15주차

날짜	@2024년 6월 18일
과제	강의 요약 출석 퀴즈 캐글 필사
세부내용	[딥러닝 4단계] 1. 합성곱 신경망
자료	[Week15] 캐글필사 [Week15] 출석퀴즈

합성곱 신경망(CNN)

컴퓨터비전

- 컴퓨터비전에서 딥러닝에 관심을 갖는 이유
  - 컴퓨터 비전의 빠른 발전이 많은 새 애플리케이션이 만들어지게 함
  - 컴퓨터 비전 연구가 신경망 구조와 알고리즘에 서로 영감을 줌

    ex) 컴퓨터 비전 개념을 가지고 음성인식에 사용

Computer Vision Problems

- 이미지 분류(Image classification)
- 물체 감지(object detection) → 자율주행
- 신경망 스타일 변형(Neural Style Transfer)

  - 왼쪽 이미지를 오른쪽 스타일로 변형

Deep Learning on Large Images

- 컴퓨터 비전 문제 : 입력이 매우 클 수 있음
  - 64×64 크기의 컬러 이미지 → 64 x 64 × 3 = 12,288
  - 1000×1000 크기 컬러 이미지 → 1000 x 1000 × 3 = 3,000,000 (고해상도)

    ⇒ 과적합 방지 어려움 + 메모리 문제
- 합성곱 연산 구조를 통해 입력이 큰 문제를 해결할 수 있음

모서리 감지 예시

Computer Vision Problem

- 신경망 하위층 : 모서리 감지 → 다음 층 : 가능성있는 물체 감지 → 이후 층 : 온전한 물체 확인

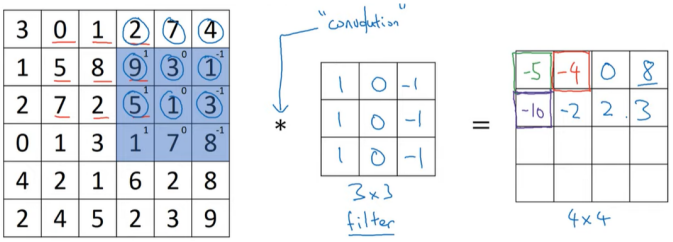
- 모서리 감지 방법

  1) 이미지에서 수직인 모서리를 찾기

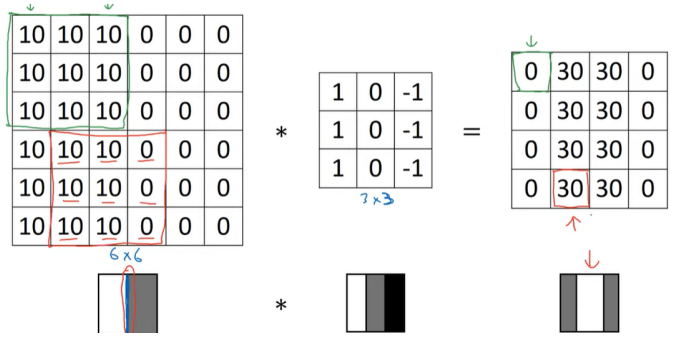
  2) 이미지에서 수평인 모서리를 찾기

Vertical Edge Detection

- 수직 모서리 감지를 위해 필터(3x3 행렬)를 만든 뒤 입력과 합성곱(convolution)



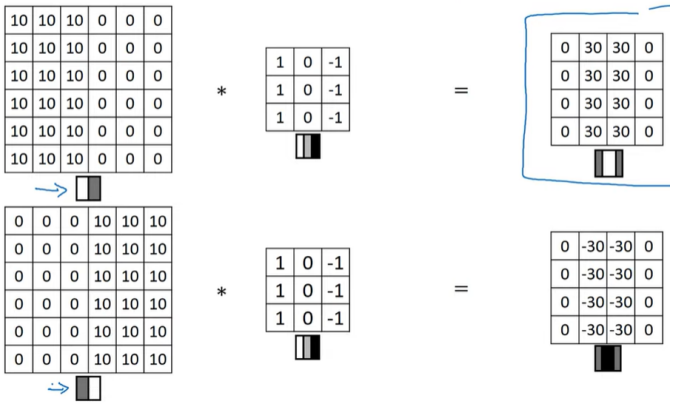
- 프로그래밍으로 구현할 때 합성곱은 별도의 함수로 수행
  - python : conv_forward
  - tensorflow : tf.nn.conv2d
  - keras : conv2D
- 어떻게 수직 모서리를 감지할까?



