1. 합성곱 층의 필요성

이미지를 분석할 때는 단순히 픽셀 데이터만을 사용하지 않고, 공간적 구조와 패턴을 고려해야 합니다. 전통적인 신경망에서는 이미지를 한 줄로 펼쳐서 처리하기 때문에 데이터의 공간적 정보가 손실되기 쉽습니다. 이를 방지하기 위해 도입된 것이 **합성곱 층**입니다.

2. 합성곱 신경망의 구조

합성곱 신경망은 주로 다음 다섯 개의 계층으로 구성됩니다.

1. **입력 층 (Input Layer)**: 이미지 데이터를 가장 먼저 받는 층입니다. 이미지의 높이, 너비, 채널(RGB 채널 등) 정보를 사용하여 3차원 배열 형태로 데이터를 입력받습니다.
2. **합성곱 층 (Convolutional Layer)**: 이미지의 **특징**을 추출하는 역할을 합니다. 이미지 전체를 처리하는 대신, 이미지의 작은 부분(국소적 영역)을 커널(필터)이라는 도구로 훑어가며 특징을 추출합니다.
3. **풀링 층 (Pooling Layer)**: 합성곱 층에서 추출한 특징 맵의 크기를 줄여 연산량을 감소시키고, 모델의 성능을 높입니다. 주로 **최대 풀링(Max Pooling)**을 사용하여 중요한 특징을 유지하면서 데이터를 압축합니다.
4. **완전 연결 층 (Fully Connected Layer)**: 추출된 특징을 바탕으로 이미지를 분류하거나 예측하는 역할을 합니다. 1차원 벡터로 변환된 후, 이 층에서 신경망은 각 클래스에 대해 확률을 계산합니다.
5. **출력 층 (Output Layer)**: 최종 결과를 출력하는 층입니다. 주로 **소프트맥스(Softmax)** 함수와 같은 활성화 함수를 사용하여 각 클래스의 확률을 계산합니다.

3. 합성곱 층

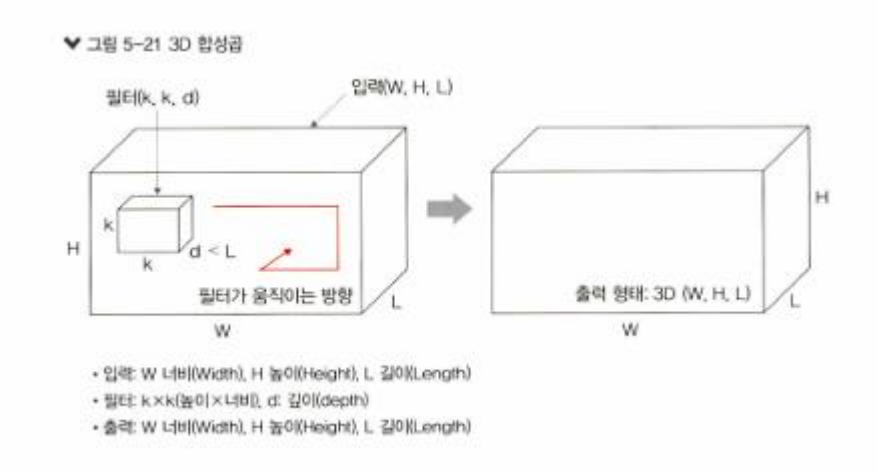
합성곱 층에서는 **커널(필터)**을 사용해 이미지의 특징을 감지합니다. 커널은 작은 크기(3x3, 5x5 등)로 지정되어 이미지의 국소적 부분을 훑어가며, 해당 부분의 정보를 추출합니다. 이때, **스트라이드(stride)**라는 간격에 따라 커널이 이미지 상에서 이동하며 연산을 진행합니다.

