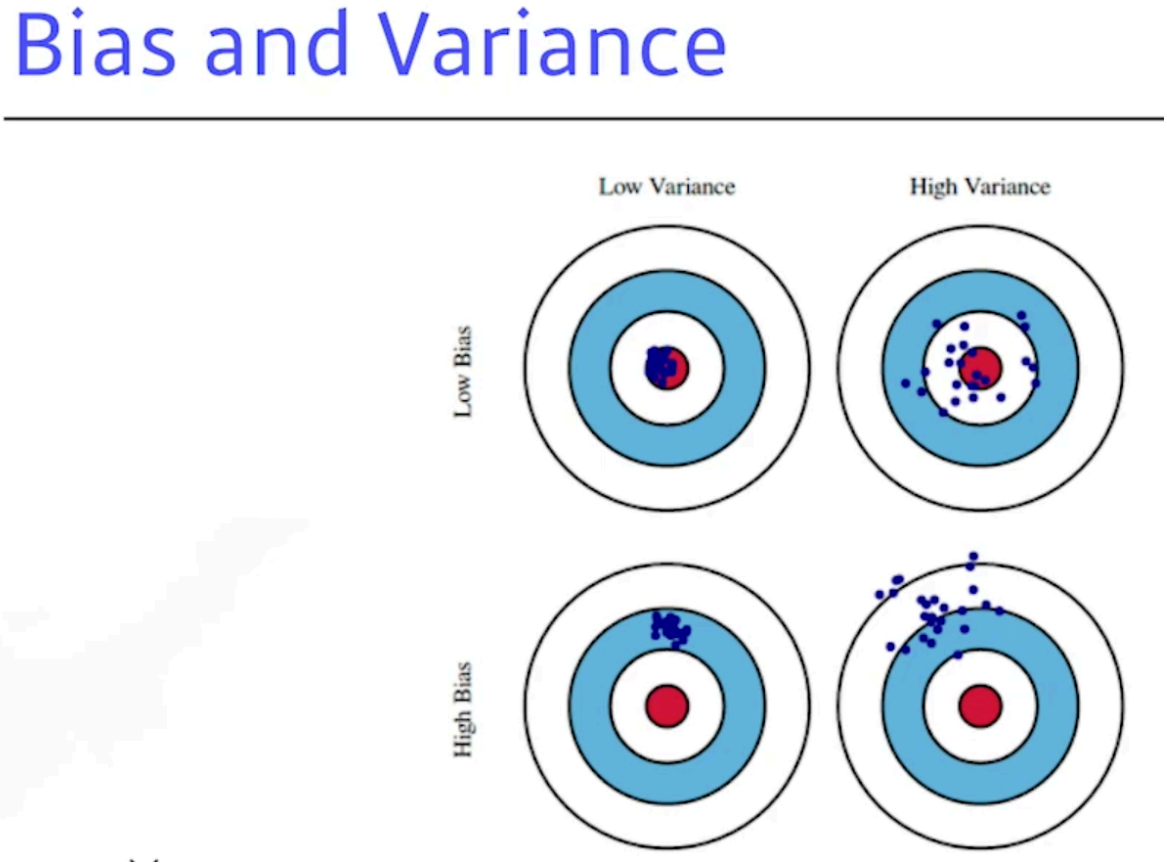
Week3-2

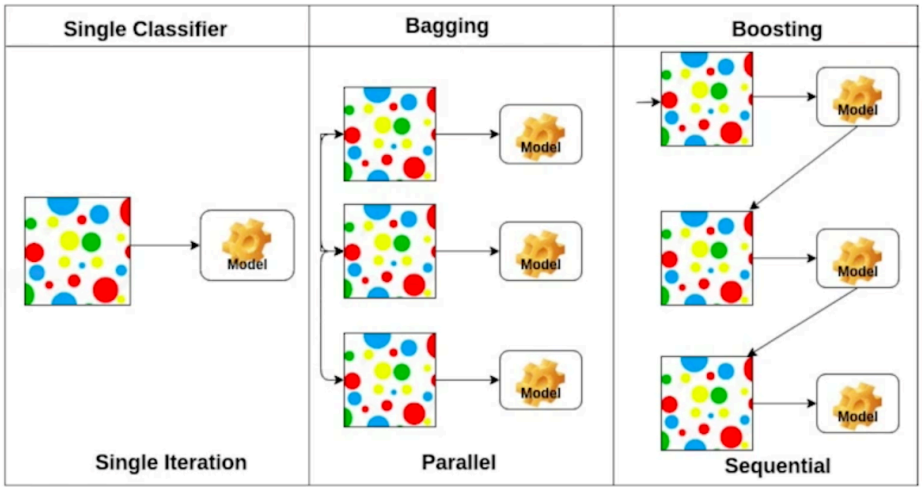
1. 학습정리
    1. 최적화
        1. 일반화
            1. 일반화 에러
                1. 트레이닝 에러와 테스트 에러의 갭
                2. 높으면 오버피팅
            2. cross-validation
                1. 트레이닝 데이터를 k 개로 나눔
                2. 그 중 하나로 validation 하고, 나머지로 학습시킴
                3. 테스트 데이터는 어떤식으로든 활용하면 안됨
            3. Bias
                1. 바이어스가 낮으면 트루타겟에 접근
            4. Variance
                1. 출력이 얼마나 일관적인지
                2. 낮으면 밀집



