Convolutional Neural Network

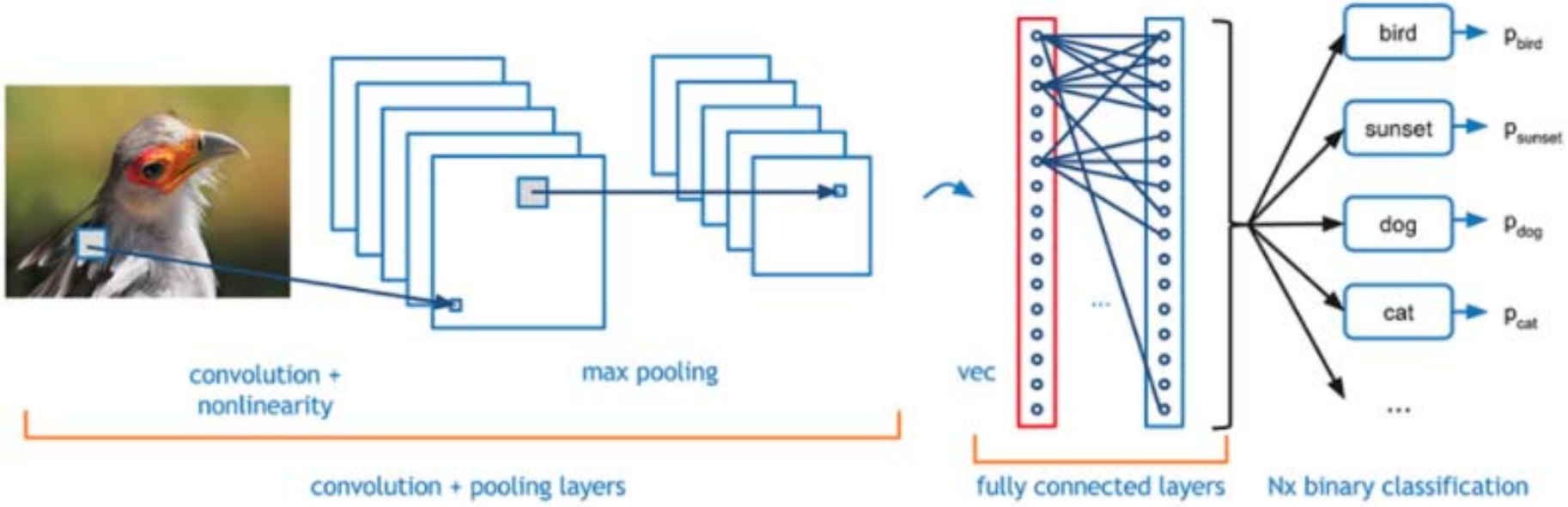
신경망을 이용한 학습 과정의 이해

김 대 환

2022.

합성곱 신경망 (CNN)

- **소개**
  - 페이스북은 자동 태그 알고리즘에, 구글은 사진 검색에, 아마존은 상품 추천에, 핀트리스트는 홈 피드 설정에 그리고 인스타그램은 검색 기반에 신경망을 활용하고 있다
  - 네트워크의 가장 고전적이고 인기 많은 적용 사례는 이미지 처리이다



합성곱 신경망 (CNN)

- **입력과 출력**
  - 컴퓨터는 입력받은 이미지를 픽셀 값으로 이해

What We See

What Computers See

  - 위 이미지의 크기와 해상도에 의해 숫자로 된 32 x 32 x 3 배열 값이 된다 (3은 RGB 값에 대한 표현)
  - 각각의 숫자는 0과 255 사이의 값이며 해당 지점에 대한 픽셀 세기를 표시

합성곱 신경망 (CNN)

- **입력과 출력**
  - 컴퓨터는 가장자리나 곡선과 같은 저 수준의 형상을 찾고 일련의 convolutional layer를 통해 보다 추상적인 개념을 구성함으로써 이미지 분류를 수행

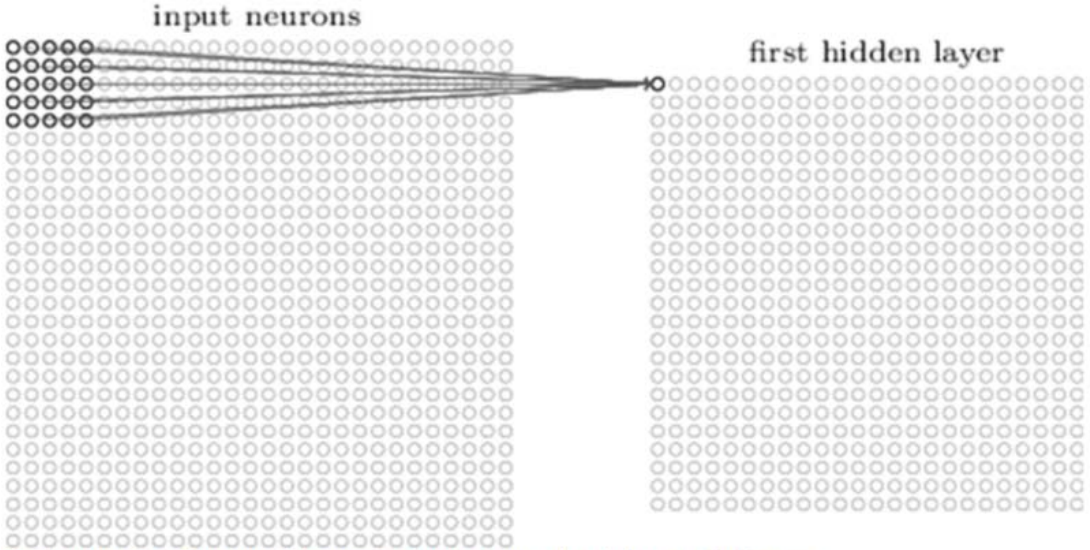
- **CNN 개요**
  - 일련의 convolutional, nonlinear, pooling (down sampling), 연결된 계층들(connected layers)로 이미지를 전달해서 출력을 구함
  - 출력은 단일 클래스이거나 그 이미지를 가장 잘 설명하는 클래스의 확률일 수 있다