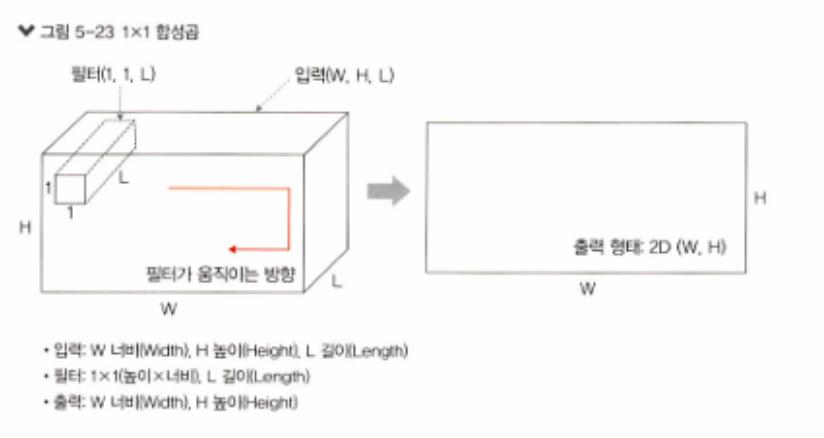
예를 들어, 스트라이드가 1일 경우, 커널이 한 칸씩 이동하면서 이미지를 스캔하게 됩니다. 각 커

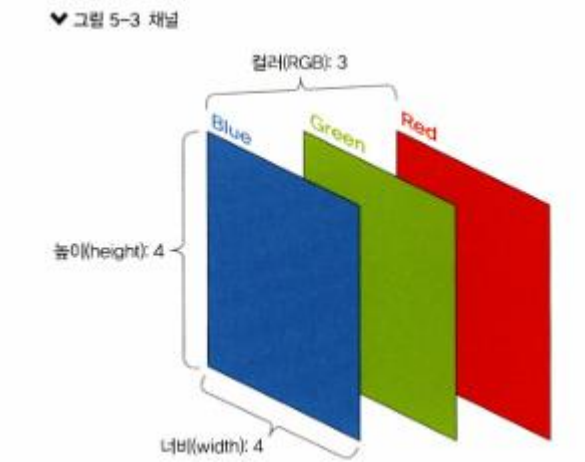
널이 적용된 영역의 픽셀 값과 커널의 값이 대응하여 곱해진 후, 이 결과들이 모두 더해져 새로운 값(특징 맵의 요소)이 생성됩니다.

1. 합성곱 연산

- **이미지 크기 변화**:
    - 원본 이미지 크기가 (6, 6, 1)이고, 3×3 크기의 커널/필터를 스트라이드 1로 이동하며 합성곱 연산을 수행하면 출력 특성 맵의 크기는 (4, 4, 1)이 됩니다.
- **컬러 이미지 합성곱**:
    - **특징**:
        - 필터의 채널 수는 3이며, 이는 RGB 각 채널에 대해 필터가 적용됨을 의미합니다.
        - 각 채널별로 다른 가중치를 사용하여 합성곱을 적용한 후 결과를 합산합니다.
    - **주의사항**:
        - 필터 채널이 3이라고 해서 필터 개수가 3개인 것은 아니며, 실제로는 필터 개수가 1개입니다.







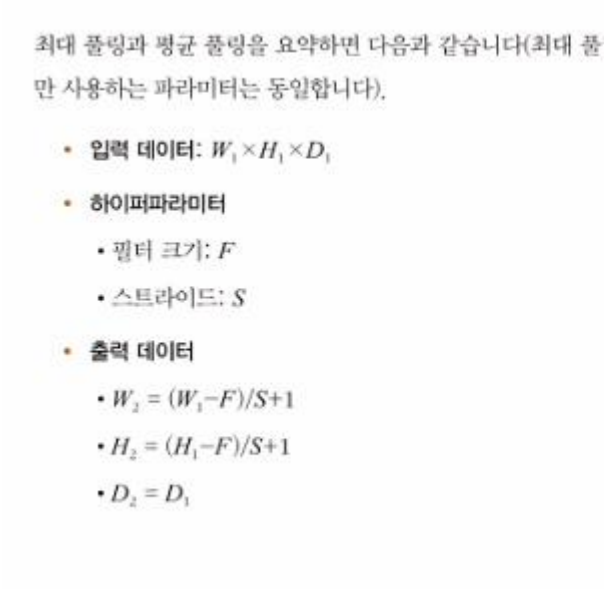
즉, 합성곱층을 요약하면 다음과 같습니다.

