Convolutional Neural Network

Made by Jay Hong

Check

Convolutional Neural Network

1. CNN?
2. Training Technique?
3. CNN Architectures

1. CNN?

1. CNN x 생물학적 Motivation
2. CNN 기반의 Models
3. Convolution Layer 특징

생물학적 Motivation

- ✓ 인간은 점 -> 선 -> 면 -> 객체 순으로 인식함
- ✓ Model도 이처럼 특징을 합쳐나가면 인식에 도움이 될 것이라 판단.

History

- ✓ LeNet-5 : Conv NN 의 시작

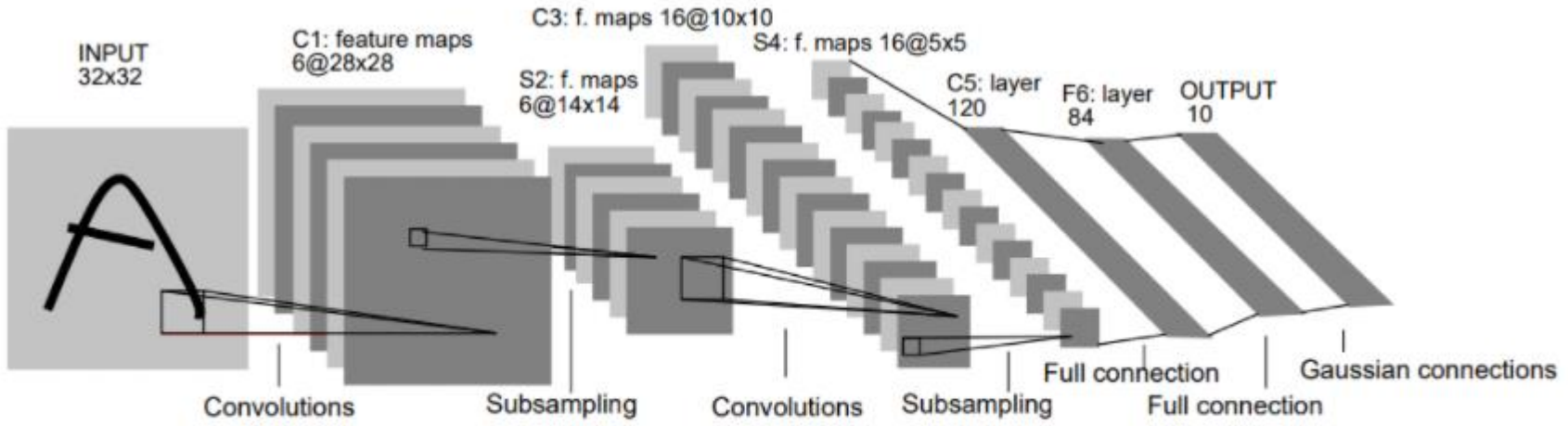


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

History

- ✓ AlexNet : 깊게 쌓기의 시작

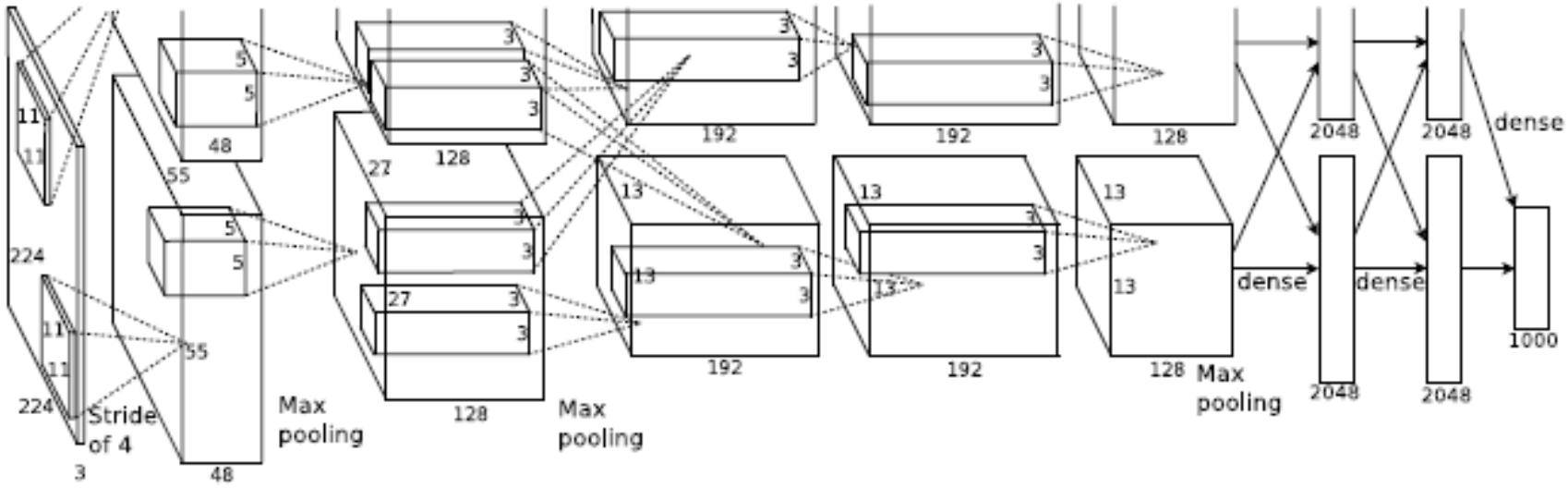


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

History

- ✓ Recent Models : FC 없애는 추세

- ✓ FC : Fully Connected Layer
  ⇒ 1Dim으로, 공간 정보가 사라져있음
  ⇒ 학습의 의미가 없어질 수 있다.

CNN의 Hyper Parameter

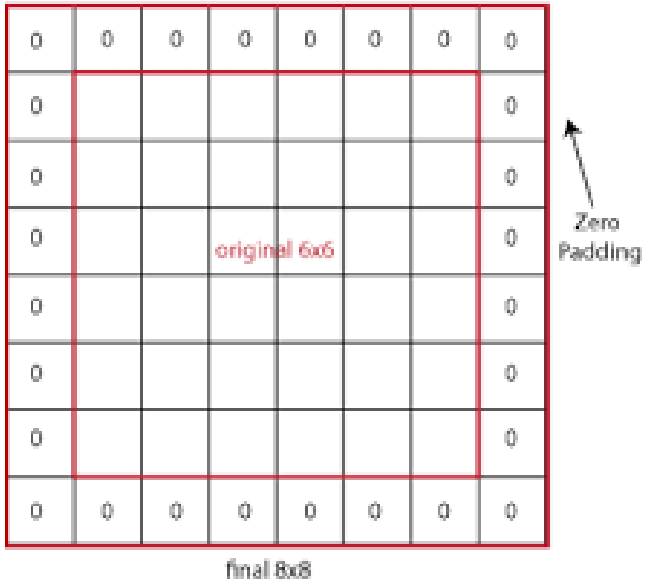
- ✓ Filter = Kernel
