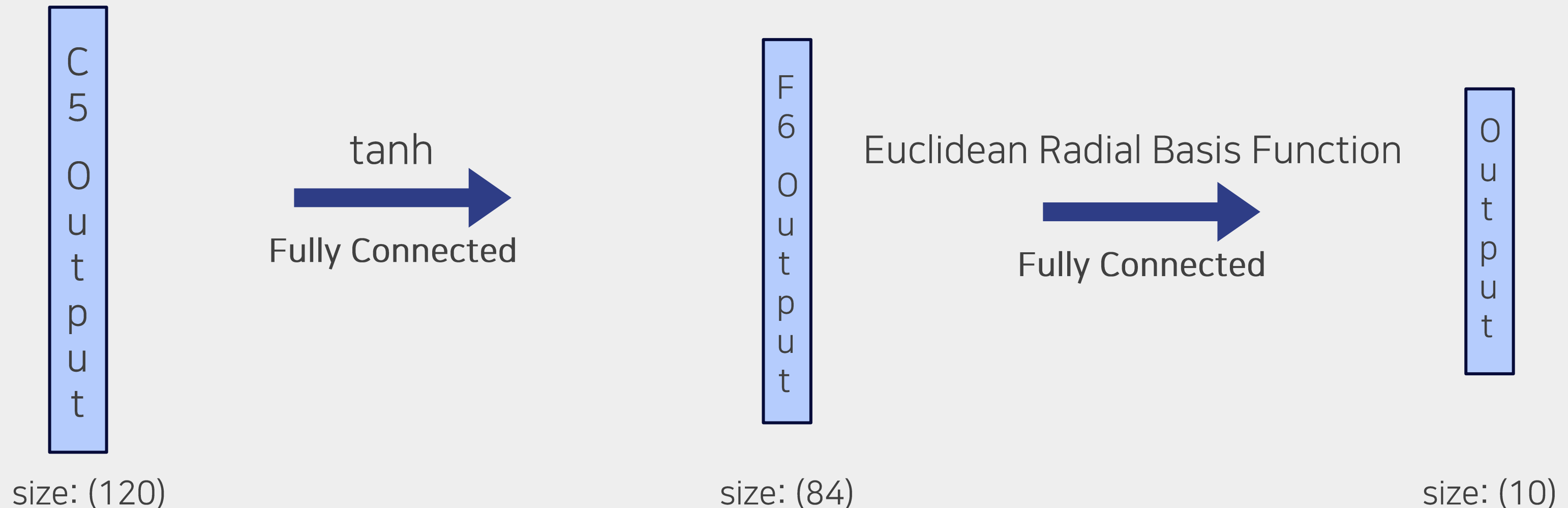
# 기초 모델 아키텍처

## LeNet

### Fully Connected layer

84개의 유닛을 가진 피드포워드 신경망

10개의 Euclidean radial basis function(RBF) 유닛들로 구성된 Output layer (최종적으로 이미지가 속한 클래스를 알려줌)

