인공지능의 이해

Understanding Artificial Intelligence

유준수 (Joon Soo Yoo, Ph.D.)

Fall 2026

Defense & Advanced Security Lab (DAS Lab)
Department of Advanced Defense Technology and Industry
Jeonbuk National University

Deep Learning and Computer Vision

From MLP to CNN

Week 07

1. From Week 06 to Week 07

지난 Week에는 MNIST 숫자 이미지를 MLP로 분류했다.

$$28 \times 28 \text{ image} \rightarrow 784 \text{ numbers} \rightarrow 128 \text{ hidden units} \rightarrow 10 \text{ classes}$$

- MNIST 이미지는 원래 28×28 형태의 2D 이미지이다.
- MLP에 넣기 위해 이미지를 한 줄의 벡터로 펼쳤다.
- 이 과정을 Flatten이라고 한다.

$$X \in \mathbb{R}^{28 \times 28} \quad \rightarrow \quad \mathbf{x} \in \mathbb{R}^{784}$$

생각해볼 점

Flatten을 하면 Pixel 값 자체는 사라지지 않는다. 하지만 위, 아래, 왼쪽, 오른쪽처럼 이미지 안에 있던 공간적 관계는 Model 구조에 직접 반영되지 않는다.

1-1. Why Do We Need a Validation Set?

지난 Week에서는 Training Data로 MLP를 학습하고, Test Data로 최종 성능을 평가했다.

$$\textit{Training Data} \rightarrow \textit{Model Training}$$

$$\textit{Test Data} \rightarrow \textit{Final Evaluation}$$

그런데 Model을 만들 때는 여러 선택이 필요하다.

- Hidden Neuron을 32개, 64개, 128개 중 몇 개 사용할까?
- Hidden Layer는 몇 개 사용할까?
- Epoch는 5, 10, 20 중 몇 번으로 설정할까?

생각해볼 점

Test Accuracy를 계속 확인하면서 가장 좋은 Model을 선택한다면, Test Data가 더 이상 순수한 최종 평가용 Data라고 할 수 있을까?

1-2. Training vs. Validation vs. Test

Model을 학습하고 선택하고 평가하기 위해 Data를 세 부분으로 구분할 수 있다.

Dataset	Purpose
Training Set	Weight와 Bias를 학습한다.
Validation Set	Hyperparameter를 조정하고 Model을 선택한다.
Test Set	선택이 끝난 Model의 최종 성능을 평가한다.

예를 들어 MNIST의 기존 Training Data 60,000개 중 10%를 Validation으로 사용하면

$$54{,}000\ (\textit{Train}) + 6{,}000\ (\textit{Validation})$$

별도의 Test Data 10,000개는 최종 평가에 사용한다.

핵심

Training은 Parameter를 학습하고, Validation은 Model을 선택하며, Test는 최종 성능을 평가한다.

1-3. Validation and Overfitting

Training 과정에서 Loss가 계속 감소한다고 해서 Model의 일반화 성능이 항상 좋아지는 것은 아니다.

Epoch	Training Loss	Validation Loss
1	0.80	0.85
3	0.50	0.55
5	0.30	0.40
7	0.20	0.48
10	0.10	0.65

- Training Loss는 계속 감소한다.
- Validation Loss는 Epoch 5 이후 증가한다.
- 이는 Overfitting의 징후일 수 있다.

생각해볼 점

Epoch 10의 Training Loss가 가장 낮다. 그렇다면 Epoch 10의 Model이 가장 좋은 Model일까?

2. What Is Deep Learning?

Deep Learning은 여러 Layer를 사용하여 Input을 단계적으로 변환하는 방법이다.

$$\mathbf{x} \to \mathbf{h}^{(1)} \to \mathbf{h}^{(2)} \to \mathbf{h}^{(3)} \to \hat{\mathbf{y}}$$

여기서 각 Hidden Layer의 Output은 새로운 Representation이다.

$$\mathbf{h}^{(1)} = f_1(\mathbf{x}), \qquad \mathbf{h}^{(2)} = f_2(\mathbf{h}^{(1)}), \qquad \mathbf{h}^{(3)} = f_3(\mathbf{h}^{(2)})$$

- 첫 번째 Layer는 Raw Input을 단순한 특징 표현으로 바꾼다.
- 다음 Layer는 이전 특징들을 다시 조합하여 더 다른 표현을 만든다.
- 마지막 Layer는 그 표현을 이용해 Class나 값을 예측한다.