  - ✓ 정방행렬로, 위에서부터 훑는 창의 크기
  - ✓ 픽셀 단위로 옮겨가며,
    이미지 좌상단부터 우하단까지 slide over 하며 차례대로 훑음
  - ✓ 필터는 해당 지역에서 내적곱을 해주고, 맨 끝까지 이동
  - ✓ 해당 지역에서 Weight Matrix와 Bias를 학습
  - ✓ 필터와 이미지의 Channel size는 동일해야 함
  - ✓ Example)
    - ✓ 32x32x3 이미지에 5x5x3 필터 적용하면 28x28x1 Activation Map 나옴
    - ✓ 필터의 수를 늘려 Channel의 크기를 늘릴 수 있음
    - ✓ 필터의 수가 늘어도, 개별적으로 잡아내는 특징이 다 다름

CNN의 Hyper Parameter

- ✓ Stride
  - ✓ 입력 텐서와 곱하는 합성곱의 위치를 제어
  - ✓ 몇 pixel식 건너뛰며 볼 것인지에 대한 정의
  - ✓ Example)
    - ✓ 7x7x3에 3x3x3 필터
    - ✓ if Stride == 1 :
      - ✓ 1~3번, 2~4번, 3~5번, 4~6번, 5~7번을 조사
    - ✓ elif stride == 2 :
      - ✓ 1~3번, 3~5번, 5~7번을 조사
      - ⇒ 중복을 최소화하며 계산에 이점을 둠
  - ✓ 단, Stride를 적용했을 때, 정보의 손실이 발생하지 않는지 반드시 조사.
    - ⇒ Stride를 적용했을 때의 Output size “(N – F) / stride + 1”
      이 값이 정수가 아니면 정보의 손실이 있음