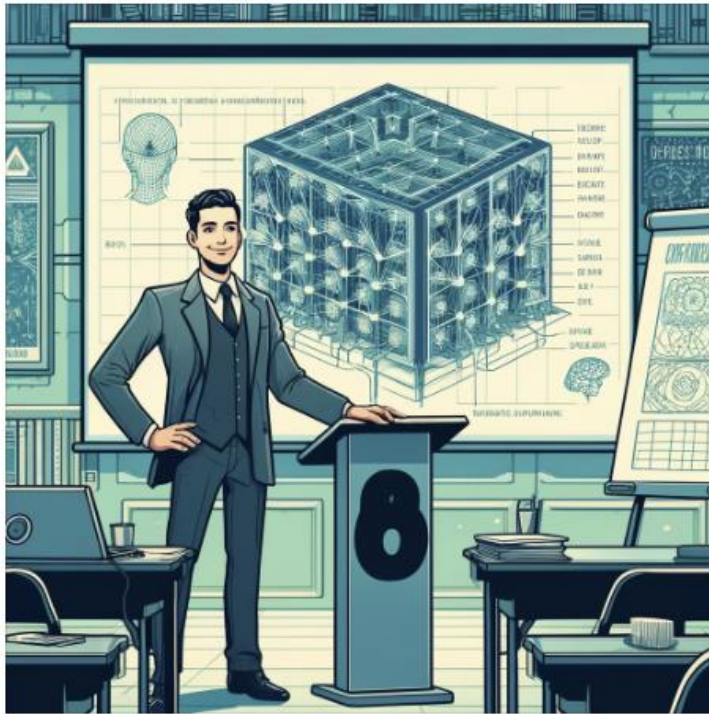


# 기초 모델 아키텍처

## VGG

**You**  
VGG를 자랑스럽게 소개하는 학자를 그림으로 그려줘! 딱 봐도 VGG임이 딱 드러나게!

**ChatGPT**



**You**  
아니 그게 아니라 VGG인 게 딱 보면 알게 그려달라고!! 그리고 저렇게 복잡하게 그리지마!! 그리고 한 국어로 대답해줘

**ChatGPT**



이번엔 VGG 네트워크의 특징을 간단하고 명확하게 드러내는 그림을 준비했어요. 발표자가 학술 대회에서 자신감 있게 VGG 네트워크를 소개하고 있는 모습을 담았습니다. 포스터에는 'VGG 네트워크'라고 크게 적혀 있고, 간단한 레이어 구조가 도식화되어 있어 VGG임을 한눈에 알아볼 수 있습니다.

🔊 📄 🔄 🗑️

**You**  
아니 애가 왜 이럴까... VGG가 한눈에 보이게 ... 자랑스러워하는 사람도 보이게...

**ChatGPT**



이번엔 학자가 대형 VGG 네트워크 아키텍처 다이어그램 옆에서 자랑스럽게 서 있는 모습을 그렸어요. 'VGG'라는 글자와 함께 여러 층의 컨볼루션 레이어와 풀링 레이어가 순차적으로 배열된 구조가 단순화하여 표현되어 있습니다. 학자의 행복하고 자랑스러운 표정이 VGG 네트워크에 대한 자신감을 잘 보여주고 있죠.

🔊 📄 🔄 🗑️

# 기초 모델 아키텍처

## VGG

### 목표

더 깊은 네트워크를 만들면서, 성능이 높은 모델을 만들고 싶다!

### 동기

더 깊은 네트워크를 만들고 싶은데  
파라미터 수가 너무 많아지면...

