5️⃣ Alexnet

정의

- ILSVRC 의 2012년 대회에서 top-5 test error(모델이 예측한 최상위 5개 범주 가운데 정답이 없는 경우의 오류율)가 15.4% 로 1위를 차지하고 주목을 받았다.

구조

- 256X256X3 의 이미지를 받아서 총 1000개의 클래스로 구분을 해야 하는 모델이므로 LeNet보다 훨씬 복잡한 구성을 갖는다.
- 총 2개의 분기로 구성되어있으며, 두 분기의 네트워크는 서로 다른 특징을 학습

특징

- 기존 LeNet에 비해 훨씬 복잡한 구조
- Max-pooling layer 사용 (3X3 overlapping maxpooling)
  - stride < kernel size (kernel 중첩됨)
- Local response normalization 사용 (최근에는 잘 사용하지 않음)
- Data augementation : 과적합 줄이고 데이터 양 늘림
  - image translation, horizontal reflectionn, REB channel값 변경

  쉽게 말해서 하나의 이미지를 가지고 여러 장의 비슷한 이미지를 만들어 내는 것이다. 이미지를 좌우 반전시키거나, AlexNet이 허용하는 입력 이미지 크기인 227 x 227 × 3보다 좀 더 큰 이미지를 조금씩 다르게 잘라서 227 x 227 × 3으로 만들어줘서 여러 장을 만들어낸다. 같은 내용을 담고 있지만 위치가 살짝 다른 이미지들이 생산된다. 예제로 가져온 그림을 보면 원본 이미지를 좌우 대칭 시키거나(mirror image), 끝 부분을 조금씩 자르는 등의 하나의 이미지로 여러 개의 서로 다른 비슷한 이미지들을 만들어 낼 수가 있다.
- GPU 학습에 사용
  - 더 큰 네트워크를 위해 2개의 GPU를 이용해 병렬 연산 수행
  - 3번째 convolution layer와 fully connected layer를 제외하면 독립적으로 훈련을 진행
  - training error를 1~2% 줄일 수 있었고, 학습 시간도 더 빨라짐

  → 딥러닝 모델에서 GPU를 이용해 학습하는 것이 중요
- Dropout 적용
  - 히든 레이어의 특정 뉴런의 출력을 일정 확률로 0으로 만드는 것
  - 하나의 뉴런의 값에 크게 의존하지 않도록 모델이 학습
  - train때만 사용

  → overfitting (과적합)을 막는 중요한 방법 중 하나
- ReLU 활성화 함수 사용
  - 기존 sigmoid에서는 기울기(미분값이) 0.25이므로 chain rule에 의해서 계속 0.25씩 곱하다보면 네트워크의 초반쪽에서는 값이 0으로 수렴해가고, 이렇게하면 학습이 진행이 안됨.
- Weight decay
  - train model의 복잡도(클수록 과적합할 가능성도 큼)를 줄이기 위해 학습 중 weight가 너무 큰 값을 가지지 않도록 하는 것
  - Loss function에 weight를 높임 → penalty 부여
  - L2 regulrization을 사용
  - 이 방법 또한 과적합을 막아주는 효과가 있음
- 가중치 초기화
  - 평균 0, 표준편차 0.01의 정규분포로 가중치를 초기화
  - 2,4,5번째 convolution layer와 fully-connected layer에 한해 1로 초기화

