Lec 14 : Convolutional Neural Network

대뇌의 시각피질 연구로부터 시작되었고, 주로 Computer Vision에서 활용되나, 그에 국한되지 않고 자연어처리와 음성인식에도 많이 사용된다.

1. **시각피질의 구조:**

   시각피질의 뉴런들은 전체 시각 자극에 반응하지 않는다.

   "일부 범위 안에 있는" 시각 자극에만 반응한다. : 여기서 '일부 범위'가 "국부 수용장"이다.

   국부수용장은 서로 겹쳐 있으며, 이를 합치면 전체 시야를 감싸게 된다.

   +또 어떤 뉴런들은 특정한 각도의 선분에, 어떤 뉴런들은 큰 수용장을 바탕으로 저수준 패턴이 조합된 복잡한 패턴에 반응한다.

   고수준의 뉴런이 저수준의 뉴런의 출력에 기반한다는 것이다.

   이러한 구조를 바탕으로, 시야에 있는 모든 종류의 다양한 패턴을 관찰할 수 있다.

> (기존의 심층 신경망도 이미지를 처리하는 능력이 있지만, 선형화되는 과정에서 위치정보가 상실되고,
>
> 픽셀 하나 당 하나의 클래스가 할당되어 클래스의 수가 폭증하는 등, 효율성이 상당히 떨어지는 문제를 가진다.)

CNN의 시초: LeNet-5 구조 :

합성곱층(Convolutional Layer)와 풀링층(Pooling Layer)이라는 두 가지 구성 요소가 확립된다.

- 합성곱층
  - 첫번째 합성곱층의 뉴런은, 이미지의 모든 픽셀이 아닌 **수용장 내부의 일부 픽셀**에만 연결된다.

    두번째 합성곱층의 뉴런들에는 첫번째 층의 작은 영역 안에 위치한 뉴런들이 연결된다.

    세번째 합성곱층에는 다시 두번째 층의 작은 영역에 위치한 뉴런들이 연결된다.

    ... (반복) ...

    즉, Lower Layer가 저수준의 , 기본적인 패턴을 인식하고 Upper Layer들은 Lower Layer의 출력으로 이루어진 새로운 패턴을 인식한다.

    > 다만, 상위 층으로 갈수록 층의 너비가 작아지며 데이터가 손실되는 문제가 발생하므로, 너비를 제로 패딩을 통해 유지한다.

    여기서 Lower Layer에 비해 Upper Layer의 크기가 얼마나 줄어드는지를 나타내는 Hyper Parameter를 **"Stride"** 라고 한다.

상위층의 i행, j열에 있는 뉴런이

이전층의 i X Sh 에서 i X Sh+Fh-1 까지의 행과 j X Sw 에서 j X Sw + Fw -1 까지의 열에 위치한 뉴런과 연결된다.

> Sh = Stride height Sw = Stride width Fh, Fw = Field height, width

- **Filter** (Convolution Kernel)

  일종의 가중치

  수용하려고 하는 부분을 제외한 나머지 수용장 내부의 픽셀들은 전부 무시한다.

  층의 전체 뉴런에 적용된 하나의 필터는 하나의 특성 맵을 만든다. 이 특성맵은 필터를 가장 크게 활성화하는 이미지의 영역을 강조한다.

  : CNN은 자동으로 가장 유용한 필터를 찾고, 상위 층은 이들을 연결하여 더욱 복잡한 패턴을 학습한다.

- 특성맵 쌓기

  이전 레이어가 필터의 stride를 통해 특성맵으로 만들어질 때, 그 특성맵의 한 픽셀은 하나의 뉴런에 해당한다. 또, 하나의 특성맵에서 모든 뉴런이 같은 패러미터를 공유한다. (따라서, 패러미터의 수가 급격히 줄어든다.)

  - 사실 패러미터는 공유될 수 밖에 없다. 패러미터가 바로 필터이기 때문이다.

  BUT, 다른 특성맵의 뉴런은 다른 패러미터를 사용한다.

  한 뉴런의 수용장은, 이전층에 있는 모든 특성맵에 걸쳐 확장된다.

  일반적으로 일반 이미지는 컬러 채널(R, G, B)마다 하나씩, 여러 서브층으로 구성된다.

- 메모리 요구사항

  합성곱층은 많은 양의 RAM 용량을 요구한다. Back Propagation 시, 정방향 계산시의 모든 중간값을 필요로 하기 때문이다.

  ex)

  5*5 필터, Stride = 1

  150*100크기의 특성맵 200개 만드는 convolution layer

  패러미터 수: (5 x 5 x 3 + 1) x 200 (특성 맵 200개)

  각 특성맵마다 150 x 100개의 뉴런, 5 x 5 x 3 개의 입력에 대한 가중합을 계산

  -> 2억 2천만개의 실수 곱셈

  만약 특성 맵이 32비트의 float으로 표현된다면, 사용되는 메모리 용량은

  하나의 샘플에 대해 200 x 150 x 100 x 32 = 9천 6백만 비트

  -> 컴퓨팅 자원이 굉장히 많이 든다.

- 풀링층(Pooling Layer)
  - 풀링의 목적은, 과대적합과 메모리 사용량을 줄이도록, 패러미터 수를 줄이기 위해 입력 이미지의 축소본인 **"부표본"**을 만드는 것이다.