→ 과적합될 가능성 ↑

→ 계산 비용 ↑

### 핵심

1. 단순한 네트워크 구조
2. 기울기 소실 문제 해결
3. 깊은 네트워크 구현

### 단점

파라미터 개수가 너무 많음

→ GoogleNet에서  
Avg Pooling layer를  
사용함으로써 해결

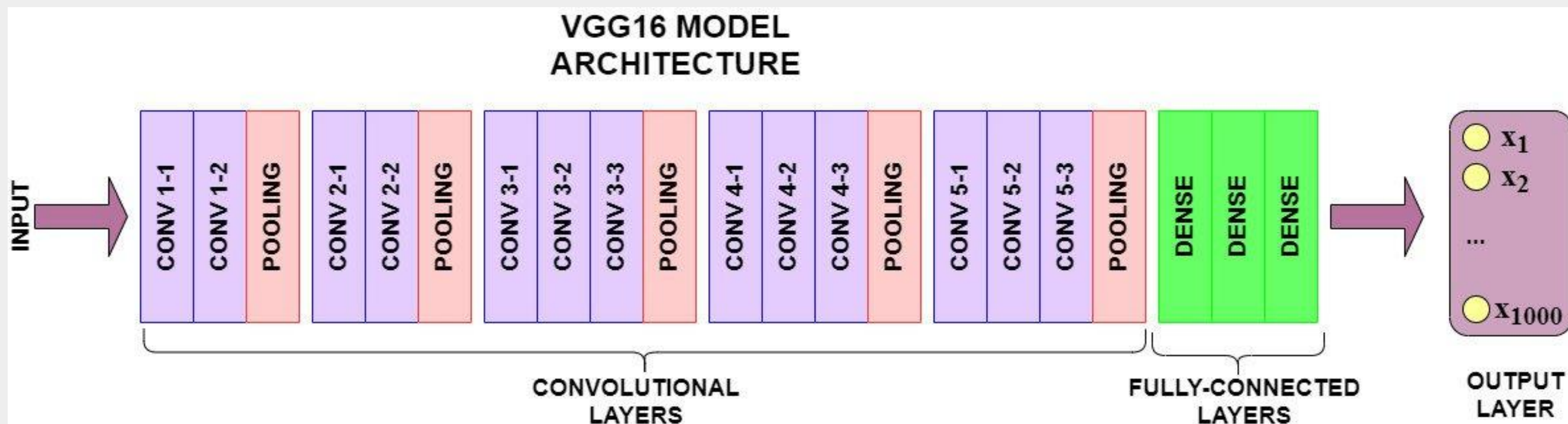


# 기초 모델 아키텍처

## VGG

### 개요

- VGG 부터 네트워크의 깊이가 확 깊어짐
- 3 x 3 의 작은 필터를 사용한 컨볼루션 레이어를 깊게 중첩한다는 것이 가장 큰 특징
- [Input - C1 - C2 - MaxPool2 - C3 - C4 - MaxPool4 - C5 - C6 - C7 - MaxPool7 - C8 - C9 - C10 - MaxPool10 - C11 - C12 - C13 - MaxPool13 - FC14 - FC15 - FC16(Output)]