CNN의 Hyper Parameter

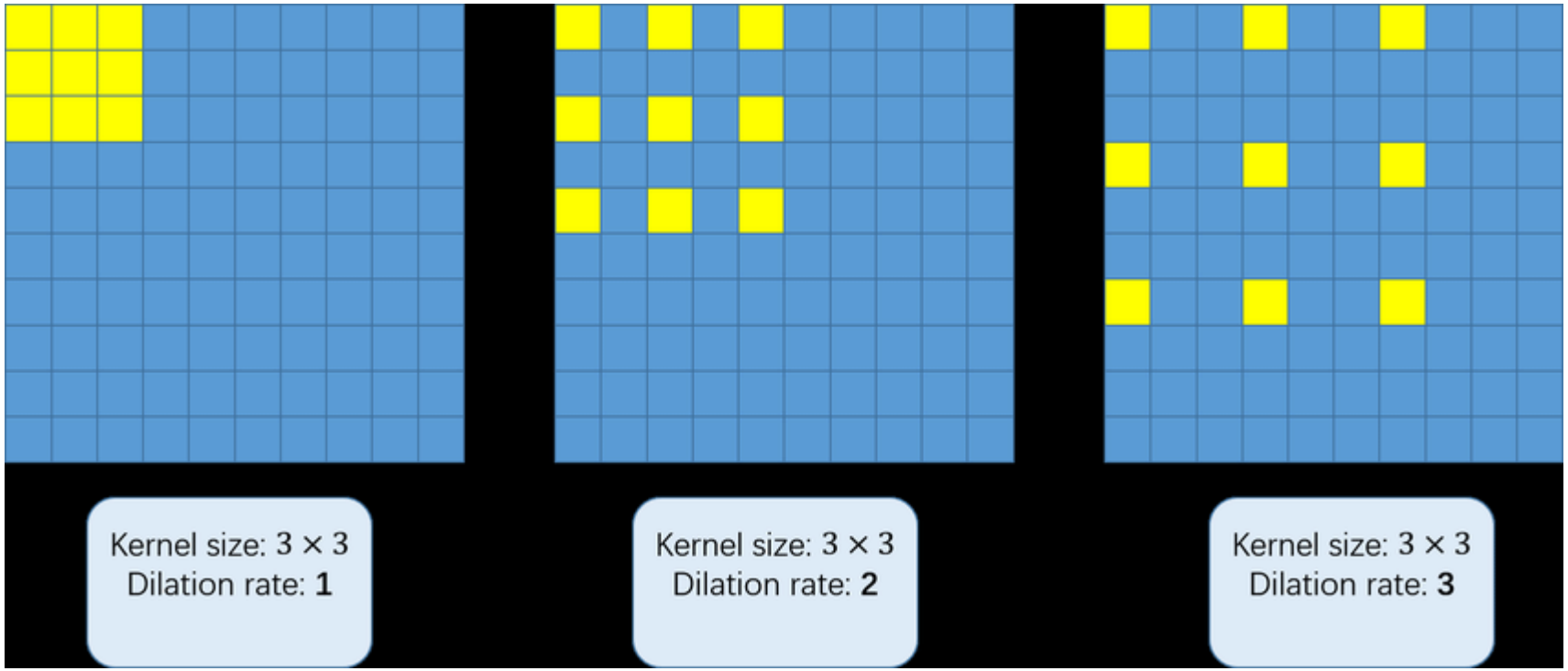
- ✓ Padding
  - ✓ 이미지의 공간 정보 형태를 그대로 유지하고 싶을 때, 바깥을 0으로 채움
    - ✓ Example)
      기존 7x7x3 -> 3x3x3 Filter => 5x5x1 Image
      기존 7x7x3 -> 3x3x3 Filter => 패딩으로 9x9x3으로 변경 => 7x7x1 Image



