인공지능2 과제1 보고서

2020035805 엄태곤

제가 이번 AlexNet구현에 사용한 데이터는 Fashion MNIST데이터입니다. 이 데이터는 열개의 클래스를 가지고 있고, train에 6만개, test에 1만개 데이터를 가지고 있습니다. .csv 확장자로 파일이 있으며 각 행의 첫번째 열에는 라벨이 있고, 나머지 열에는 픽셀데이터로 이루어져 있습니다. 이미지로 변환하면 흑백 이미지가 나옵니다. 이미지 원본의 크기는 28x28입니다.

코드설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

* 필요한 패키지들을 import하는 과정입니다. 사전에 venv를 만들어서 콘솔에서 pip로 필요한 패키지들을 미리 설치해둡니다.

텍스트, 라인, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* cuda코어가 사용 가능하면 사용할 수 있도록 미리 변수에 저장해 둡니다. 사용할 수 없으면 cpu로 반환하게 되어있습니다. pytorch버전도 출력해줍니다.

텍스트, 폰트, 소프트웨어, 스크린샷이(가) 표시된 사진

자동 생성된 설명

csv파일을 image로 바꿀 때 필요한 transform객체의 값을 미리 정의하여 transform이라는 변수에 미리 저장해 둡니다. 28x28 이미지를 256x256으로 키우고, 다시 그 이미지 안에서 무작위로 227x227 크기의 이미지를 크롭해서 사용할 것이고, 데이터를 tensor로 바꾼 뒤 마지막으로 평균Mean은 0.1307, 표준편차std는 0.3081로 해서 각 채널별(흑백이니 1개) 이미지 픽셀 값에서 평균값을 뺀 뒤, 표준편차로 나누어서 정규화를 진행합니다.텍스트, 폰트, 라인, 번호이(가) 표시된 사진

자동 생성된 설명

csv파일을 pandas로 읽어옵니다. class\_names에 label에 번호별로 이름을 저장해둡니다.

6만개의 train data중에서 8천개만 학습에 사용할 수 있도록 여기서 조정해줍니다.

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

csv파일을 PIL image형태로 변환할 때 사용하는 class를 정의해 줍니다.

텍스트, 라인, 폰트, 그래프이(가) 표시된 사진

자동 생성된 설명

Dataloader객체에 이미지를 담아서 train/test용으로 변수에 담아둡니다.

Batch\_size와 Shuffle 은 각각 64, true로 설정해서 무작위 추출을 하고 batch size를 64로 지정해줍니다.

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명AlexNet을 class에 정의해줍니다.

텍스트, 폰트, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명

Model에 alexnet class인스턴스를 담아주고 optimizer는 adam으로 설정합니다. Loss function은 crossentropyloss로 설정했습니다.

텍스트, 스크린샷, 메뉴, 번호이(가) 표시된 사진

자동 생성된 설명

* 최종 레이어 구조를 출력했습니다.

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명train 함수를 구현해줍니다.

Dataloader의 객체를 순차적으로 꺼내서 forwardpass/backpropagation을 수행합니다.

마지막으로 accuracy와 loss값을 학습중에 터미널에 출력해줍니다.

텍스트, 스크린샷, 디스플레이, 폰트이(가) 표시된 사진

자동 생성된 설명test함수를 구현해 줍니다. dataloader에 있는 test데이터셋을 이용해 모델을 실행하면서 손실률, 정확도 등을 터미널에 출력해줍니다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

마지막으로 epoch(한 데이터셋을 한번 다 학습하는 단위) 를 30으로 설정해주고 위에서 선언한 train/test 함수들을 적절히 배치해서 실행하고 마지막으로 정확도와 손실률을 출력해줍니다.

학습에는 84분이 걸렸고 결과는 이렇게 나왔습니다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

배운점

AlexNet을 실제로 구현해 보면서 많은 정보들을 찾아보았고, 그 과정에서 많은 것들을 배울 수 있었습니다. 이미지 크기를 layer크기에 맞추어 전처리를 해줘야 한다는 것을 배웠고 기존에 사용하던 activation function들(sigmoid,tanh) 대신 relu를 사용하여 vanishing gradient 문제를 해결하여 학습속도를 개선시킬 수 있다는 것을 배웠습니다. 입력값이 음수일 경우엔 leaky relu로 해결할 수 있다는 것도 배웠습니다. 이번엔 병렬 gpu를 구현해보진 못했지만 당시 하드웨어의 한계를 병렬 gpu로 해결했다는 것도 배울 수 있었습니다. 입력에 이미 activation function으로 relu가 적용되어 있어 입력에 대해 normalization을 수행하지 않아도 되지만 Local Response Normalization을 사용했을 때 에러 비율이 일부 감소했다고 하여 저도 코드에 추가했습니다. (찾아보니까 흑백 이미지에서는 효과가 없다는 의견을 찾을 수 있었습니다.) Overlapping pooling을 사용하여(일반적으로 pooling을할 때 중복되지 않게 하지만 중복되게 함) 에러비율과 과적합을 낮출 수 있음을 배웠습니다. 과적합 문제를 해결하기 위해 Data augmentation이라는 기법을 배울 수 있었습니다. 저는 Random crop을 사용해보았고 추가로 Horizontal reflection,(좌우반전) Altering the intensities of RGB(RGB색상채널별로 변환) 기법도 있음을 배웠습니다. Dropout이라는 개념도 잘 배울 수 있었습니다. 저는 50퍼센트 확률로 뉴런을 제거하는 것으로 구현해봤습니다. 다음엔 학습이 끝나기 전에 결과를 .pth파일로 저장해서 관리할 수 있으면 좋을 것 같습니다.

느낀점

제가 amd그래픽카드가 아니고 nvidia사의 그래픽카드를 가지고 있었다면 더 빨리 학습을 끝낼 수 있었을 것 같습니다. 과제 시작하기 전에 윈도우에서 7900XTX를 연산자원으로 쓸 수 있는 방법을 오래 찾아봤지만 결국 찾을 수 없었습니다. Pytorch에서 rocm으로 지원하는 버전을 아직 공개하지 않은 것 같습니다.