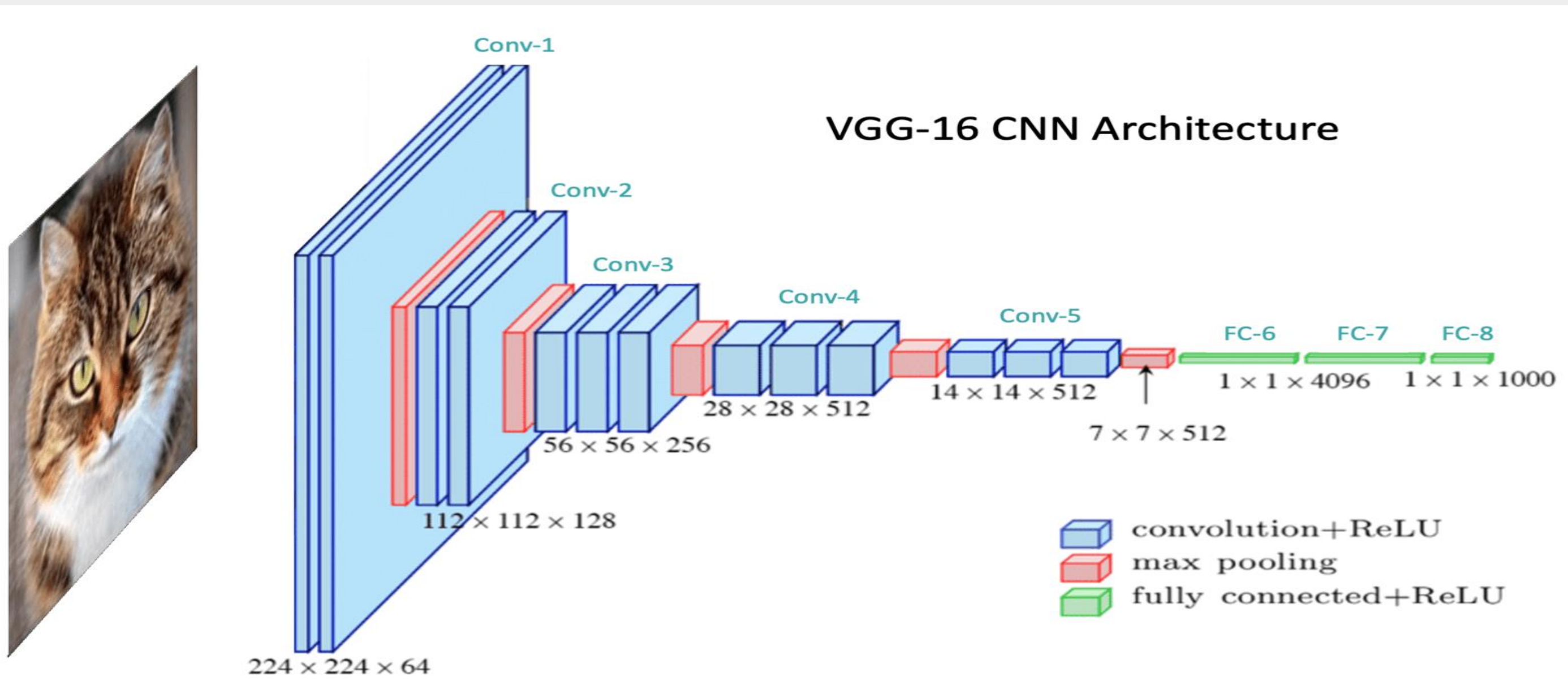




# 기초 모델 아키텍처

## VGG

### 구조



# 기초 모델 아키텍처

## VGG

### 특징

깊은 신경망

3 x 3 Convolution Layer

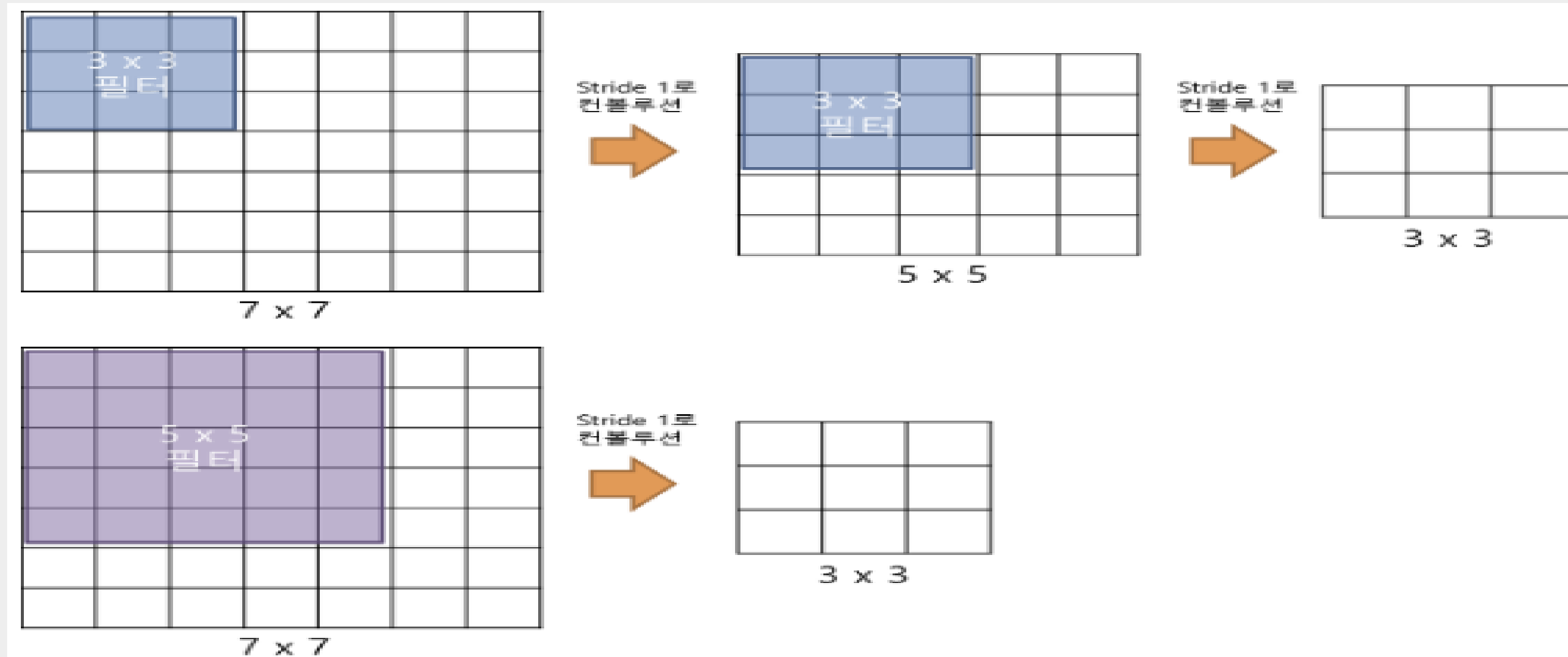
- 이전에 주로 사용하던 5 x 5 필터 대신 Convolutional layer의 커널 (필터) 사이즈를 3 x 3 으로 사용하고 zero-padding을 사용함으로써 Convolutional layer를 더 깊게 쌓을 수 있게 됨
- 층을 깊게 쌓아서 기존 7 x 7, 11 x 11 필터 이상의 표현력을 가질 수 있게 됨

즉, 작은 사이즈의 필터를 사용함으로써 layer를 더 많이 쌓을 수 있게 되고 파라미터 수 증가를 상대적으로 최소화할 수 있음과 동시에 모델이 깊어진 만큼 표현력도 좋아짐

# 기초 모델 아키텍처

## VGG

### 3 x 3 Filter



3x3필터  
3개는  
27개의  
가중치

7x7필터  
1개는  
49개의  
가중치

→ 3 x 3 필터로 2번 Convolution 하는 것과 5 x 5 필터로 1번 Convolution 하는 것은 동일한 사이즈의 Feature Map을 나타냄