합성곱 신경망의 계층

- **첫 번째 계층 (First layer)**
  - 항상 Convolutional Layer가 됨
  - 이전 그림의 예에서 입력은 32x32x3 픽셀 값에 대한 배열
- **필터 (Filter)**
  - 숫자(가중치 혹은 파라미터라 부름) 들의 배열
  - 때로는 뉴런(Neuron) 혹은 커널(Kernel)로 불린다
  - 필터의 깊이(depth)는 입력의 깊이와 같아야 한다
  - 필터는 입력 이미지 주위를 움직이면서 (**convolving**) 필터의 값과 이미지의 원래 픽셀 값을 곱하고, 곱한 값은 모두 더한다. (필터의 크기가 5x5 인 경우, 예에서는 전부 75번의 곱셈이 이루어짐)
  - 필터를 움직인 후에는 28 x 28 x 1 배열에 숫자를 가지게 되며, 이를 활성 맵 (**active map**) 혹은 형상 맵 (**feature map**) 이라 한다
    - ✓ 28 x 28 배열을 가지는 이유는 5 x 5 필터가 32 x 32 입력 이미지를 비추는데 784개의 다른 위치가 존재하기 때문
  - 5 x 5 x 3 필터를 두 개 사용한다면 출력 크기는 28 x 28 x 2 된다

합성곱 신경망의 계층

- **수용 필드 (Receptive field)**
  - 필터와 만나는 원본 이미지 영역
- **첫 번째 계층 (First layer)**



Visualization of 5 x 5 filter convolving around an input volume and producing an activation map

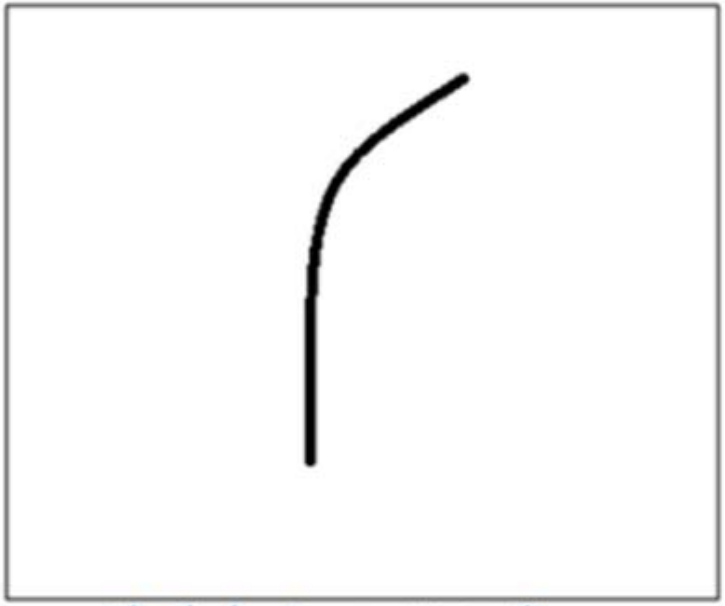
  - 각각의 필터는 필터 식별자 (filter identifier)로, 형상(features)은 직선 가장자리, 단순한 색상, 곡선 같은 것들을 의미

합성곱 신경망의 계층

- **첫 번째 계층 (First layer)**
  - 아래 7x7x3 의 곡선 검출 필터인 경우 곡선 모양인 영역을 따라 더 큰 숫자 값의 픽셀 구조를 가짐

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

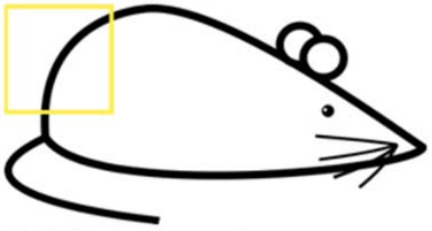


Visualization of a curve detector filter

합성곱 신경망의 계층

- **첫 번째 계층 (First layer)**
  - 예제 이미지를 가져와 필터를 왼쪽 상단에 위치시켰을 때
    - ✓ 필터가 표현하는 곡선과 닮은 모양이 있다면 모든 곱셈을 합했을 때 큰 값이 된다

Original image



Visualization of the filter on the image

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

합성곱 신경망의 계층

- **첫 번째 계층 (First layer)**
  - 합성곱이 크면 입력 볼륨에 필터를 활성화하도록 하는 곡선이 있을 가능이 높다는 의미
  - 아래 그림처럼 값이 작은 것은 곡선 검출 필터에 응답하는 것이 없기 때문이다

Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

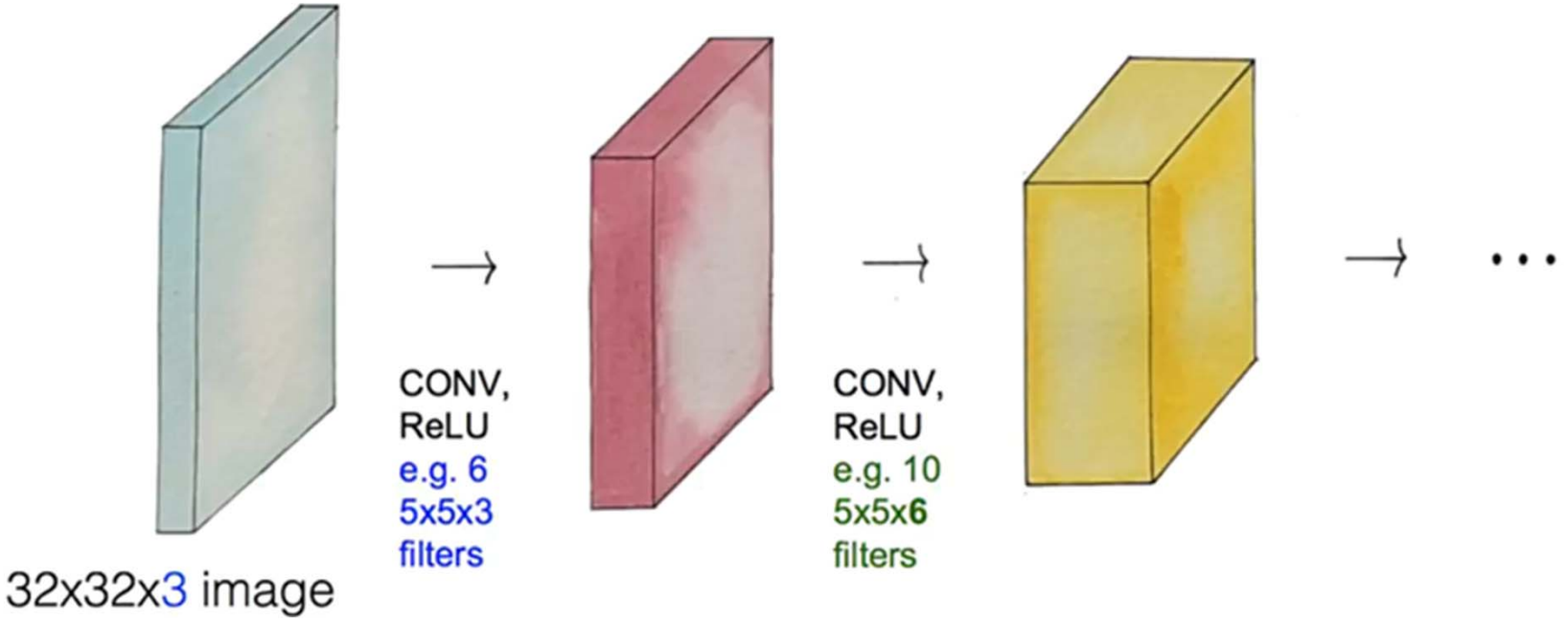
Pixel representation of filter

Multiplication and Summation = 0

  - 필터가 많아질수록 활성화 맵의 깊이가 커지고 더 많은 정보를 갖게 된다

합성곱 신경망의 계층

- **Convolution layers**
  - Weight 변수의 크기와 설정



합성곱 신경망의 계층

- **네트워크를 통한 진행**
  - Convolutional 계층들 사이에 배치되는 다른 계층들이 존재한다
  - 두 번째 conv 계층의 입력은 첫 번째 계층의 결과로 생긴 활성화 맵이다
  - 각 계층의 입력은 기본적으로 원본 이미지에서 낮은 레벨의 특정한 형상이 나타나는 위치를 기술한다
  - 필터들을 적용하면 (두 번째 계층으로 전달하면) 출력은 더 높은 수준의 형상들을 표현하게 된다
  - 고전적인 CNN 구조는 아래와 유사한 형태를 가진다.