핵심

Deep Learning은 사람이 Feature를 모두 직접 설계하기보다, Model이 여러 Layer를 통해 유용한 Representation을 학습하도록 만든다.

2-2. Why Go Deeper?

Layer가 깊어지면 여러 단계의 변환을 만들 수 있다.

$$\mathbf{x} \to \mathbf{h}^{(1)} \to \mathbf{h}^{(2)} \to \mathbf{h}^{(3)} \to \hat{\mathbf{y}}$$

Image를 예로 들면 다음과 같이 이해할 수 있다.

$$\textit{Pixels} \to \textit{Local Patterns} \to \textit{Combined Patterns} \to \textit{Class Prediction}$$

- 첫 단계에서는 밝기 변화, 작은 선, 간단한 질감 같은 정보를 표현할 수 있다.
- 다음 단계에서는 여러 Local Pattern을 조합하여 더 큰 형태를 표현할 수 있다.
- 마지막에는 이렇게 만들어진 Representation을 이용해 Class를 예측한다.

핵심

Depth는 여러 번의 변환을 가능하게 하며, 단순한 입력을 점차 분류에 유용한 표현으로 바꾸는 데 도움을 줄 수 있다.

2-2. Depth vs. Width

Width

한 Layer 안의 Neuron 수를 늘린다.

$$784 \to 256 \to 10$$

Depth

Hidden Layer의 개수를 늘린다.

$$784 \to 128 \to 64 \to 10$$

핵심

Depth와 Width는 서로 다른 Architecture Hyperparameter이다. 어느 쪽도 크다고 항상 좋은 것은 아니다.

2-3. How Do We Choose Depth and Width?

Depth와 Width를 정하는 완벽한 공식은 없다.

보통은 작은 Model에서 시작하여 점진적으로 조정한다.

$$\textit{Small Model} \rightarrow \textit{Larger Model} \rightarrow \textit{Validation Performance Check}$$

예를 들어 MLP에서는 다음과 같이 Width를 바꿔볼 수 있다.

$$784 \rightarrow 64 \rightarrow 10$$

$$784 \rightarrow 128 \rightarrow 10$$

$$784 \rightarrow 256 \rightarrow 10$$

- 성능이 부족하면 Neuron 수나 Layer 수를 늘려본다.
- 성능이 더 이상 좋아지지 않으면 Model이 충분히 클 수 있다.
- Training 성능은 좋은데 Test 성능이 나쁘면 Overfitting을 의심한다.

핵심

Architecture는 보통 Validation 성능, 계산 비용, Overfitting 여부를 보며 실험적으로 결정한다.

3. Review: MNIST with MLP

MNIST Image는 28×28 Pixel이다.

$$\mathbf{x} = [x_1, x_2, \ldots, x_{784}]^T$$

Dense Neuron 하나는 모든 Input을 사용한다.

$$h_i = \mathrm{ReLU}(\mathbf{w}_i^T \mathbf{x} + b_i)$$

핵심

MLP의 Dense Layer에서는 각 Neuron이 Input 전체와 연결된다.

3-1. Images Have Spatial Structure

Image는 단순한 숫자 목록이 아니라, 2D 공간 위에 배치된 Pixel들의 배열이다.

$$X \in \mathbb{R}^{28\times 28}$$

예를 들어 숫자 7의 일부를 생각해보자.

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

이 배열에서 중요한 것은 Pixel 값 하나하나만이 아니다.

- 인접한 Pixel들이 모이면 선이 된다.
- 선들이 모이면 모서리나 곡선이 된다.
- 모서리와 곡선이 모이면 숫자나 객체의 형태가 된다.

핵심

Image에서는 Pixel의 값뿐 아니라, Pixel들이 어디에 있고 서로 어떻게 이웃하는지가 중요하다.

3-2. What Does Flatten Change?

원래 2D Image가 다음과 같다고 하자.

$$X = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

Flatten 후에는

$$\mathbf{x} = [1, 1, 0, 0, 1, 0, 0, 1, 1]^T$$

가 된다.

- Pixel 값과 개수는 그대로 유지된다.
- 하지만 Dense Layer에는 인접한 Pixel을 우선 처리하는 구조가 없다.