- 학습률(learning layer)은 0.01로 시작해 오차가 더 줄지 않으면 10으로 나누는 식으로 0.01부터 총 3번 나눔

[논문 리뷰] Alexnet Pytorch 구현

The Dataset

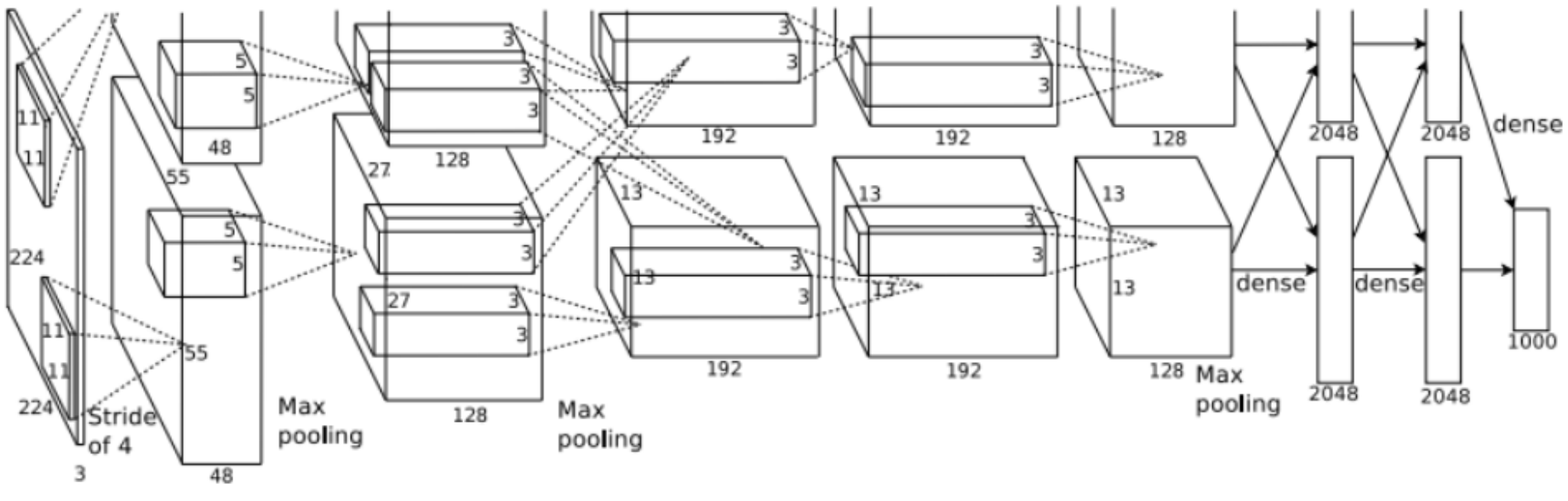
1. 사용한 data

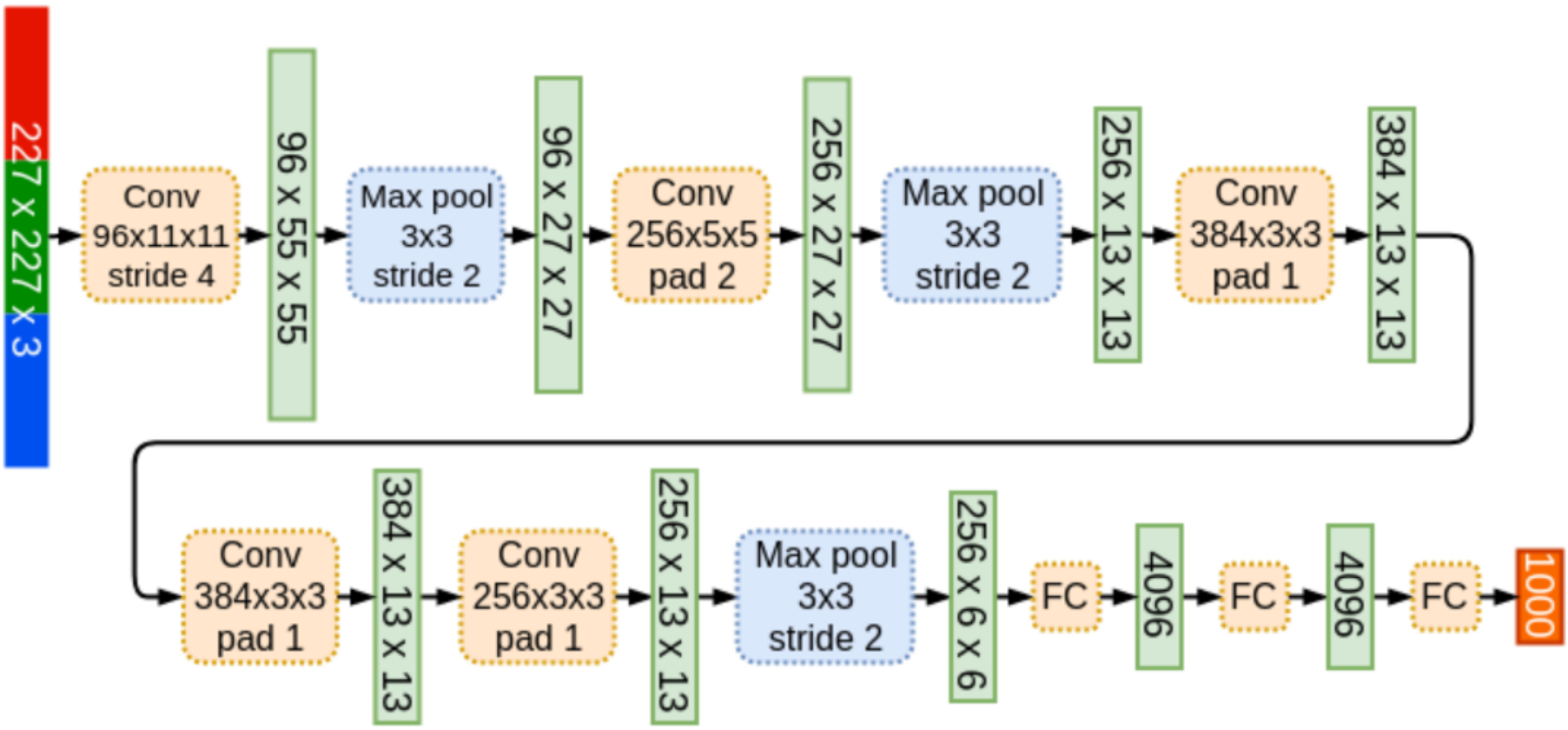
- Imagenet dataset
  - Imagenet dataset의 subset 사용
  - 120만개의 training images + 50000개 validation images + 150000개 testing image
  - 우리는 1개의 클래스당 50개가 있는 1000개의 클래스가 있는 데이터(validation data)로만 사용