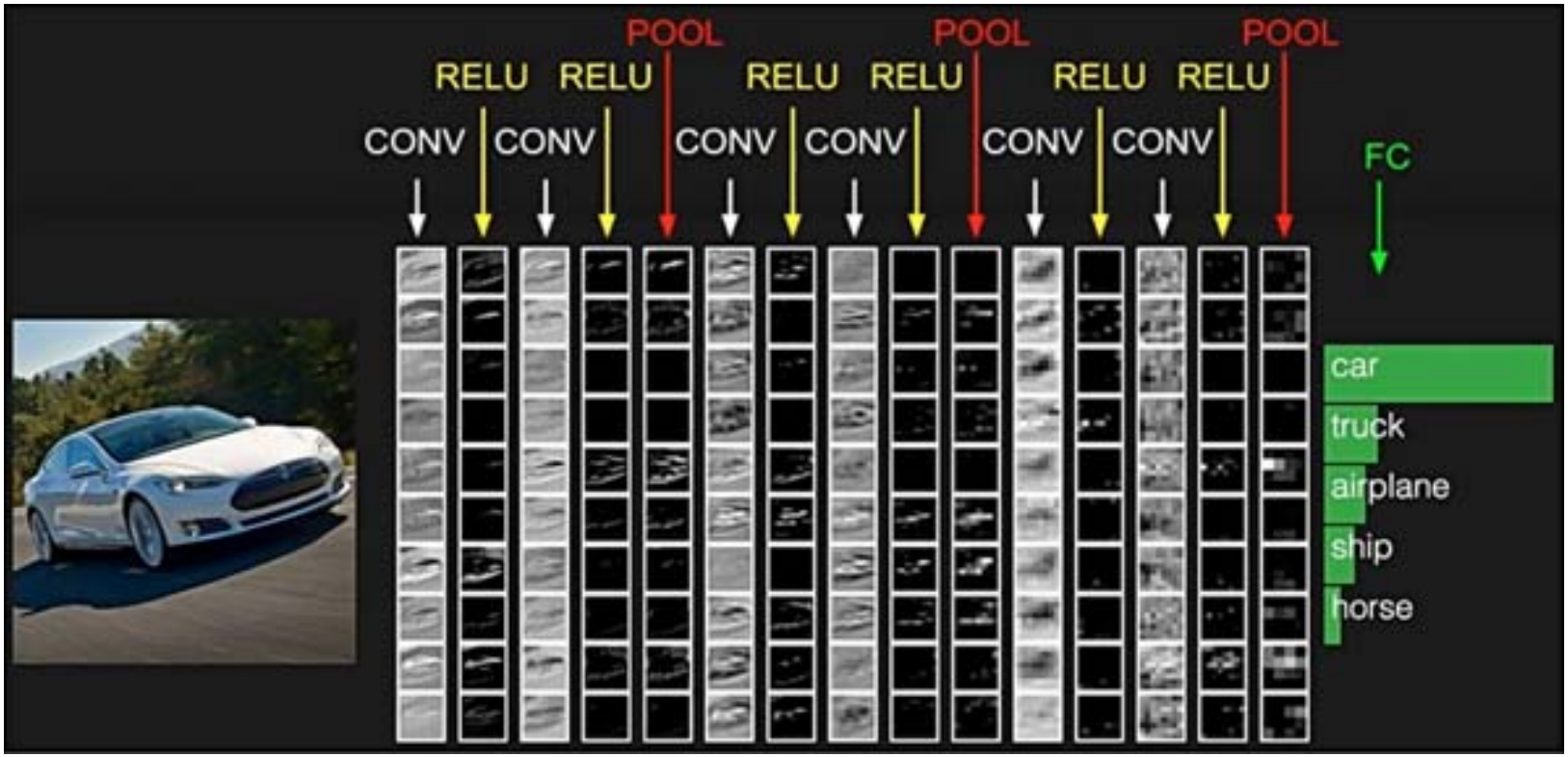
Input -> Conv -> ReLU -> Conv -> ReLU -> Pool -> ReLU -> Conv -> ReLU -> Pool -> Fully Connected

합성곱 신경망의 계층

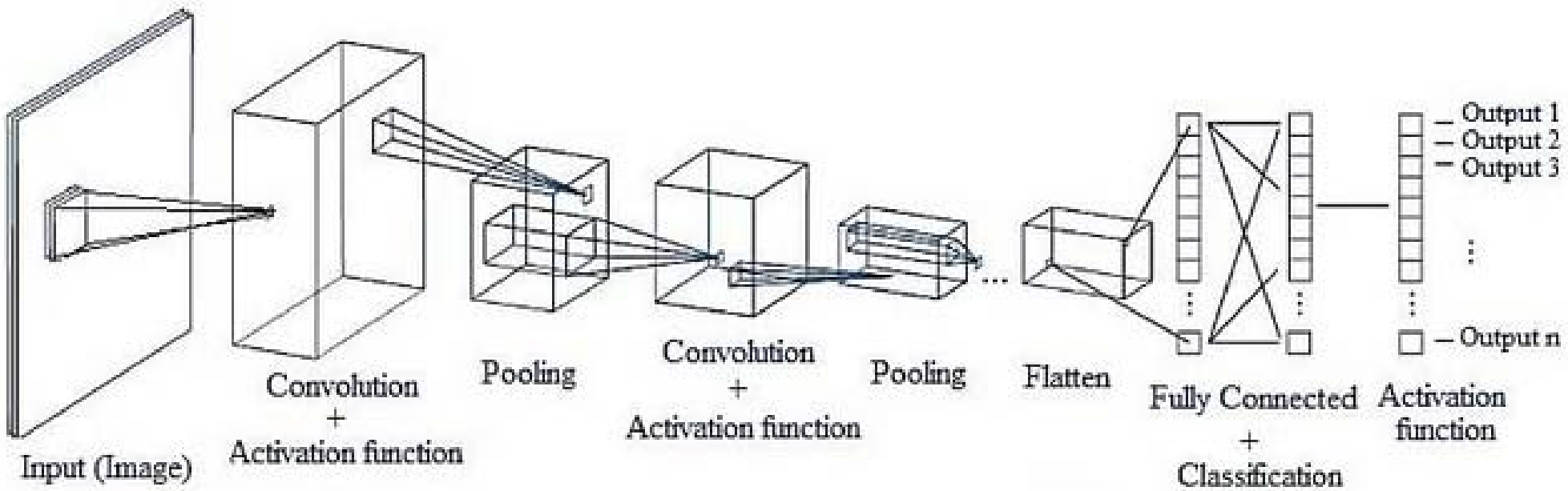
- **Fully Connected Layer**
  - 네트워크의 끝에 전체가 연결된 계층 추가
  - 입력 볼륨 (conv 혹은 ReLU 혹은 pool 계층의 출력인지 관계없이)을 가지고 N 변위의 벡터를 출력
    - ✓ 여기서 N은 프로그램이 선택해야 하는 클래스의 수
    - ✓ 예를 들어, 10진수를 분류하는 프로그램을 원한다면 N은 10이 된다
  - 전체가 연결된 계층이 동작하는 방식은 바로 직전 계층의 출력을 보는 것
  - 높은 수준의 형상들이 특정 클래스와 가장 깊은 상관관계가 있는지 살펴보고 특정한 가중치를 가지게 됨
  - 가중치와 이전 계층 간의 결과물을 계산하여 클래스의 정확한 확률 (가능성)을 구할 수 있다