핵심

Flatten은 정보를 삭제하지 않는다. CNN은 2D Spatial Structure를 활용하도록 설계된 Architecture이다.

4. Introducing CNN

CNN = Convolutional Neural Network

Input Image → Convolution → Feature Maps → Classification

핵심

CNN은 Image의 Local Pattern과 Spatial Structure를 활용한다.

4-1. What Is a Filter (Kernel)?

예를 들어 3×3 Filter는 다음과 같다.

$$K = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

- Filter는 작은 Weight Matrix이다.
- Image의 Local Patch에 적용한다.
- 같은 Filter를 다른 위치에서도 사용한다.

참고

실제 CNN에서 Filter Weight는 Training으로 학습될 수 있다.

4-2. What Does a Filter Detect?

다음과 같은 Filter를 생각해보자.

$$K = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

왼쪽과 오른쪽 Pixel 값의 차이를 이용하므로, 특정 방향의 밝기 변화에 반응할 수 있다.

- Filter의 값은 어떤 Pattern에 반응하는지를 결정한다.
- 학습된 Filter가 반드시 사람이 붙인 이름 하나에 대응하는 것은 아니다.
- 여러 Filter는 서로 다른 Pattern에 반응하도록 학습될 수 있다.

질문

Image의 어느 위치에서 같은 Pattern이 나타나도 같은 Filter로 탐지할 수 있을까?

5. Convolution: One Calculation

$$X_{patch} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad K = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

대응하는 원소끼리 곱하여 더한다.

$$z = (1+0+0) + (1+0+0) + (0+0-1) = 1$$

핵심

Local Patch 하나와 Filter 하나로 Output Value 하나를 계산한다.

5-1. Sliding the Filter

Filter를 Image 위에서 이동시키며 같은 계산을 반복한다.

$$\text{Patch } 1 \rightarrow \text{Output } 1$$

$$\text{Patch } 2 \rightarrow \text{Output } 2$$

$$\vdots$$

$$\text{Multiple Outputs} \rightarrow \text{Feature Map}$$

핵심

하나의 Filter를 여러 위치에서 반복해서 사용한다.

5-2. Interactive Demo: Convolution

Open Interactive Visualizer

Convolution and Pooling Tutorial

1. Filter의 첫 위치를 확인한다.
2. Filter가 이동하면 Input Patch가 어떻게 변하는지 본다.
3. 각 위치에서 Output이 어떻게 계산되는지 관찰한다.

질문

Filter가 이동할 때 Weight 값도 바뀌는가?

5-3. Stride: How Far Does the Filter Move?

Stride는 Filter를 한 번에 이동시키는 칸 수이다.

Stride 1	한 칸씩 이동, 더 많은 Output 위치
Stride 2	두 칸씩 이동, 더 적은 Output 위치

예: Input 5, Kernel 3, Padding 0일 때

$$S = 1: \quad 3 \text{ positions}, \qquad S = 2: \quad 2 \text{ positions}$$

핵심

Stride가 커지면 보통 Feature Map의 Spatial Size가 작아진다.

6. What Is a Feature Map?

$$\textit{Image} \xrightarrow{\textit{Filter } K} \textit{Feature Map}$$

- Feature Map은 Filter가 여러 위치에서 만든 Output들의 배열이다.
- 다른 Filter는 다른 Feature Map을 만든다.
- 32개의 Filter는 일반적으로 32개의 Output Channel을 만든다.

핵심

Feature Map은 Filter가 어느 위치에서 강하게 반응하는지 보여준다.

6-1. Local Connectivity and Weight Sharing

Concept	Meaning
Local Connectivity	작은 Image 영역을 먼저 처리
Weight Sharing	같은 Filter를 여러 위치에 사용

핵심

CNN은 Local Connectivity와 Weight Sharing을 이용한다.

6-2. Comparing Parameter Counts

784 Input, 128 Neuron을 가진 Dense Layer:

$$784 \times 128 + 128 = 100{,}480$$

Grayscale Image에 3×3 Filter 32개를 사용한 Conv Layer:

$$(3 \times 3 \times 1 + 1) \times 32 = 320$$

주의

서로 다른 Layer의 예시다. 전체 CNN이 항상 더 작거나 성능이 높은 것은 아니다.