        - 풀링층의 각 뉴런은, 이전 층의 수용장 내부의 뉴런의 출력과 연결되어있다.(합성곱 층과 동일)

          이전과 마찬가지로 스트라이드, 크기, 패딩을 지정해야 하나

          가중치가 없다. 즉, **합산 함수를 이용해 입력값을 더하는 게 전부다.**

          예를 들어, 2x2 MAX 풀링 커널(필터)에 1, 5, 4, 2의 입력값이 입력되었다면, 다음 층에는 오직 5만이 전달된다.

          이미지에서는 이웃한 픽셀 사이의 연관성이 깊기 때문에, 풀링이 효과적으로 적용될 수 있다.

        - 풀링은 작은 변화에도, 일정 수준의 불변성을 만들어준다. 예를 들어, 이미지가 몇 픽셀 이동하더라도, 크게 영향받지 않는다.
        - 풀링은 입력값의 많은 부분이 손실된다는, **매우 파괴적**이라는 점에서 단점도 가진다.

          불변성이 필요하지 않은 경우, 등변성이 목표가 되어야 하고, 입력의 작은 변화가 출력에 반영되어야 하므로,

          이 경우 풀링은 적합하지 않다.

- CNN의 일반적인 구조
    - CNN의 전형적인 구조는 **(입력 - 합성곱층1, 2, 3, ... - 풀링 - 합성곱층9, 10, 11, ... - 풀링 - ... - 완전연결층 - 출력)** 과 같다.

      이미지의 크기는 점 점 줄어들고, 대신 점 점 깊어진다.

      즉, 이미지 크기는 줄어들며 패턴의 수에 따라 그 깊이가 증가한다.
    - ex) MNIST 데이터셋 문제를 위한 CNN
        - 1st layer: 64 filters(7*7), stride = 1
        - pooling layer: max pooling layer(2*2)
        - repeat upper structure twice(이미지가 크면, 더 반복해도 된다.)
        - CNN 출력층에 가까울수록 필터 개수가 늘어난다. 저수준 특성은 몇개 없지만(원, 사각형, 수평선..) 이를 조합할 경우 많은 경우의 수가

          확인 될 수 있으므로, 이러한 구조가 합리적이다.
        - 두개의 hidden layer, 하나의 output layer로 구성된 완전연결 뉴럴 네트워크(과대적합 줄이기 위해, 50%의 dropout 비율을 가진 층 추가)

- 다양한 CNN 모델들 : LeNet-5 , AlexNet, GoogLeNet, VGGNet, ResNet, Xception, SENet

- AlexNet:
- GoogLeNet:
- ResNet:
- Xception:
- SENet:

- 분류와 위치추정
  - 물체의 위치를 **회귀** 작업으로 찾아내는 것
  - 물체에 대한 바운딩 박스의 **수평, 수직좌표, 높이, 너비**를 예측하는 것

  - **객체탐지**: CNN이 전 영역을 슬라이딩 하며 모든 nxn 필터 영역을 본다.
    - 조금씩 다른 위치에서 동일한 물체를 여러번 감지하며, 불필요한 바운딩 박스를 제거하기 위해 사후 처리가 필요하다.
      - 존재여부 점수가 가장 높은 바운딩 박스를 찾은 후 해당 박스와 많이 중첩된 바운딩 박스를 모두 제거한다.
      - BUT, CNN을 여러번 실행시켜야 하므로, 물체를 여러번 보는 동안 아주 느려진다.

    - 이러한 문제를 해결하기 위해, **완전 합성곱 신경망(FCN)**을 사용한다.
      - CNN 맨 위의 밀집층(Fully Connected Layer)을 합성곱 층으로 바꿀 수 있다.
      - 예를 들어, 기존에는 200개의 뉴런을 가진 밀집층이 200개의 특성맵을 생성했다면, FCN에서는 200개의 필터와 valid 패딩(패딩하지 않는)을 가진 합성곱층으로 바꾸는 것.
      - 둘 모두 200개의 1x1 이미지, 즉 숫자를 출력하며, 연산적 특징은 밀집층과 합성곱층이 동일하다.
    - FCN에서 합성곱층의 역할
      - 밀집층은, 입력 특성마다 하나의 가중치를 두므로 입력이 특정한 크기여야 하지만
      - 합성곱층은 필터의 Stride로 진행되는 만큼 입력의 크기에 구애받지 않는다.
      - 즉, FCN의 핵심은 모든 Layer를 Convolutional로 만들어서 이미지의 크기에 구애받지 않고 이미지를 처리할 수 있다는 것
- Semantic Segmentation
  - 일반적인 CNN은 1이상의 stride를 사용하는 층들 때문에, 점진적으로 위치 정보를 잃고, 따라서 위치 정보가 중요한

    Semantic segmentation에는 적합하지 않다.
  - 여기에서 FCN을 사용한다.
  - 우선, 기존의 CNN에서 적용하는 전체 스트라이드가 32개라면, 손실을 막기 위해 해상도를 32배만큼 늘리는 Upsampling Layer를 통해 손실을 막는다.
  - 업샘플링에는 **전치 합성곱 층(0으로 채워진 행과 열을 끼워 넣은 뒤, stride와 convolution을 통해 커진 이미지를 만든다.), 이중 선형 보간** 등이 있다.
  -