  - 밝은 부분 → 어두운 부분 사이 경계에 존재하는 세로선을 감지
  - 3×3 필터 : 왼쪽에는 밝은 픽셀, 오른쪽에는 어두운 픽셀
  - 고화질 이미지를 사용하면 더 정교하게 수직 경계선을 찾을 수 있음

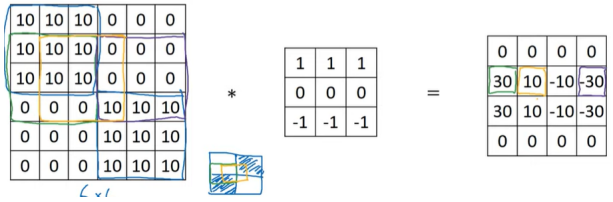
더 많은 모서리 감지 예시

Vertical Edge Detection Examples



Vertical and Horizontal Edge Detection

- 수직윤곽선 : 상대적으로 왼쪽이 밝고 오른쪽이 어두움
- 수평윤곽선 : 상대적으로 위쪽이 밝고 아래쪽이 어두움
- 수평 모서리 감지 예



- 서로 다른 필터가 세로 또는 가로 윤곽선을 감지함

Learning To Detect Edges

1	0	-1
1	0	-1
1	0	-1

vertical

1	0	-1
2	0	-2
1	0	-1

sobel

3	0	-3
10	0	-10
3	0	-3

scharr

1	1	1
0	0	0
-1	-1	-1

horizontal

1	2	1
0	0	0
-1	-2	-1

sobel

3	10	3
0	0	0
-3	-10	-3

scharr

- Sobel filter : 중간 부분의 픽셀에 더 중점을 둠
- Scharr filter
- filter의 값을 변수로 두고 역전파로 필터를 학습

패딩(Padding)

Padding

- padding 적용 전 : (n x n) * (f x f) = (n-f+1 x n-f+1)
  - 합성곱 연산마다 이미지가 축소되는 문제 → 심층 신경망에서 큰 문제
  - 가장자리 픽셀은 결과 이미지에서 한 번만 사용되는 문제 → 정보 손실

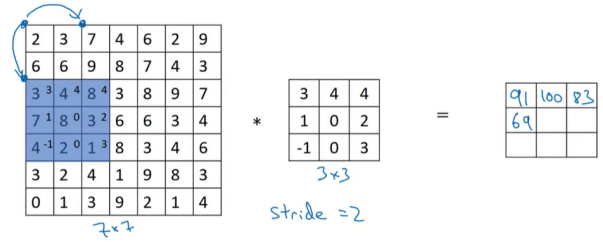
  ⇒ padding 적용
- padding (p=1) : (n+2 x n+2) * (f x f) = (n+2p-f+1 x n+2p-f+1)

Valid and Same Convolutions

- Valid(유효 합성곱) : No padding
  - (n x n) * (f x f) = (n-f+1 x n-f+1)
- Same(동일 합성곱) : Pad so that output size = input size
  - (n+2p x n+2p) * (f x f) = (n+2p-f+1 x n+2p-f+1)
  - n+2p-f+1 = n 을 만족하는 p = (f-1)/2
- 컴퓨터 비전에서 필터 크기는 홀수
  - 홀수의 필터에는 중심 픽셀이 존재
  - 짝수이면 비대칭 → 패딩을 왼쪽과 오른쪽 다르게 적용해주어야 함

스트라이드(Stride)

Strided Convolution



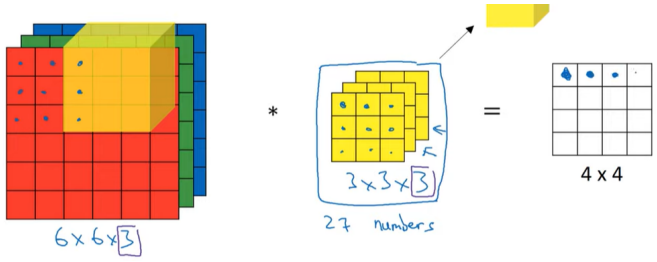
- stride : 필터를 움직이는 간격 → 행 방향, 열 방향 모두 stride만큼 움직임
- padding p, stride s일 때,
  (n x n) * (f x f) = ($\lfloor \frac{n+2p-f}{s} + 1 \rfloor$ x $\lfloor \frac{n+2p-f}{s} + 1 \rfloor$)
- 정수 형태가 아닌 경우 내림을 해줌
  → 보통 필터에 맞춰서 정수가 되도록 패딩과 스트라이드 수치를 맞춤