합성곱 신경망의 계층

- **Fully Connected Layer**
  - 보통의 신경망처럼 전체 입력 볼륨과 연결되는 뉴런을 포함



합성곱 신경망의 계층


Input (Image)
Convolution
+
Activation function
Pooling
Convolution
+
Activation function
Pooling
Flatten
Fully Connected
+
Classification
Activation
function
Output 1
Output 2
Output 3
Output n

학습 과정

- **오차 역전파(Backpropagation)**
  - 필터 값(혹은 가중치)은 오차 역전파라 부르는 학습 과정을 통해서 이루어진다
  - **Forward pass -> Loss function -> Backward pass -> Weight update** 4개의 섹션으로 분류
- **Forward pass**
  - 입력 이미지를 (이전 예에서, 32x32x3 숫자 배열) 전체 네트워크를 통해 전달
  - 가중치 혹은 필터 값은 무작위로 초기화
- **Loss function**
  - 여러가지 방법이 있지만 일반적인 것은 MSE(mean squared error)

$$E_{total} = \sum \frac{1}{2}(\text{실제 값} - \text{예측값})^2$$

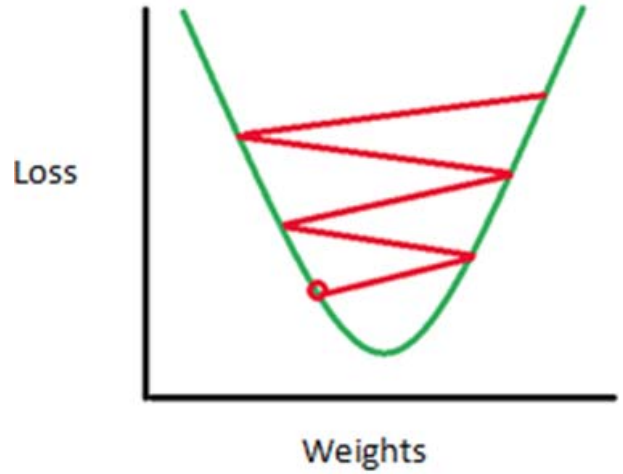
  - 처음 손실 값은 매우 클 것이지만 최종 목표는 학습 데이터의 라벨과 같은 지점에 도달하는 것이다

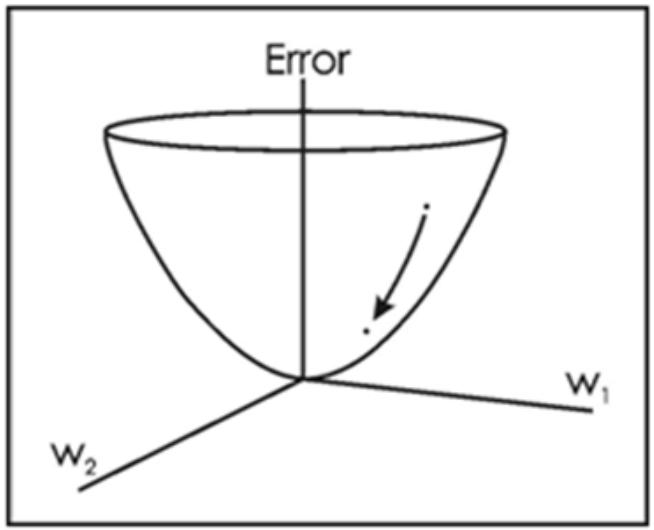
학습

- **Loss function**
  - 미적분학의 최적화 문제로 시각화
    - ✓ 신경망의 가중치가 독립 변수이고, 종속 변수가 손실인 3차원 그래프
- **Backward pass**
  - 손실의 최소화를 위해 손실에 기여가 가장 큰 가중치를 결정
    - ✓ 이는 가중치로 손실을 미분한 `**dL/dW**`의 수학적 등가물
- **Weight update**

$$w = w_i - \eta \frac{dL}{dW}$$

ω : 조정 가중치
ω_i : 직전 가중치
h : 학습률

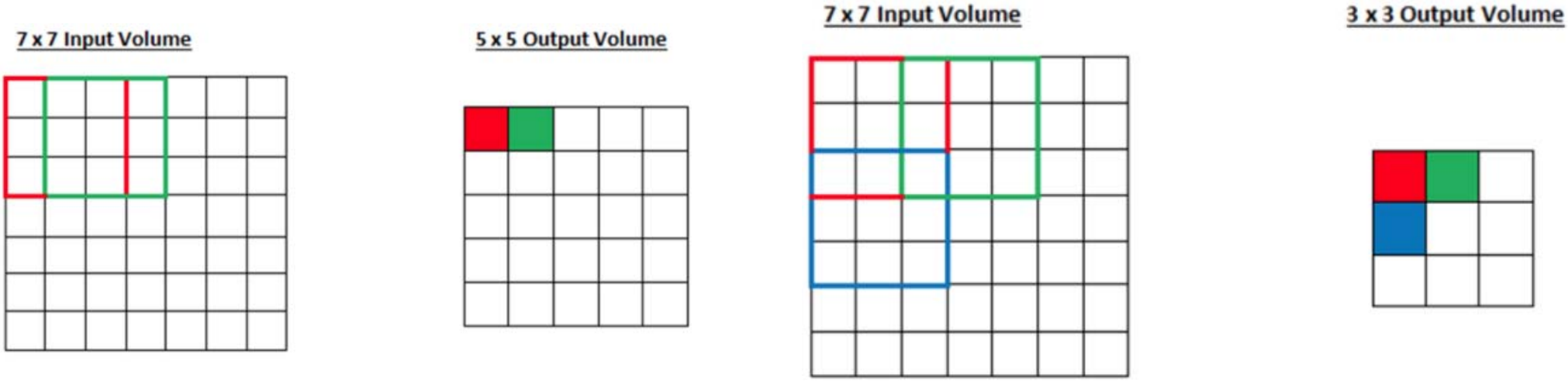




$$E_{total} = \sum \frac{1}{2}(\text{실제 값} - \text{예측값})^2$$

Convolutional Neural Network 세부 사항

- **Stride**
  - 필터가 입력 볼륨을 어떻게 convolve 할 지를 제어
    - ✓ Stirde를 증가하면 수용필드를 더 적게 오버랩하고, 더 작은 출력 볼륨을 만든다
    - ✓ 7x7 입력 볼륨에 대해 3x3 필터를 stride 1로 적용했을 때와 stride 2를 적용 했을 때,