            5. 부트스트래핑
                1. 정해진 수의 트레이닝 셋의 서브샘플 여러개로 여러개의 모델을 만드는 것
                2. 배깅 (패러랠)
                    1. 여러개 모델의 평균이나 보팅으로 출력
                3. 부스팅 (시퀀셜)

1. 간단한 모델을 만듦 (위크 러너)
2. 그 모델이 잘 학습하지 못하는 걸 잘 학습하는 모델을 만듦
3. 이런식으로 생성된 여러개의 모델을 시퀀셜하게 합쳐서 스트롱 러너를 만듦

Bagging vs. Boosting

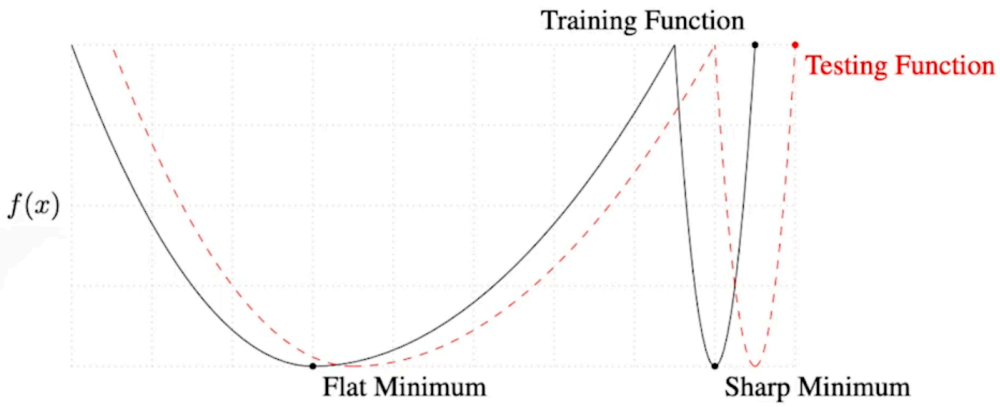


6. 그래디언트 디스센트

Gradient Descent Methods

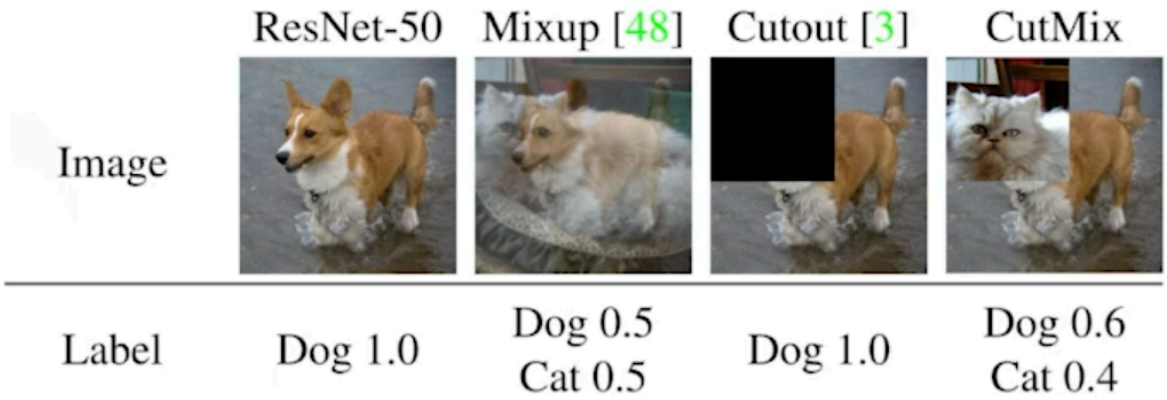
- Stochastic gradient descent
  - Update with the gradient computed from a single sample.
- Mini-batch gradient descent
  - Update with the gradient computed from a subset of data.
- Batch gradient descent
  - Update with the gradient computed from the whole data.