# 기초 모델 아키텍처

## VGG

3 x 3 Filter

### Layer가 깊어짐과 동시에...

→ 각 Convolution 연산은 ReLU 함수를 포함하기 때문에 layer가 증가함에 따라 **비선형성 증가**

→ 7x7 filter 로 convolution 연산을 진행하면 ReLU연산이 1번 적용되는 반면

3x3 filter로 convolution 연산을 하면 ReLU연산이 3번 적용 되기 때문

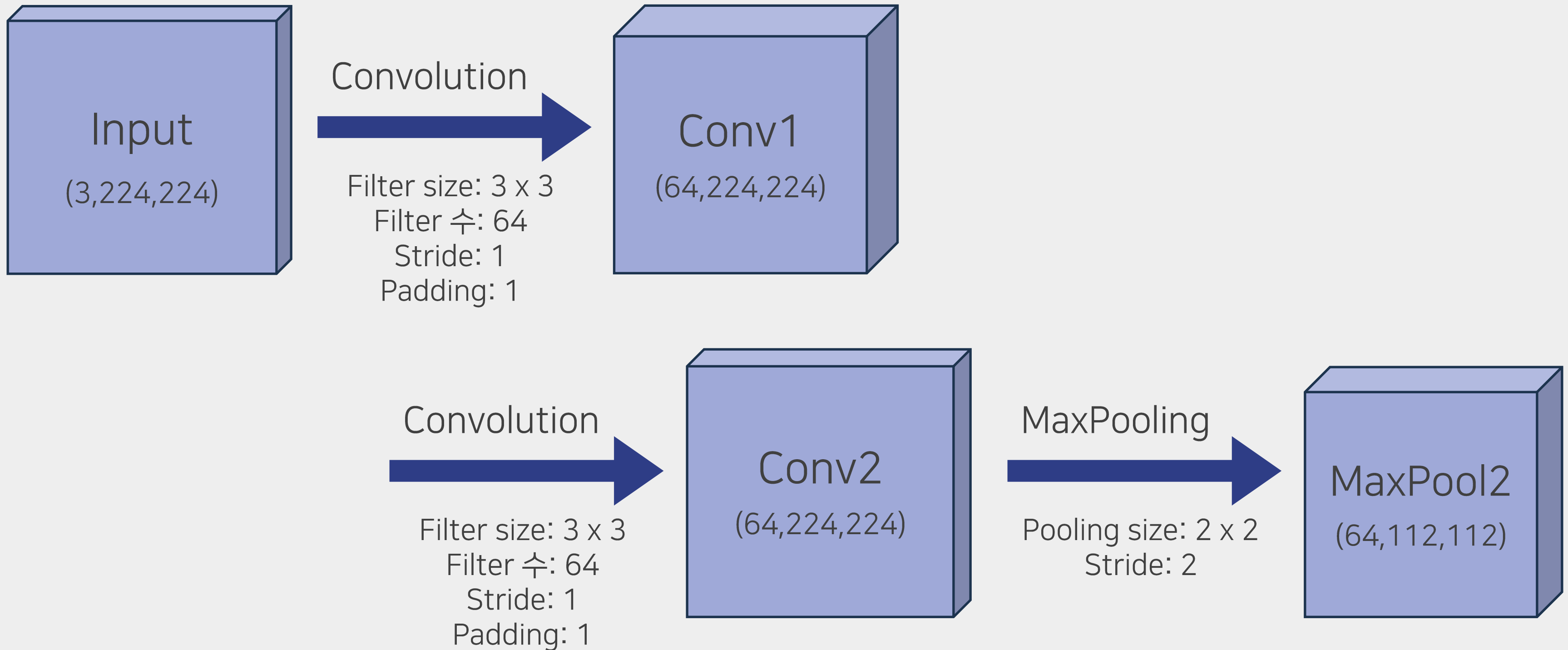
→ 비선형성이 증가하면 모델의 식별성이 증가

→ **학습 파라미터 수 감소**

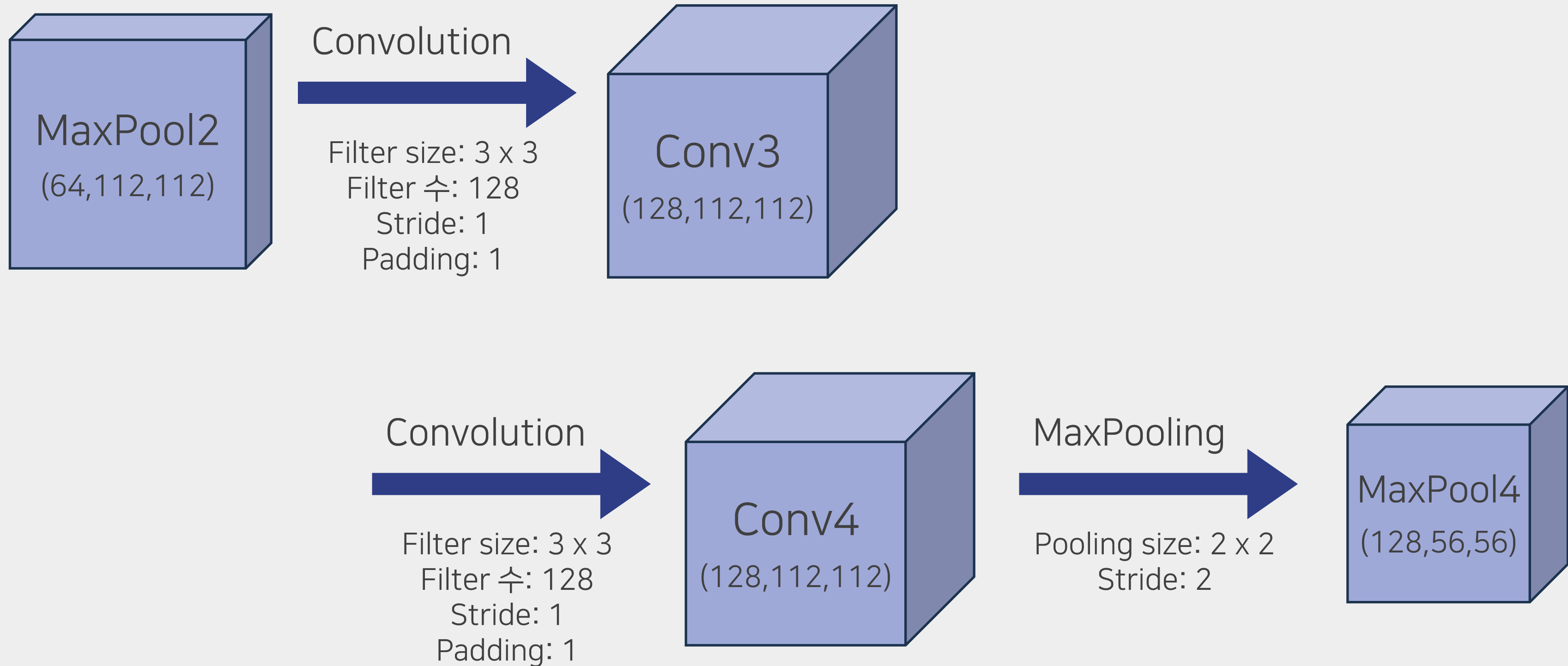
→ 7x7 filter 1개는  $7 \times 7 = 49$ 개의 파라미터가 필요