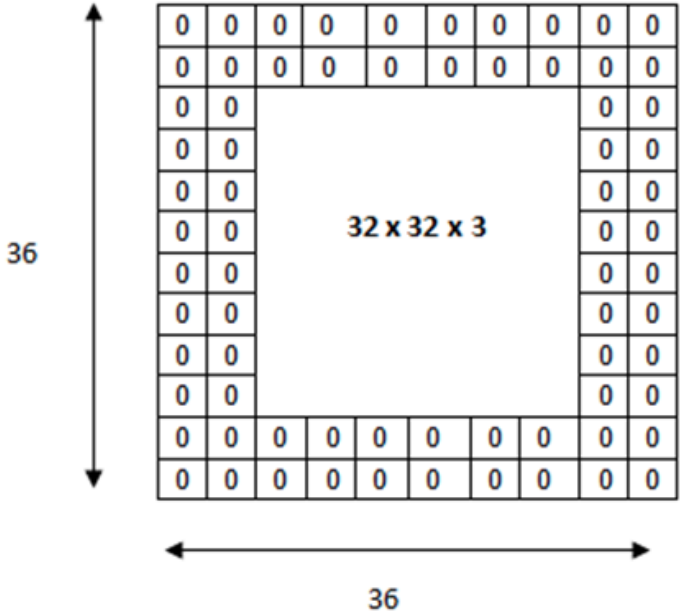


Stride 1 적용

Stride 2 적용

Convolutional Neural Network 세부 사항

- **Padding**
  - Convolution 계층이 늘어나면 출력 볼륨의 크기가 빠르게 감소한다
  - 출력 볼륨의 크기를 유지하고 싶을 때는 **padding**을 추가한다.
    - ✓ 5x5x3 필터를 사용하고, 32x32x3 입력 볼륨에 2개의 **zero padding**을 적용했을 때,
    - ✓ 이 경우, 출력 볼륨의 크기에 변화가 없다



Convolutional Neural Network 세부 사항

- **Stride와 Padding**
  - Stride가 1일 때 입력과 출력 볼륨을 같게 할 경우 적용할 padding의 수 (K는 필터의 크기)

$$ZeroPadding = \frac{(K-1)}{2}$$

  - Conv 계층의 출력 크기 계산 공식
    - ✓ O는 출력 높이/길이를, W는 입력 높이/길이를, K는 필터 크기를 , P는 패딩을, S는 Stride를 의미

$$O = \frac{(W-K+2P)}{S} + 1$$

하이퍼 파라미터 (Hyper-parameters)

- **의미**
  - 사용자의 직관과 경험 등을 토대로 직접 지정하는 파라미터 값
  - 정해진 표준이 없으며, 네트워크는 사용자가 가진 데이터 형식에 의존
  - 데이터 크기, 이미지의 복잡성, 이미지 처리 작업의 유형 등에 따라 변한다

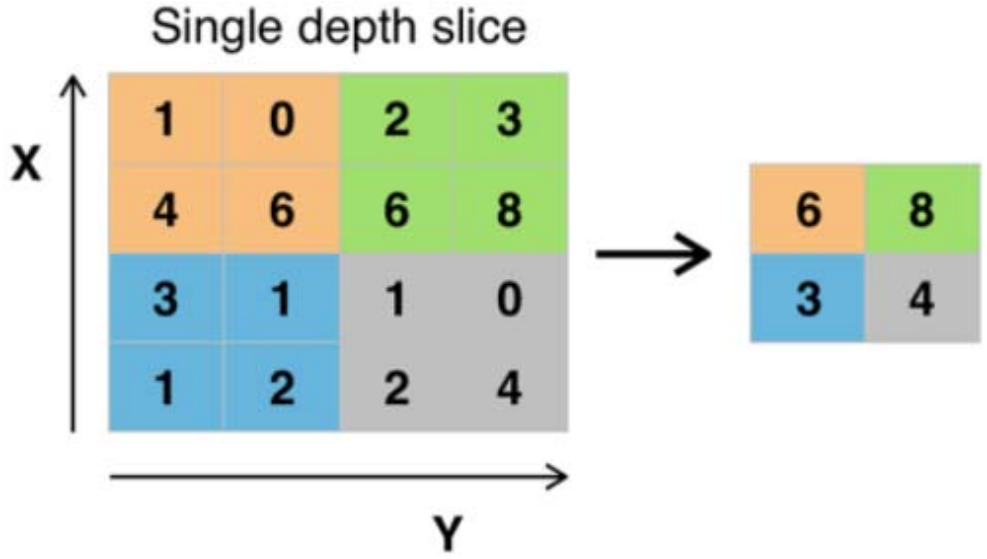
- **종류**
  - 사용할 계층 수, Conv 계층 수, 필터 크기, Stride와 Padding에 대한 값 등