7. ReLU After Convolution

CNN Explainer — Interactive Visualization

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

$$H = \mathrm{ReLU}(\mathrm{Conv}(X, K) + b)$$

$$\begin{bmatrix} -2 & 3 & 1 \\ 4 & -1 & 2 \\ 0 & 5 & -3 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 3 & 1 \\ 4 & 0 & 2 \\ 0 & 5 & 0 \end{bmatrix}$$

핵심

CNN의 ReLU도 Week 06에서 배운 Nonlinear Activation과 같다.

7-1. Why Show ReLU in CNN Explainer?

Convolution의 출력에는 양수와 음수가 모두 나올 수 있다.

$$Z = K * X + b$$

ReLU는 음수 값을 0으로 바꾼다.

$$A = \mathrm{ReLU}(Z) = \max(0, Z)$$

- Convolution은 Local Patch에 대한 Filter Response를 계산한다.
- ReLU는 음수 Response를 제거하고 Nonlinearity를 추가한다.
- CNN Explainer는 두 단계를 따로 보여주어 값의 변화를 관찰하게 한다.

핵심

ReLU 이후의 Feature Map은 Filter Response 중 양수 Activation만 남긴 결과이다.

7-2. Interpreting Red and Blue Colors

CNN Explainer의 색은 Activation 값을 보기 쉽게 표현한 것이다.

Color	Meaning
Blue	Positive activation
White or pale color	Near zero activation
Red	Negative activation before ReLU

ReLU를 지나면 음수 Activation은 0이 된다.

$$\textit{Red values} \rightarrow 0$$

주의

빨간색은 “나쁜 특징”이 아니다. 해당 Filter의 선형 응답이 음수였다는 뜻이다.

8. What Is Max Pooling?

2×2 영역에서 가장 큰 값을 선택한다.

$$\begin{bmatrix} 1 & 3 \\ 4 & 2 \end{bmatrix} \xrightarrow{\textit{Max Pooling}} 4$$

- Feature Map의 Spatial Size를 줄인다.
- Local Region의 강한 Activation을 남긴다.
- Max Pooling 자체에는 학습되는 Weight가 없다.

핵심

Max Pooling은 대표적인 Downsampling Operation이다.

8-1. Max Pooling: Full Example

2×2 Window, Stride 2를 사용하자.

$$\begin{bmatrix} 1 & 3 & 2 & 0 \\ 4 & 2 & 1 & 5 \\ 0 & 1 & 6 & 2 \\ 3 & 2 & 4 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 4 & 5 \\ 3 & 6 \end{bmatrix}$$

$$4 \times 4 \to 2 \times 2$$

질문

Output의 4, 5, 3, 6은 각각 어떤 영역에서 나왔는가?

8-2. Interactive Demo: Max Pooling

Convolution and Pooling Tutorial

Open Interactive Tutorial

1. 2×2 Pooling Window를 확인한다.
2. Window 안의 최댓값을 찾는다.
3. Pooling 전후 Feature Map의 크기를 비교한다.

질문

Pooling에서 Weight를 Update할 필요가 있을까?

8-3. Why Pooling?

- 이후 Layer의 Computation을 줄일 수 있다.
- Local Feature Response를 요약한다.
- 작은 위치 변화에 덜 민감해질 수 있다.

하지만 일부 Spatial Information은 손실된다.

참고

현대 CNN에서는 Pooling 대신 Strided Convolution을 사용하기도 한다.

8-4. What Is Lost During Pooling?

2×2 영역에서 최댓값만 남기면 다른 세 값은 Output에 직접 남지 않는다.

$$\begin{bmatrix} 1 & 3 \\ 4 & 2 \end{bmatrix} \to 4, \qquad \begin{bmatrix} 0 & 0 \\ 4 & 0 \end{bmatrix} \to 4$$

서로 다른 두 영역이 같은 Output 4로 요약된다.

핵심

Pooling은 계산량을 줄이는 데 유용하지만 일부 위치와 세부 정보가 손실될 수 있다.

9. A Simple CNN for MNIST

Layer	Output Shape
Input	$28 \times 28 \times 1$
Conv2D (32, 3×3, valid)	$26 \times 26 \times 32$
MaxPooling2D (2×2)	$13 \times 13 \times 32$
Flatten	5408
Dense (10, Softmax)	10

핵심

Convolution과 Pooling으로 Feature를 처리한 뒤 Dense Layer로 분류한다.