2. Data 전처리

- image size 256X256 으로 고정
  - 이미지를 동일한 크기(256X256)으로 고정시켜줌
  - 나중에 FC layer의 입력 크기가 들어있어야함.
  - 입력 이미지 크기가 다르다면 FC layer에 입력되는 feature의 수가 다르게 됨.
  - Resize 방법 : 이미지의 넓이와 높이 중 더 짧은 쪽을 256으로 고정한 후 중간 부분을 256X256 크기로 crop해줌.
- 각 image의 pixel에 training set의 평균을 빼서 normalize 시켜줌

Alexnet Architecture





AlexNet의 아키텍처

- 구성
  - [Input layer → Conv1 → MaxPool1 → Norm1 → Conv2 → MaxPool2 → Norm2 → Conv3 → Conv4 → Conv5 → Maxpool3 → FC1 → FC2 → Output lauer]
- Input layer
  - 224X224X3 크기의 이미지 → 227X227X3
- Conv1
  - 11X11 사이즈의 96개의 kernel
  - stride(폭) = 4
  - padding = 0 (zero padding)
  - Input : 224X224X3
  - Output : 55X55X96
- MaxPool1
  - 3X3 size의 kernel
  - stride=4
  - Input : 55X55X96
  - Output : 27X27X96
- Norm1
  - LRN을 사용한 normalization layer
  - Input : 27X27X96
  - Output : 27X27X96

  → 사이즈 변화 없음
- Conv2
  - 5X5 size의 kernel 256개
  - stride=1
  - padding=2
  - Input : 27X27X96

    - Output : 13X13X256
- MaxPool2
    - 3X3 kernel
    - stride=2
    - Input : 27X27X256
    - Output : 13X13X256
- Norm2
    - LRN을 사용한 normalization layer
    - Input : 13X13X256
    - Output : 13X13X256
- Conv3
    - 3X3 size인 kernel 384개
    - stride=1
    - padding=1
    - Input : 13X13X256
    - Output : 13X13X384
- Conv4
    - 3X3 size의 kernel 384개
    - stride=1
    - padding=1
    - Input : 13X13X384
    - Output : 13X13X256
- Conv5
    - 3X3 size의 kernel 256개
    - stride=1
    - padding=1
    - Input : 13X13X384
    - Output : 13X13X256
- MaxPool3
    - 3X3 kernel
    - stride=2
    - Input : 13X13X256
    - Output : 6X6X256
- FC1
    - fully connected layer with 4096 neurons
    - Input : 6X6X256
    - Output : 4096
- FC2
    - fully connected layer with 4096 neurons
    - Input : 4096
    - Output : 4096

- Output layer
  - fully connected layer with 1000-way softmax
  - Input : 4096
  - Output : 1000

Alexnet 구조에 적용된 특징

1. ReLU Nonlinearity

- 활성화 함수로 ReLU 적용
- saturating nonlinearity(tanh, sigmoid)보다 non-saturating nonlinearity (ReLU)의 학습 속도가 몇배는 빠르다
- ReLU와 tanh의 epoch의 실험결과 비교

2. Training on Multiple GPUs

- network를 2개의 GPU로 나누어서 학습
- ex) 90개의 kernel이면 45개는 GPU1, 45개는 GPU2
- 3번째 conv layer에서만 GPU 통합시킴