ReLU (Rectified Linear Units) 계층

- **활성(Activation) 계층**
  - Conv 계층 다음에 비선형(Non-linear) 계층을 즉시 추가하는 것이 관례
  - 선형 동작을 계산(곱셈과 덧셈) 하고, 시스템에 비선형성 (Nonlinearity)을 도입
- **ReLU의 장점**
  - 과거 tanh와 sigmoid 같은 비선형 함수들이 유행했으나 최근에는 정확도에 큰 차이 없이 네트워크가 훨씬 빨리 학습할 수 있는 (계산 효율 때문에) ReLU 가 많이 사용
  - 경사(gradient)가 계층을 지나면 속도가 기하급수적으로 감소하기 때문에 네트워크의 하위 계층들이 매우 느리게 진행(또는 학습)되는 문제 (vanishing gradient problem)를 완화
- **동작 방식**
  - 입력 볼륨 모든 값에 $f(x) = \max(0, x)$ 함수를 적용
    - ✓ Negative activatio을 모두 0으로 변경
    - ✓ Conv 계층의 수용 필드에 영향을 미치지 않고 모델과 전체 네트워크에 비선형 속성을 증가

풀링 (Polling) 계층

- **개요**
  - 일부 ReLU 계층 이후에 풀링(pooling) 계층이 적용 가능하며 downsampling 계층이라고도 한다
  - average pooling, L2-norm pooling 등 몇 가지 계층 옵션들 중에 **Maxpooling**이 가장 인기가 많다
    - ✓ 기본적으로 필터 (보통 2 x 2의 크기)와 같은 길이의 Stride를 입력 볼륨에 적용
    - ✓ convolve 하는 모든 하위 영역에서 최대 값을 출력



풀링 (Polling) 계층

- **적용 이유**
  - 큰 활성화 값을 갖는 특정 형상이 원본 이미지에 있다는 것을 알면 이에 대한 정확한 위치가 다른 형상들에 대한 상대적 위치만큼 중요한 것은 아니다
  - 입력 볼륨의 공간적인 치수를 획기적으로 감소
    - ✓ 길이와 너비가 변하지만 깊이는 변하지 않는다

- **장점**
  - 파라미터 혹은 가중치의 양이 75% 정도 줄어 계산 비용이 절감
  - 오버피팅(Overfitting)을 제어

드롭아웃 계층 (Dropout Layers)

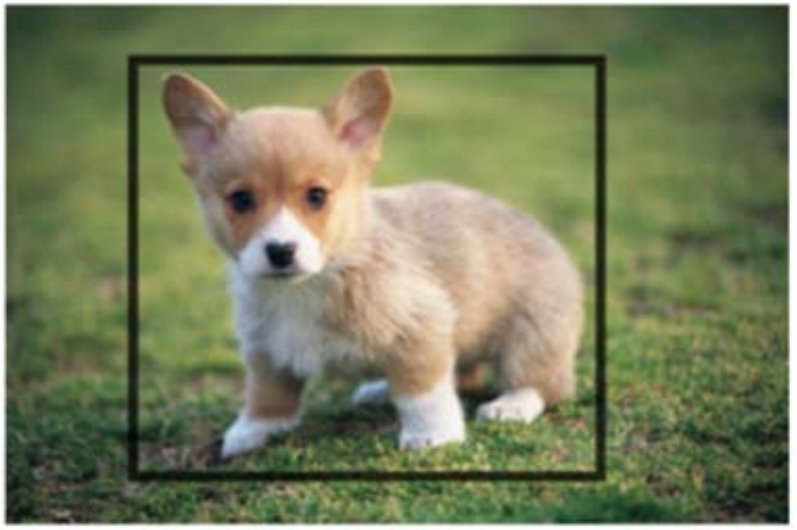
- **네트워크의 가중치가 학습 예제에 지나치게 최적화 (overftting) 되었을 때**
  - 새로운 예제에 대해 네트워크가 잘 동작하지 않는다

- **적용 방법**
  - 임의의 활성 군집을 0으로 설정하여 드롭아웃(**dropout)** 한다
    - ✓ 네트워크가 학습 데이터에 지나치게 적합(fitting) 해지는 것을 방지

- **주의**
  - 활성화의 일부가 누락되어도 특정 예제에 대해서 올바른 분류나 출력을 제공해야 한다
  - 학습 시에만 이 계층을 사용하고 테스트에는 사용하지 않는다.

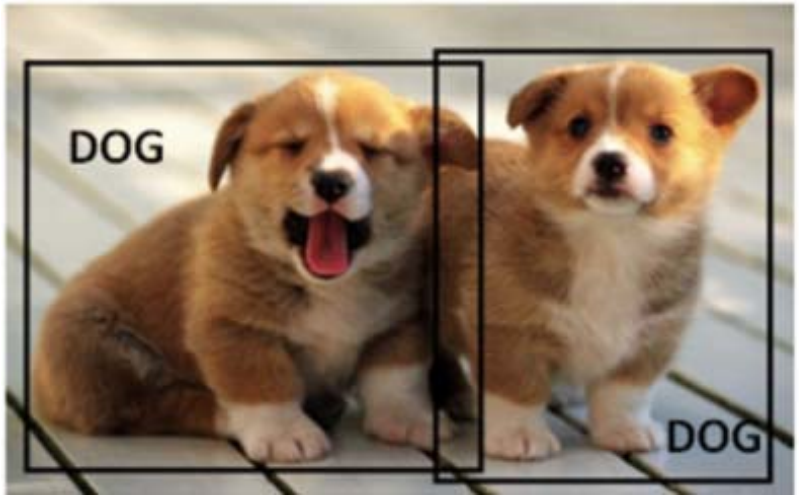
Classification, Localization, Detection, Segmentation