9-1. Why 26 by 26? (Reference)

Input Size 28, Kernel Size 3, Padding 0, Stride 1일 때:

$$28 - 3 + 1 = 26$$

따라서 32개의 Filter를 적용하면

$$28 \times 28 \times 1 \rightarrow 26 \times 26 \times 32$$

가 된다.

참고

Output Shape는 Kernel Size, Stride, Padding에 따라 달라진다.

9-2. Why 13 by 13?

26×26 Feature Map에 2×2 Max Pooling (Stride 2)을 적용하면

$$26 \times 26 \times 32 \rightarrow 13 \times 13 \times 32$$

Flatten 후 값의 개수는

$$13 \times 13 \times 32 = 5408$$

이다.

핵심

Pooling은 Spatial Size를 줄이면서 Channel 수는 유지할 수 있다.

9-3. Why Flatten Again?

Convolution과 Pooling을 거친 결과는 여전히 3D Tensor이다.

$$13 \times 13 \times 32$$

이를 마지막 Dense Classifier에 전달하기 위해 한 줄로 펼칠 수 있다.

$$13 \times 13 \times 32 = 5408$$

$$5408 \xrightarrow{Dense(10)} 10$$

핵심

CNN은 앞부분에서 Spatial Feature를 추출하고, 뒤쪽에서 그 정보를 이용해 Class를 결정할 수 있다.

10. How Does CNN Learn?

CNN도 Week 06과 같은 Training Loop를 사용한다.

$$\textit{Forward} \rightarrow \textit{Loss} \rightarrow \textit{Backpropagation} \rightarrow \textit{Update}$$

Filter Weight도 학습 가능한 Parameter이다.

$$k_{11}^{(t+1)} = k_{11}^{(t)} - \eta \frac{\partial L}{\partial k_{11}}$$

핵심

CNN의 학습 원리는 MLP와 같다. 달라지는 것은 Layer의 계산 구조이다.

10-1. Which CNN Parameters Are Learned?

Component	Learned Parameters?
Conv2D Filter	Yes: Kernel Weights, Biases
ReLU	No
Max Pooling	No
Flatten	No
Dense Layer	Yes: Weights, Biases

핵심

Backpropagation은 Conv Filter와 Dense Layer의 Parameter에 대한 Gradient를 계산한다.

10-2. Total Parameters: MLP vs. CNN

Week 06 MLP의 Parameter 수:

$$784 \times 128 + 128 + 128 \times 10 + 10 = 101{,}770$$

이번 CNN 예제의 Parameter 수:

$$\underbrace{(3 \times 3 \times 1 + 1) \times 32}_{Conv2D=320} + \underbrace{(5408 \times 10 + 10)}_{Dense=54{,}090} = 54{,}410$$

참고

이 두 예시에서는 CNN이 더 적은 Parameter를 사용하지만, 모든 CNN이 항상 더 작은 것은 아니다.

11. MLP vs. CNN

	MLP	CNN
Input	Flattened Pixels	2D Image
Main Layer	Dense	Convolution
Connectivity	Fully Connected	Local
Weight Sharing	일반적으로 없음	있음
Training	Backpropagation	Backpropagation
Evaluation	Accuracy, F1 등	Accuracy, F1 등

핵심

CNN은 Image의 Spatial Structure를 Model Architecture에 반영한다.

11-1. Interactive CNN Explainer

CNN Explainer — Interactive Visualization

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

1. Input Image를 선택한다.
2. Convolution Layer와 Feature Map을 살펴본다.
3. Pooling Layer가 Output을 어떻게 줄이는지 확인한다.
4. 마지막 Classification Output을 확인한다.

핵심

한 번의 Forward Propagation을 여러 Layer에 걸쳐 관찰해보자.

11-2. CNN Explainer: What Should We Notice?

1. Input Image와 Feature Map의 Shape를 확인한다.
2. Convolution Filter가 Local Patch를 처리하는 과정을 살펴본다.
3. Pooling 전후 Spatial Size가 어떻게 변하는지 확인한다.
4. Output Layer에서 10개 Class의 Prediction을 확인한다.

질문

CNN의 어느 Layer가 Weight를 학습하고, 어느 Layer가 값만 변환하는가?

Thank You

Questions?