Batch-size Matters



    1. 배치 사이즈가 크면 샤프 미니멈
    2. 배치 사이즈가 작으면 플랫 미니멈
    3. 샤프 미니멈에선 트레이닝 하지 않은 데이터에 대해 오차가 큼
    4. 배치 사이즈를 줄이면 일반화가 좋아짐
    5. 러닝 레이트를 정하기 어려움
  7. 모멘텀
    1. 관성의 개념
    2. 관성때문에 로컬 미니멈으로 수렴하지 못하는 현상이 생김
  8. NAG
    1. 이동할 곳에서 그래디언트를 계산함
    2. 로컬 미니멈에 좀 더 빨리 수렴함
  9. adagrad
    1. 많이 변한 파라미터는 적게 변화시키고
    2. 적게 변한 파라미터는 많이 변화시킴
    3. 뒤로 가면 갈 수록 학습이 멈춤
  10. adadelta
    1. 아다그라드의 단점 보완
    2. 러닝레이트가 없음
    3. 바꿀 수 있는 게 별로 없어서 잘 안쓰임
  11. RMSprop
    1. 걍 잘되더라
  12. adam
    1. 가장 잘 됨
    2. 적응형 러닝 레이트와 모멘텀을 잘 합침
2. 규제 (Regulaization)
  1. 학습데이터 뿐만 아니라 테스트데이터에도 잘 되도록 하는 것
  2. 얼리 스타핑
  3. 파라미터 놈 패널티
    1. 뉴럴 네트워크의 파라미터가 너무 커지지 않게 함
    2. 함수를 부드럽게 함
  4. data augmentation
    1. 데이터를 약간 변화시켜 수를 늘림

5. 노이즈 robustness
    1. 입력과 웨이트에 노이즈를 넣음
6. 레이블 스무딩
    1. 디시전 바운더리를 부드럽게 만들어 줌
    2. 분류 문제의 성능을 크게 올릴 수 있음
    3. 믹스업
        1. 두 개의 라벨과 인풋을 섞어버림
    4. 컷 아웃
        1. 주어진 이미지의 일정 영역을 빼버림
    5. 컷 믹스
        1. 이미지의 특정 영역을 다른 이미지를 넣음



7. 드롭아웃
    1. 몇몇 뉴런을 탈락시킴
8. 배치 노말라이제이션
    1. 간단한 분류문제에 활용하면 좋음

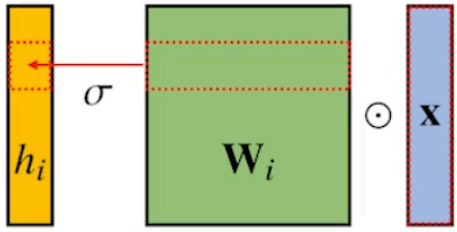
2. CNN 첫걸음

Convolution 연산 이해하기

- 지금까지 배운 다층신경망(MLP)은 각 뉴런들이 선형모델과 활성함수로 모두 연결된 (fully connected) 구조였습니다

$$h_i = \sigma\left(\sum_{j=1}^{p} W_{ij} x_j\right)$$

활성함수 가중치 행렬



각 성분 h_i 에 대응하는 가중치 행 $\mathbf{W}_i$ 이 필요하다

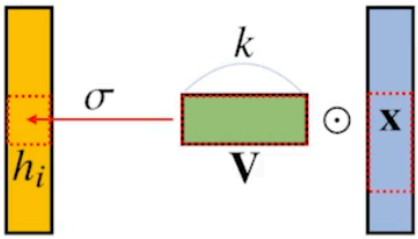
Convolution 연산 이해하기

- Convolution 연산은 이와 달리 커널(kernel)을 입력벡터 상에서 움직여가면서 선형모델과 합성함수가 적용되는 구조입니다

커널 사이즈

$$h_i = \sigma\left(\sum_{j=1}^{k} V_j x_{i+j-1}\right)$$

활성함수 가중치 행렬



활성화 함수를 제외한 Convolution 연산도 선형변환에 속합니다

1. 컨볼루션 연산
    1. 공통된 커널을 입력벡터상에서 움직여 가면서 선형모델과 합성함수가 적용됨
    2. 엄청난 크기의 가중치 행렬을 사용하지 않음 (MLP는 사용)
    3. i 와 상관없이 공통된 커널을 사용해서 파라미터 사이즈를 많이 줄일 수 있음
    4. 신호를 커널을 이용해 국소적으로 증폭 or 감소시켜서 정보를 추출