- **이미지 분류 (Image Classification)**
  - 입력 이미지를 가져와서 범주 별로 클래스 번호를 출력하는 과정
- **Object Localization**
  - 입력 이미지에서 사물에 대한 경계 박스를 만든다



Classification, Localization, Detection, Segmentation

- **Object Detection**
  - Object Localization이 필요한 Object Detection 작업 수행
- **Object Segmentation**
  - 사물의 클래스 라벨과 외곽선을 출력



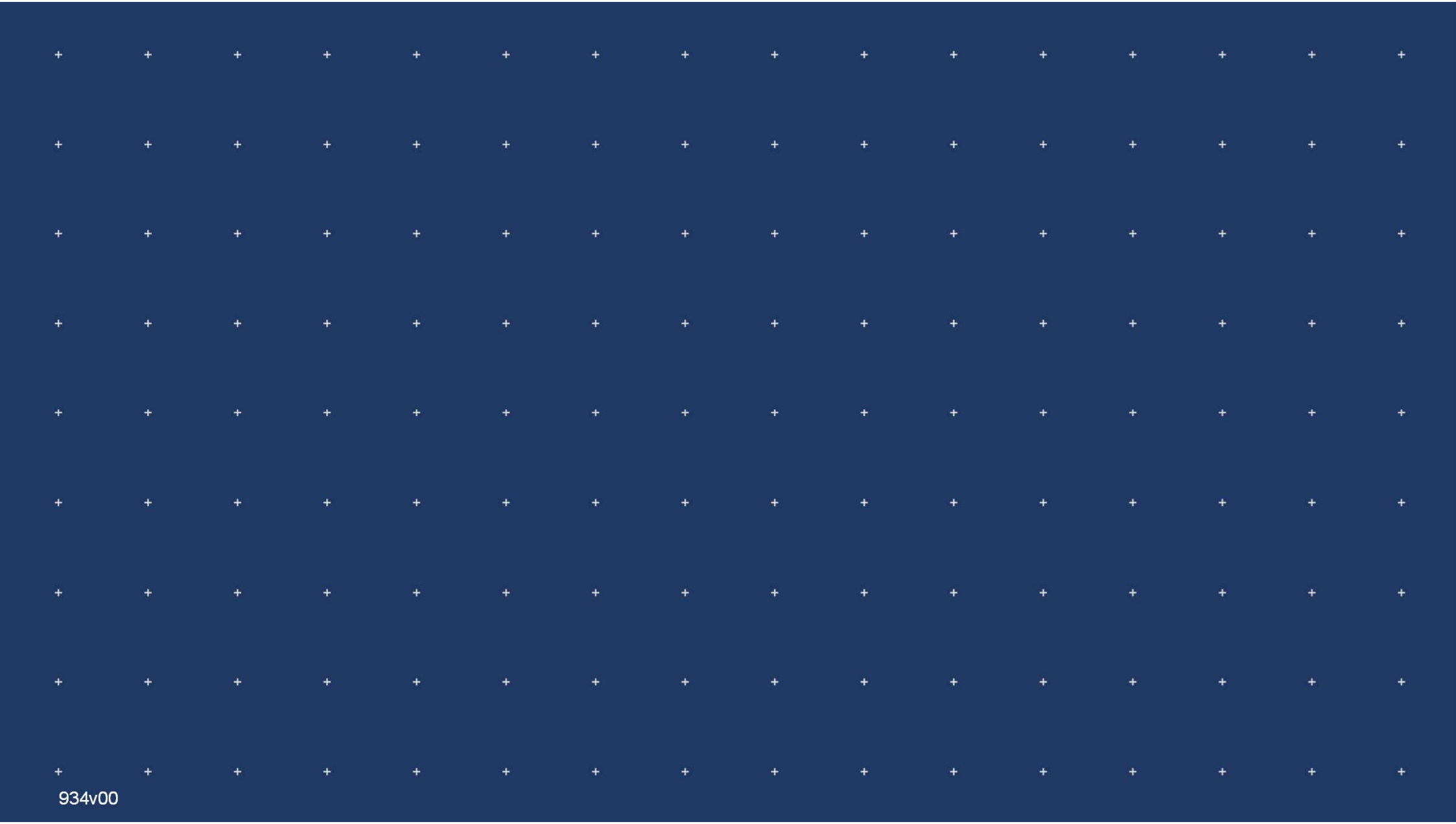
학습 전달 (Transfer Learning)

- **개요**
  - 네트워크에서 필요한 데이터의 수요를 줄여준다
  - 대규모 데이터로 학습된 모델을 (학습된 모델의 가중치와 파라미터) 가져와서 자신의 데이터 셋으로 미세 조정
  - 미리 학습된 모델을 형상 추출기로 동작하게 하는 것

- **방법**
  - 네트워크의 마지막 계층을 제거하고 문제가 되는 공간이 무엇인지에 따라 자신의 분류기로 대체
  - 다른 모든 계층의 가중치를 고정 (freezing)하고 네트워크를 정상적으로 학습
    - ✓ 고정(freezing) 한다는 것은 gradient descent/optimization 중에는 가중치를 변경하지 않는다는 의미

데이터 증강 (Data Augmentation) 기법

- **의미**
  - 라벨을 동일하게 유지하면서 배열을 변경하는 방식으로 학습 데이터를 변경하는 접근법
  - 사용자의 데이터 셋을 인의적으로 확장하는 방법
  - **grayscales, horizontal flips, vertical flips, random crops, color jitters, translations, rotations** 등


934v00