Technical Note on Cross-Correlation vs Convolution

- 일반적인 합성곱 : 합성곱 전 필터를 가로축과 세로축으로 뒤집는 연산을 진행
- 강의에서 정의한 합성곱 = 뒤집는 연산이 없는 교차상관
  ⇒ 딥러닝 분야에서는 교차상관을 합성곱이라고 부름

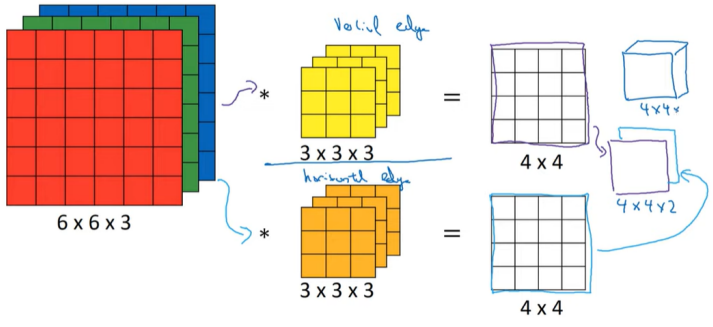
입체형 이미지에서의 합성곱

Convolutions on RGB Images



- 입력 : 6 x 6 × 3 → 3D 필터 사용 : 3 x 3 × 3 (높이 x 넓이 x 채널) ⇒ 결과 : 4 x 4 × 1
- 이미지의 채널 수 = 필터의 채널 수
- 합성곱 : 27개의 곱을 더함 (R:9, G:9, B:9)
- 필터 값을 조정해서 하나의 색에 대해서만 윤곽선을 구하는 것도 가능
  → 채널 별로 다른 필터 값을 사용할 수 있음

Multiple Filters



- 2개의 필터 → 2의 부피를 갖는 출력
- p=0, s=1 → (n x n x n_c) * (f x f x n_c) = (n-f+1 x n-f+1 x n_c')
- 채널의 수 = 깊이

합성곱 네트워크의 한 계층 구성하기

Example of a Layer

- ReLU((4×4 output) + bias(실수)) : 편향과 비선형성을 추가

  ⇒ 합성곱의 한 계층을 구성
- $z^{[1]} = w^{[1]} * a^{[0]} + b^{[1]}$ = 필터 * 입력 + 편향 → 합성곱 (선형 연산)
- $a^{[1]} = g(z^{[1]})$: 비선형성 적용

Number of Parameters in One Layer

- If you have 10 filters that are 3 × 3 × 3 in one layer of a neural network, how many parameters does that layer here?
  - 각각의 필터 : 3 x 3 × 3 → 27개의 변수 + 1개의 bias = 28 parameters
  - 10개의 필터 존재 → 28 * 10 = 280 parameters

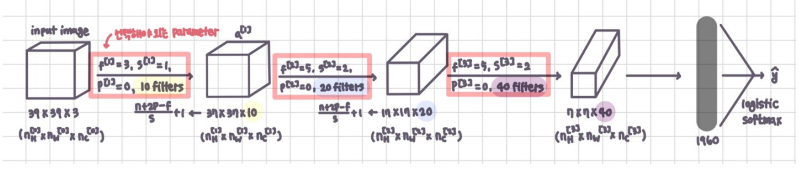
    ⇒ input 크기에 상관없이 파라미터 개수 고정

Summary of Notation

- $f^{[l]}$: filter size
- $p^{[l]}$: padding
- $s^{[l]}$: stride
- $n_c^{[l]}$: number of filters
- input : $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$
- output : $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$
- $n^{[l]} = \lfloor \frac{n^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \rfloor$
- each filter : $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$
- activations : $a^{[l]} \rightarrow n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$
  - $A^{[l]} \rightarrow m \times n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$ (m : 배치 사이즈)
- weights : $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$
- bias : $1 \times 1 \times 1 \times n_c^{[l]}$ → 필터마다 하나의 값을 가짐