다양한 차원에서의 Convolution

- Convolution 연산은 1차원뿐만 아니라 다양한 차원에서 계산 가능합니다

1D-conv $$[f * g](i) = \sum_{p=1}^{d} \boxed{f(p)} g(i+p)$$

i, j, k 가 바뀌어도 커널 f 의 값은 바뀌지 않습니다

2D-conv $$[f * g](i, j) = \sum_{p,q} \boxed{f(p,q)} g(i+p, j+q)$$

3D-conv $$[f * g](i, j, k) = \sum_{p,q,r} \boxed{f(p,q,r)} g(i+p, j+q, k+r)$$

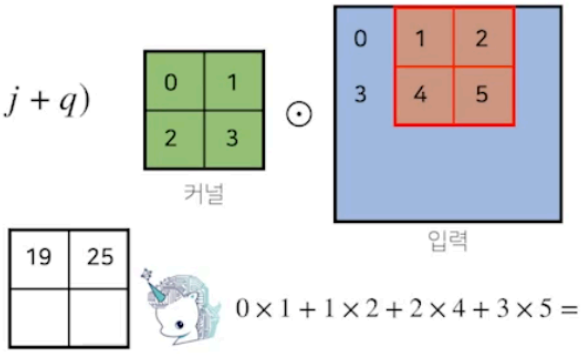
2차원 Convolution 연산 이해하기

- 2D-Conv 연산은 이와 달리 커널(kernel)을 입력벡터 상에서 움직여가면서 선형모델과 합성함수가 적용되는 구조입니다

$$[f * g](i, j) = \sum_{p,q} f(p, q) g(i + p, j + q)$$

커널

입력



$0 \times 1 + 1 \times 2 + 2 \times 4 + 3 \times 5 = 25$

×

2차원 Convolution 연산 이해하기

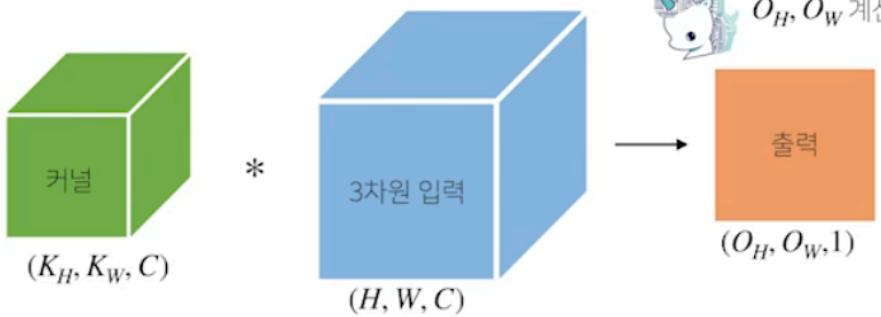
- 2D-Conv 연산은 이와 달리 커널(kernel)을 입력벡터 상에서 움직여가면서 선형모델과 합성함수가 적용되는 구조입니다
- 입력 크기를 (H, W), 커널 크기를 (K_H, K_W), 출력 크기를 (O_H, O_W) 라 하면 출력 크기는 다음과 같이 계산합니다

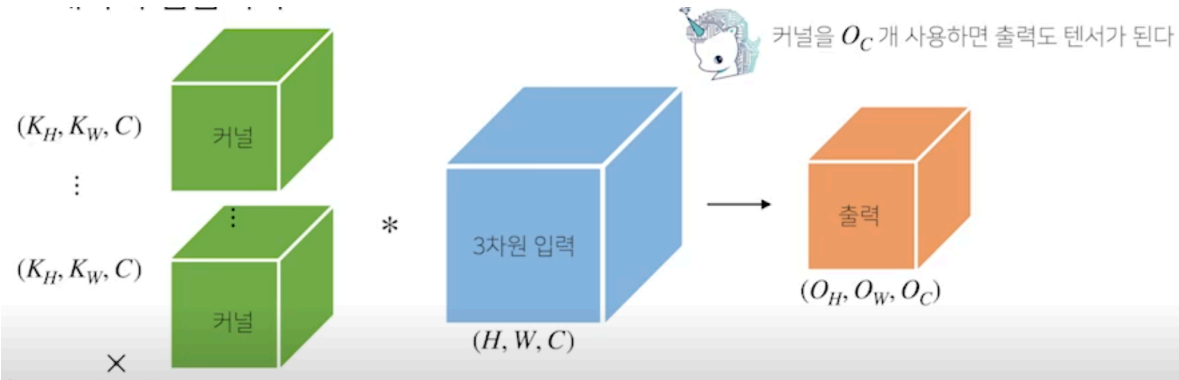
$$O_H = H - K_H + 1$$
$$O_W = W - K_W + 1$$