CNN의 Hyper Parameter

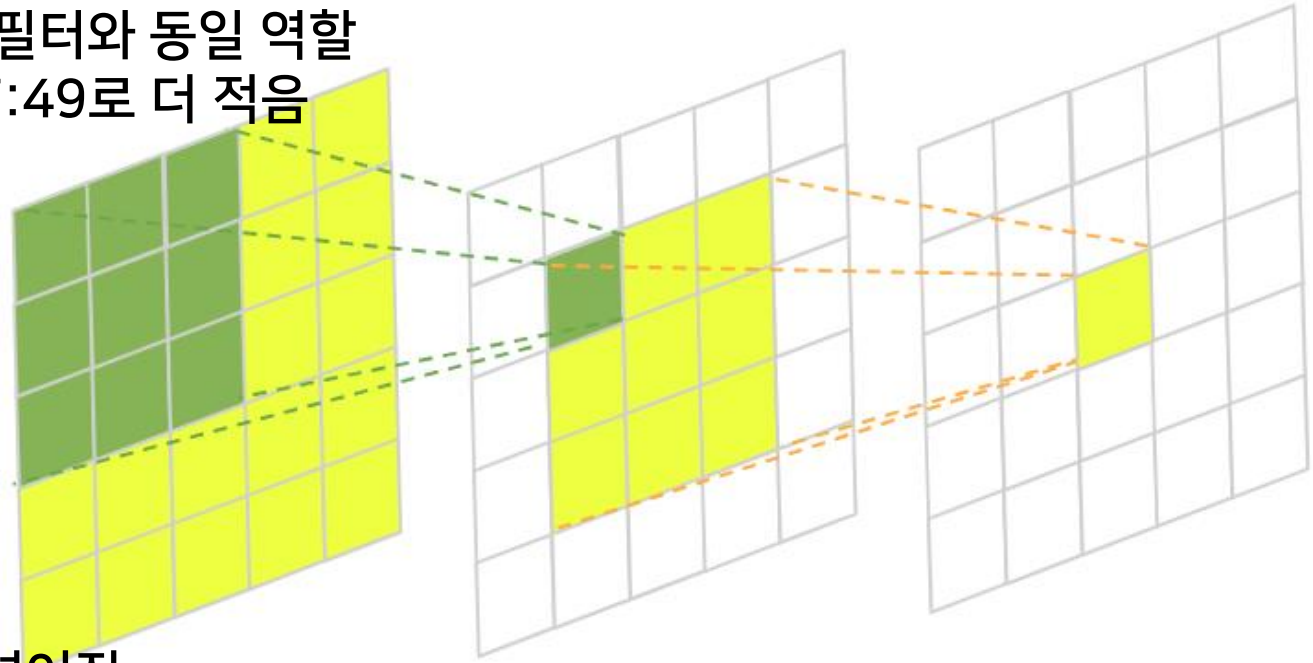
- ✓ dilation

  - ✓ 간격을 얼마나 띄워 곱할지를 제어함 => 넓은 공간을 요약해서 설명하는데 유용
  - ✓ NLP 분야에서는 자주 적용해서 사용
  - ✓ 단어와 단어 사이를 건너 띄면서 파악하기 위해서



CNN의 특징

- ✓ Receptive Field
  - ✓ CNN은 층을 깊게 쌓을 수록, 정보의 확장성이 있음
  - ✓ 이는 계산에 굉장한 이점이 됨
    - ✓ 3x3 필터 3개는 7x7 필터와 동일 역할
    - ✓ 하지만 파라미터는 27:49로 더 적음
  - ✓ 딥러닝의 측면에선,
    이미지의 특징을 점점
    더 크게 파악함을 의미
  - ✓ 인간의 점->선->면과 동일
  - ✓ dilation을 적용하면
    Receptive Filed는 훨씬 넓어짐



CNN의 특징

- ✓ 1x1 Convolution Layer
  - ✓ Channel의 관점에서 정보를 압축하는 방법
  - ✓ Fiber라고도 부름
  - ✓ Example)
    - ✓ 56x56x64 이미지 + 1x1x64 필터 32개
      -> 56x56x32로 Channel 압축 (padding='same')
  - ✓ 장점
    - ✓ Channel 수 조절 => 우리가 원하는 Size로 조정 가능
    - ✓ 연산량 감소 => Channel의 감소는 파라미터 수의 감소에 직접적 영향
    - ✓ 비선형성 추가 => 연산량이 감소하니, 더 깊게 쌓을 수 있어 비선형성 추가 가능

2. Training Technique?

1. Normalizations

Normalization Tips

- ✓ Dropout : p%의 hidden node를 0으로 만들어 연결을 끊는 방법
  - ✓ 학습 자체를 못하게 막는 Regularization 방법