간단한 합성곱 예시

Example ConvNet



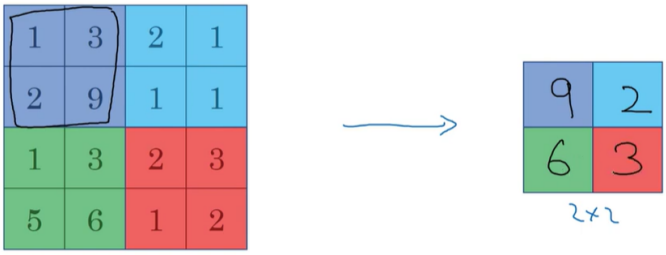
- 신경망이 깊어질수록 이미지 크기는 줄어들고 채널의 수는 늘어남

Types of Layer in a Convolutional Network

- Convolution
- Pooling
- Fully Connected

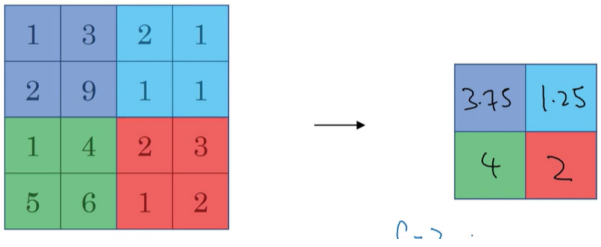
풀링(Pooling)층

Pooling Layer : Max Pooling



- Hyperparameters : f=2, s=2
- 어떤 4x4 영역에서 가장 큰 값이 가장 중요한 특성이라고 판단 → 최대풀링 결과
- 학습할 수 있는 변수 없음 = No parameters
- (n x n) → ($\lfloor \frac{n+2p-f}{s} + 1 \rfloor$ x $\lfloor \frac{n+2p-f}{s} + 1 \rfloor$)
- 3차원에서의 풀링 → 각 채널에 적용하여 동일한 채널 수를 가짐
  (5 × 5 x 채널) → (3 × 3 x 채널)

Pooling Layer : Average Pooling



- 최대값 대신 평균을 취함
- 평균 풀링보다 최대 풀링을 주로 사용
  - 예외 : 아주 깊은 신경망

Summary of Pooling

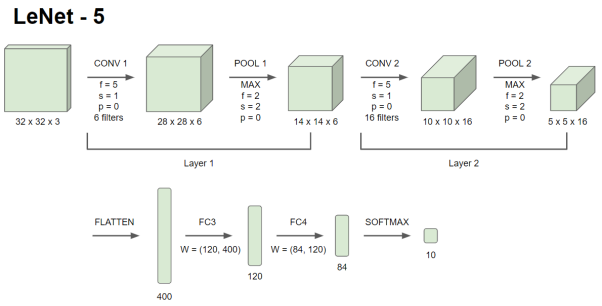
- Hyperparameters
  - f : filter size
  - s : stride
  - Max or Average

- p : padding (일반적으로 사용하지 않음)

- $(n_H \times n_W \times n_c) \rightarrow (\lfloor \frac{n_W - f}{s} + 1 \rfloor \times \lfloor \frac{n_H - f}{s} + 1 \rfloor \times n_c)$
- No parameters to learn → 역전파할 변수가 없음

CNN 예시

Neural Network Example (LeNet-5와 유사한 방법으로)



- 신경망 층을 셀 때는 파라미터가 있는 것을 기준으로 함

  → convolution층과 pooling층을 하나로 묶어서 하나의 층으로 여김

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072	0
CONV1 (f=5, s=1)	(28,28,8)	6,272	208
POOL1	(14,14,8)	1,568	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	416
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48,001
FC4	(84,1)	84	10,081
Softmax	(10,1)	10	841

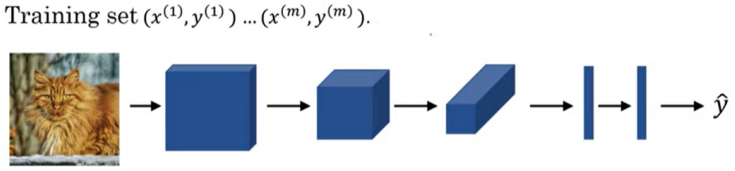
- 대부분의 parameters는 완전연결층에 존재, 풀링층에는 존재하지 않음

왜 합성곱을 사용할까요?

Why Convolutions

- 변수 공유 : 이미지 특성을 검출하는 필터를 공유하여 적은 변수를 필요로 함
- 희소 연결 : 출력값이 이미지 일부(작은 입력값)에 영향을 받기 때문에 과적합 방지 가능
- 이동 불변성 포착 : 이미지가 조금 변형되더라도 감지할 수 있음

Putting It Together



- 완전연결층 : Cost $J = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$

  → 여러 경사하강법 알고리즘으로 최적화 가능