가령 28x28 입력을 3x3 커널로
2D-Conv 연산을 하면 26x26 이 된다

5. 커널f(p)이 위치(i,j,k)에 따라 바뀌지 않음

O_H, O_W 계산법은 앞 슬라이드 참조

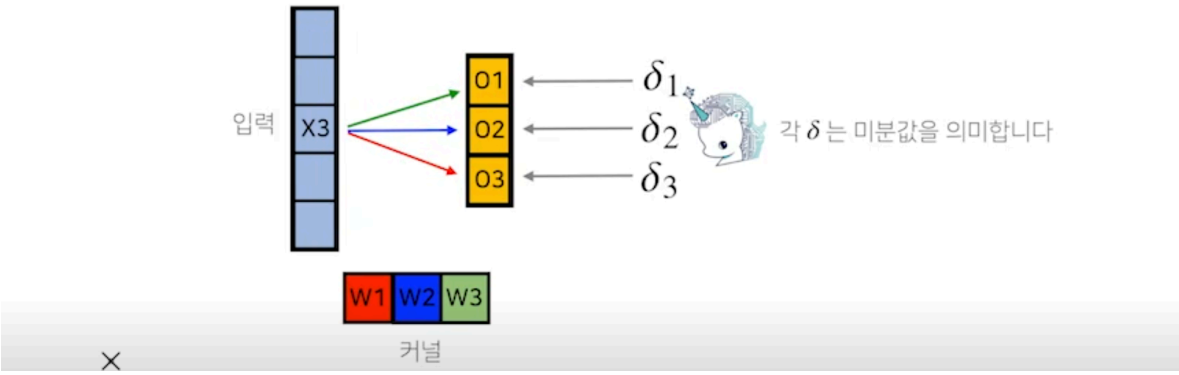




2. 컨볼루션 채널 여러개일 때 (입력이 텐서일 때)
    1. 채널의 개수만큼 커널을 만듦
    2. 각각의 2차원 입력에 컨볼루션 연산
    3. 결과물들을 더해줌
    4. 이렇게 하면 출력의 차원은 2차원
    5. 출력도 텐서로 받고 싶으면 커널을 많이 만들어서 적용

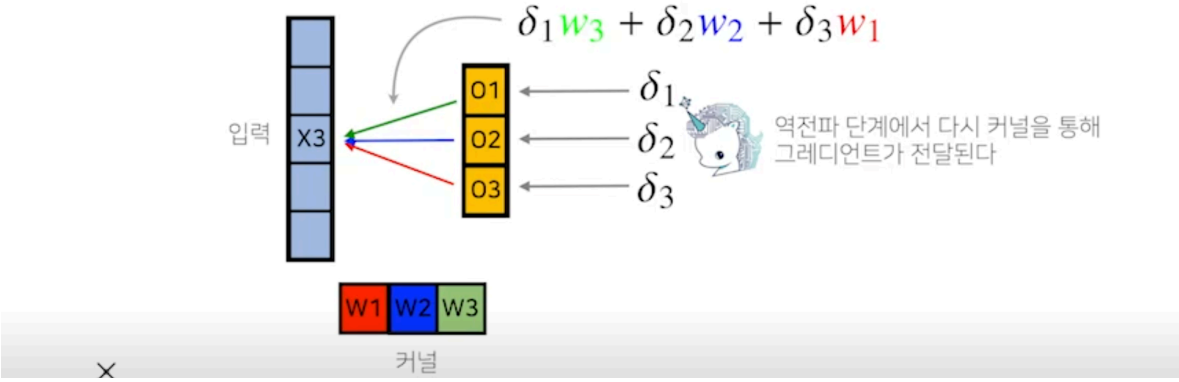
Convolution 연산의 역전파 이해하기

- Convolution 연산은 커널이 모든 입력데이터에 공통으로 적용되기 때문에 역전파를 계산할 때도 convolution 연산이 나오게 됩니다