- ✓ Normalization
  - ✓ N(0,1)을 따르게 만듦

- ✓ Distribution Shift
  - ✓ 학습한 데이터의 분포와 Test 할 데이터의 분포가 달라 발생할 수 있는 문제

- ✓ Internal Distribution Shift
  - ✓ Layer의 내부 Node에서 발생하는 Distribution Shift
  - ✓ DL에서는, 이전 Layer의 Output이 현재 Layer의 Input 이 됨
  - ✓ Output의 분포가 다양해짐
  - ⇒ 학습속도와 학습 성능 자체가 안좋아지는 경향이 있음
  - ⇒ 쉽게 설명해서 , 10층 NN에 Output Distribution이 0.1씩 차이 나도, 최종 layer에서는 (1.1)^10 = 2.59가 되어 분포가 너무 많이 차이나 데이터 분포를 파악하기 너무 어려움

Normalization Tips

- ✓ Batch Normalization
  - ✓ Internal Distribution Shift를 해결하기위해 2016년에 공개된 기법
  - ✓ Batch 단위로 Normalization을 적용하자는 Idea
  - ⇒ 데이터의 분포가 동일해지니 학습 속도와 성능 자체에서 좋은 성능을 보임
  - ⇒ 원본 데이터의 특징을 살리기 위해, gamma와 beta도 사용해서 조금은 복원시켜줌
  - ✓ Test 예측을 위해 Train 데이터의 이동평균을 적용하며 Normalization record를 저장해둠

- ✓ Layer Normalization
  - ✓ batch norm도 좋지만, Batch Norm의 단점이 너무 많았음
    - ✓ RNN에 적용하기 어려움
    - ✓ Test 예측이 불편함
  - ✓ 그냥 전체 데이터를 이용해서 Normalize하자.
    - ✓ train과 test에 동일하게 적용 가능

- ✓ Instance Normalization
  - ✓ 공간 정보를 압축하자는 개념
  - ✓ 이미지의 HxW만 이용해서 Normalize하자는 개념 => Style GAN에서 자주 사용

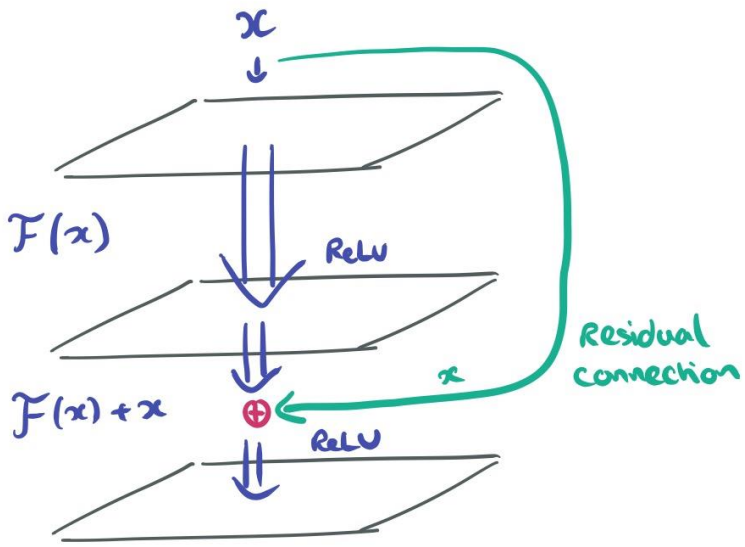
3. CNN Architectures

1. Residual Connection
2. ResNet Models

Residual Connections?

- ✓ 몇 개의 구조를 Skip하며 현재의 정보를 전달해줌

$$h^{(k+1)} = f(W h^{(k)} + b) + h^{(k)}$$



- ✓ 위의 수식을 만족할 수 있게 됨

- ✓ 필요성
  - ✓ W에서 아무것도 배우기 싫다면, W가 Identity Matrix가 되어야 함
    => 그러면 W가 그대로 전파됨
  - ✓ 하지만 Identity Matrix가 되면, 이전까지의 정보가 사라져버리게 됨
  - ✓ 따라서 억지로라도 정보를 담을 수 밖에 없음
  - ✓ 하지만, Residual Connection을 하면, W를 Identity Matrix, b를 0으로 설정할 수 있음

- ✓ 특징
  - ✓ Channel이 달라질 때 Residual Connection을 하기 어려우니 잘 살펴봐야 함
  - ✓ 모델이 깊어져도 학습하기에 좋음

ResNet 18

```
def resnet18():
    block = ResidualBlock
    model = ResNet(3, block)
    return model
```