→ 3x3 filter는  $3 \times 3 \times 3 = 27$ 의 파라미터가 필요

# VGG - Conv1, Conv2, MaxPool2

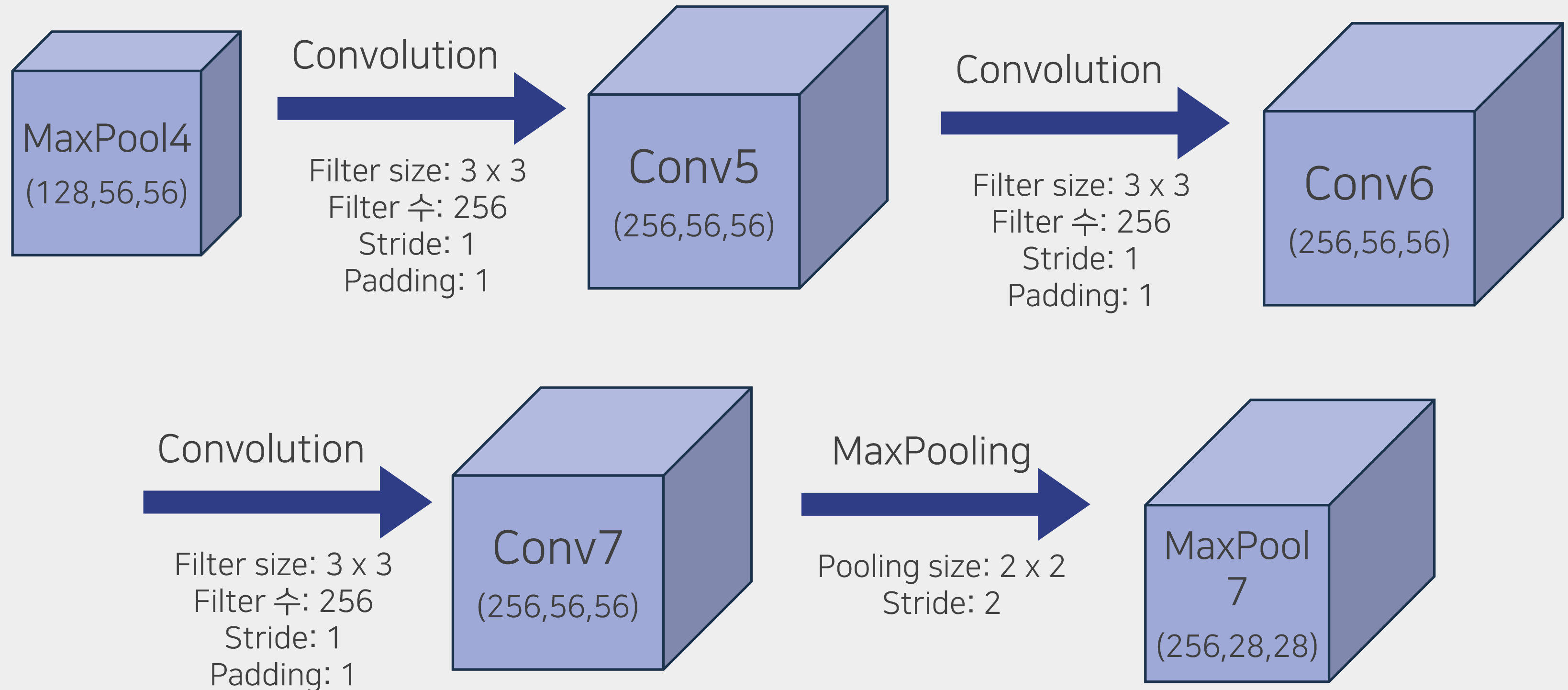


# VGG – Conv3, Conv4, MaxPool4

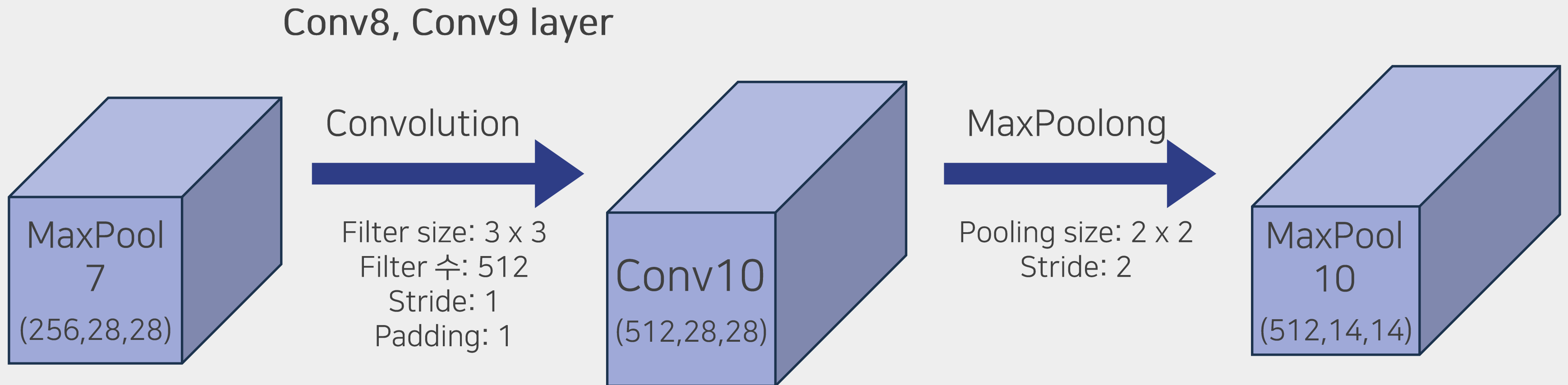




# VGG – Conv5, Conv6, MaxPool7

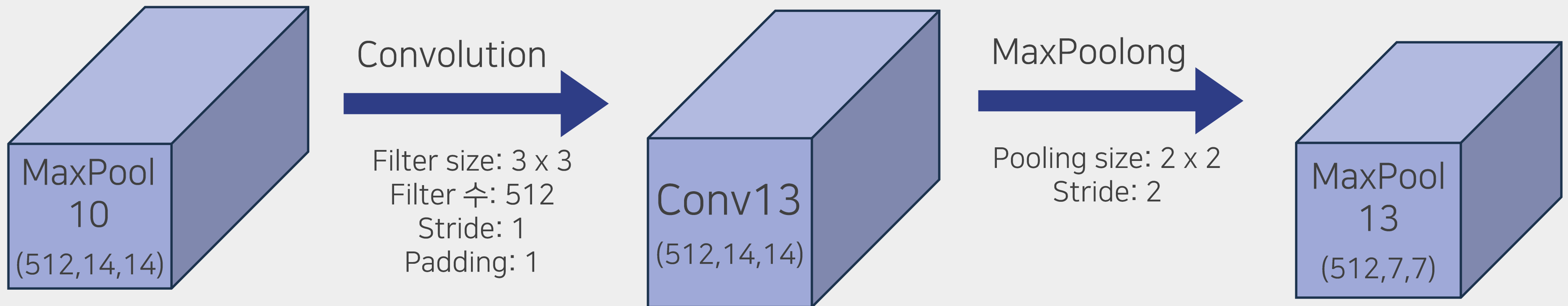


# VGG – Conv8, Conv9, Conv10, MaxPool10

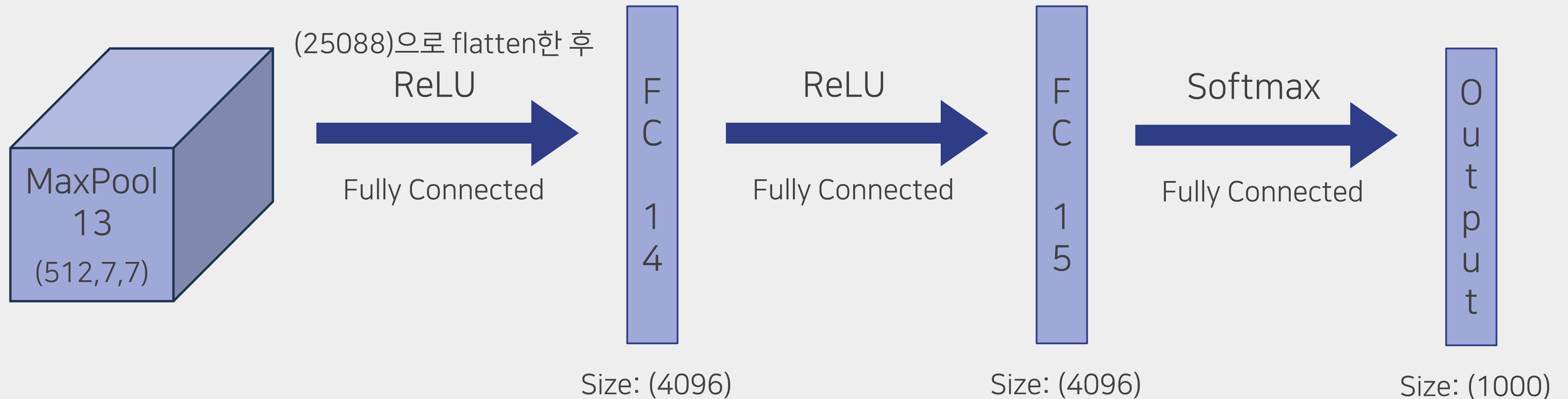


# VGG – Conv11, Conv12, Conv13, MaxPool13

Conv11, Conv12 layer



# VGG – FC14, FC15, FC16



# AlexNet

## 01. Abstract & Introduction

- 핵심 정리
- Abstract
- Introduction

## 02. Dataset

## 03. Architecture

- ReLU Nonlinearity
- Training on Multiple GPUs
- Local Response Normalization
- Overlapping Pooling
- Overall Architecture

## 04. Reducing Overfitting

- Data Augmentation
- Dropout

## 05. Details of learning

## 06. Results

# 과제

## VGG 논문 리뷰

<https://arxiv.org/pdf/1409.1556.pdf>





2024 D&A

**THANK YOU**

2024.04.11

Deep Session 5차시