- **입력 데이터**: $W_1 \times H_1 \times D_1$($W_1$: 가로, $\times H_1$: 세로, $\times D_1$: 채널 또는 깊이)
- **하이퍼파라미터**
  - 필터 개수: K
  - 필터 크기: F
  - 스트라이드: S
  - 패딩: P
- **출력 데이터**
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$
  - $D_2 = K$

2. 풀링 층

- **목적**: 특성 맵의 차원을 축소(다운샘플링)하여 연산량을 감소시키고, 중요한 특성 벡터를 추출하여 학습 효율을 높입니다.
- **다운샘플링**: 이미지나 특성 맵의 크기를 줄이는 과정입니다.

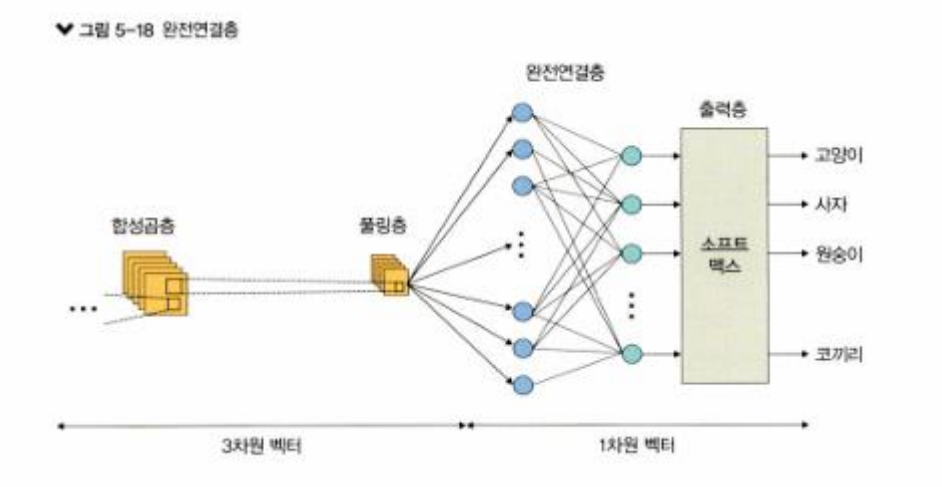
- **풀링 종류**:
    - **최대 풀링(Max Pooling)**: 대상 영역에서 최댓값을 추출합니다.
    - **평균 풀링(Average Pooling)**: 대상 영역의 평균값을 계산합니다.
- **계산 과정 예시**:
    - **최대 풀링**:
        - 첫 번째 풀링 영역: [3, -1, -3, 1] → 최댓값 3 선택
        - 두 번째 풀링 영역: [12, -1, 0, 1] → 최댓값 12 선택
        - 세 번째 풀링 영역: [2, -3, 3, -2] → 최댓값 3 선택
    - **평균 풀링**:
        - 첫 번째 풀링 영역: 평균 계산
        - 두 번째 풀링 영역: 평균 계산
        - 세 번째 풀링 영역: 평균 계산


최대 풀링과 평균 풀링을 요약하면 다음과 같습니다(최대 풀링과 평균 풀링의 계산 과정은 다르지 만 사용하는 파라미터는 동일합니다).