3. Local Response Normalization(LRN)

- generalization이 목적
- ReLU는 non-saturating nonlinerarity 함수이기 때문에 saturating을 예방하기 위한 입력 normalization이 필요로 하지 않는 성질을 갖고 있음
- LRN을 측면 억제의 형태로 구현된다 (측면억제는 강한 자극이 주변의 약한 자극을 전달하는 것을 막는 효과)

$$b^i_{x,y} = a^i_{x,y} / (k + \alpha \sum_{j=max(0,i-n/2)}^{j=min(N-1,i+n/2)} {a^j_{x,y}}^2)^\beta$$

where
$b^i_{x,y}$ – regularized output for kernel i at position x, y
$a^i_{x,y}$ – source ouput of kernel i applied at position x, y
N – total number of kernels
n – size of the normalization neigbourhood
$\alpha, \beta, k, (n)$ – hyperparameters

$$b^i_{x,y} = \frac{a^i_{x,y}}{\left(k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} (a^j_{x,y})^2\right)^\beta}$$

- a는 x,y 위치에 적용된 i번째 kernel의 output을 의미하고 이 a를 normalization하여 큰 값이 주변의 약한 값에 영향을 주는 것을 최소화 했다
- AlexNet 이후 현대의 CNN에서는 local response normalization 대신 batch normalization 기법이 쓰인다

4. Overlapping Pooling

- pooling layer는 동일한 kernel map에 있는 인접한 neuron의 output을 요약
- alexnet은 overlap을 해줌
- kernel size=3

- stride=2

Reducing Overfitting

- alexnet에는 6천만개의 parameter존재 → 이미지를 1000개의 classess로 분류하기 위해서는 상당한 overfitting 없이 수많은 parameters를 학습시키는 것은 어렵다.

1. Data Augmentation

- 현재 갖고 있는 데이터를 좀 더 다양하게 만들어 CNN 모델을 학습시키기 위해 만들어진 개념
- overfitting 방비시켜줌
- 연산량이 매우 적고 CPU에서 이루어짐

i) generating image translation and horizontal reflections

- 256X256 image → 224X224 size로 crop
- crop 위치는 중앙, 좌측 상단, 좌측 하단, 우측 상단, 우측 하단 이렇게 5개의 위치에서 crop
- crop으로 생성된 5개의 이미지를 horizontal reflection

ii) altering the intensities of the RGB channels in training images

- image의 RGB pixel의 값에 변화
- imagenet의 training set에 RGB pixel 값에 대한 PCA를 적용
- PCA를 수행하여 RGB 각 색상에 대한 eigenvalue를 찾고 이 value와 평균 0, 분산 0.1인 가우시안 분포에서 추출한 랜덤 변수를 곱해서 RGB 값에 더해줌

$$[\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3][\alpha_1\lambda_1, \alpha_2\lambda_2, \alpha_3\lambda_3]^T$$

→ 조명의 영향과 색의 intensity 변화에 대한 불변성을 지님

2. Dropout

- 서로 다른 모델의 예측을 결합하는 앙상블 기법은 test error 감소에 효과적 → but, alexnet은 학습시키는데 몇일이 걸림 → 모델 결합에 효과적인 버전인 dropout 적용
- dropout 0.5로 설정 → dropout된 neuron은 순전파와 역전파에 영향을 주지 않음 → 가중치는 공유되지만 신경망은 서로 다른 구조를 띈다.
- neuron은 특정 다른 neuron의 존재에 의존하지 않기 때문에 이 기법은 복잡한 neuron의 co-adaptation를 감소시킴
- 서로 다른 neuron의 임의의 부분 집합끼리 결합에 유용한 robust 특징을 배울 수 있음
- train에 dropout적용 → test에는 모든 neuron 사용했지만 neuron결과값에 0.5 곱해줌
- 2개의 FC layer에만 dropout 적용

Details of learning

-학습시킬 때 설정하였던 hyper parameter

- momentum : 0.9
- batch size : 128
- weight decay : 0.005
- SGD(stochastic gradient descent) 이용
- weight decay는 training error 낮춤

$$\begin{aligned} v_{i+1} &:= 0.9 \cdot v_i - 0.0005 \cdot \epsilon \cdot w_i - \epsilon \cdot \left\langle \frac{\partial L}{\partial w}\Big|_{w_i} \right\rangle_{D_i} \\ w_{i+1} &:= w_i + v_{i+1} \end{aligned}$$

→ 평균 0, 분산 0.01인 가우시안 분포 이용

- bias 초기화 : Conv2,4,5와 FC layer =1, 나머지 layer는 0으로 초기화
- learning rate = 0.01 (validation error가 상향되지 않으면 10으로 나누어줌) → 학습 종료전 3번의 learning rate 감소

데이터 전처리(사이즈 변경, 데이터 증강)

data loader