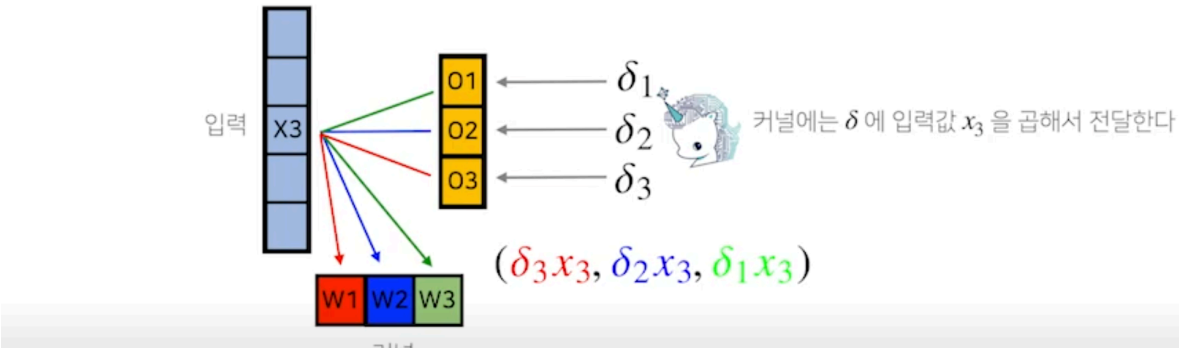


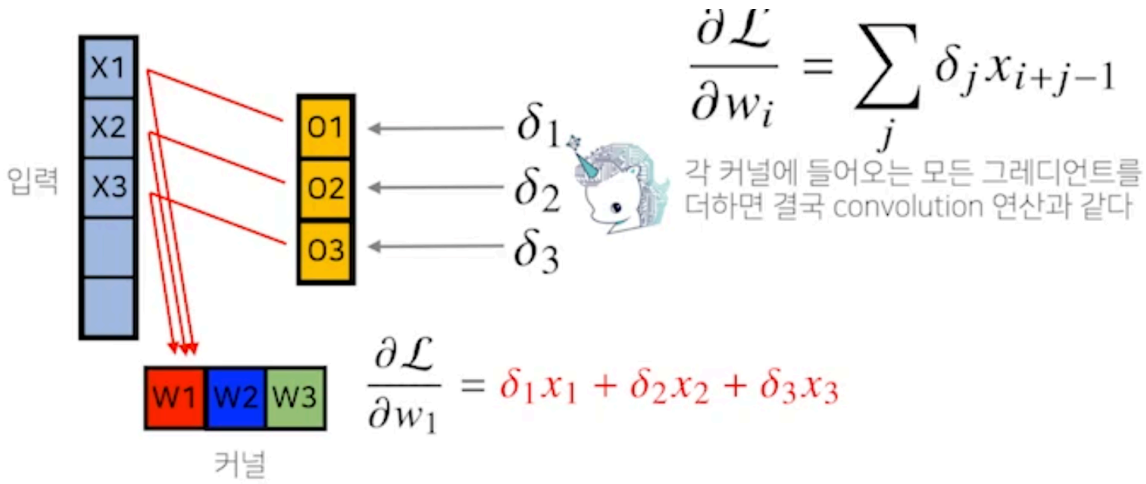
Convolution 연산의 역전파 이해하기

- Convolution 연산은 커널이 모든 입력데이터에 공통으로 적용되기 때문에 역전파를 계산할 때도 convolution 연산이 나오게 됩니다



Convolution 연산은 커널이 모든 입력데이터에 공통으로 적용되기 때문에 역전파를 계산할 때도 convolution 연산이 나오게 됩니다





3. 컨볼루션의 역전파
    1. 포워드 할 때 각 커널의 값을 곱해서 넣음
    2. 백워드 때는 미분에 곱해졌던 커널에 곱해졌던 인풋값 곱해서 전달
    3. 위 값이 커널에 그래디언트로 전달됨

2. **피어세션**
    1. 강의리뷰
        1. 어제보다 이해하기 쉬웠다는 공통된 의견

    2. 실습코드도 어제보다 눈에 들어왔음
2. numpy 스피드퀴즈
    1. numpy 강의에 나온 코드의 출력결과 맞추기
    2. 다양한 numpy 함수의 쓰임새 파악