- 입력 데이터: $W_1 \times H_1 \times D_1$
- 하이퍼파라미터
    - 필터 크기: F
    - 스트라이드: S
- 출력 데이터
    - $W_2 = (W_1 - F)/S + 1$
    - $H_2 = (H_1 - F)/S + 1$
    - $D_2 = D_1$

3. 완전 연결 층

- **역할**: 합성곱 층과 풀링 층을 거치면서 축소된 특성 맵을 1차원 벡터로 펼쳐서(flatten) 완전 연결 층에 전달합니다.
- **과정**: 이미지 데이터가 3차원 벡터에서 1차원 벡터로 변환됩니다.
- **활용**: 분류 작업 등의 최종 예측을 수행합니다.



4. 출력 층

- **활성화 함수**: 소프트맥스(Softmax)를 사용하여 입력 받은 값을 0에서 1 사이의 확률로 변환합니다.
- **결과 해석**: 각 클래스에 속할 확률 값을 출력하며, 가장 높은 확률을 가진 클래스가 최종 예측 결과로 선택됩니다.

5. 1D, 2D, 3D 합성곱

- **1D 합성곱**:
  - **특징**: 필터가 시간 축을 따라 좌우로만 이동합니다.
  - **출력 형태**: 1차원 배열.
  - **예시**: 신호 처리나 시계열 데이터에 사용.
- **2D 합성곱**:
  - **특징**: 필터가 가로와 세로 두 방향으로 이동합니다.
  - **출력 형태**: 2차원 행렬.
  - **예시**: 이미지 처리에 일반적으로 사용.
- **3D 합성곱**:
  - **특징**: 필터가 가로, 세로, 깊이 세 방향으로 이동합니다.
  - **출력 형태**: 3차원 배열.
  - **예시**: 영상 데이터 처리 등에 사용.

1. 심층 신경망 모델 정의 (FashionDNN)

심층 신경망(DNN)은 다음과 같이 정의됩니다:

- **Linear Layers (선형 계층)**:
    - fc1: 입력 크기 784 (28x28 이미지 펼친 크기), 출력 크기 256.
    - fc2: 입력 크기 256, 출력 크기 128.
    - fc3: 입력 크기 128, 출력 크기 10 (10개의 클래스).
- **Dropout Layer**:
    - 드롭아웃(drop)은 0.25 비율로 뉴런을 무작위로 제거하여 과적합을 방지합니다.
- **활성화 함수**:
    - ReLU 활성화 함수(F.relu)가 사용되어 각 층의 출력값을 활성화합니다.

❤ 표 5-1 nn.xx와 nn.functional.xx의 사용 방법 비교

구분	nn.xx	nn.functional.xx
형태	nn.Conv2d: 클래스 nn.Module 클래스를 상속받아 사용	nn.functional.conv2d: 함수 def function (input)으로 정의된 순수한 함수
호출 방법	먼저 하이퍼파라미터를 전달한 후 함수 호출을 통해 데이터 전달	함수를 호출할 때 하이퍼파라미터, 데이터 전달
위치	nn.Sequential 내에 위치	nn.Sequential에 위치할 수 없음
파라미터	파라미터를 새로 정의할 필요 없음	가중치를 수동으로 전달해야 할 때마다 자체 가중치를 정의

2. 손실 함수와 옵티마이저 설정

- **손실 함수**: 교차 엔트로피 손실 함수(nn.CrossEntropyLoss())를 사용합니다. 이는 분류 문제에 자주 사용됩니다.
- **옵티마이저**: Adam 옵티마이저를 사용하며, 학습률(learning_rate)은 0.001로 설정됩니다.

3. 학습 과정

다음 코드는 모델을 학습시키는 과정입니다:

- **이미지 전처리**: 이미지를 (100, 1, 28, 28) 크기로 변환하여 학습에 사용합니다.
- **순전파**: 모델에 데이터를 입력하여 출력값을 얻습니다.
- **손실 계산 및 역전파**: 손실을 계산하고 역전파로 가중치를 업데이트합니다.