REPORT

DLA Midterm Exam Project

학 과 : 산업경영공학부
교수님 : 김대환
학 번 : 20181794
이 름 : 김창영
제출일 : 2023.11.03

1. 개발환경

본 프로젝트의 개발환경은 [OS] Ubuntu20.04, [GPU] CUDA 11.4, NVIDA RTX A6000 1장이며, 효율적인 자원관리와 재현성을 위해 Docker를 사용하여 컨테이너 환경을 구축하였다. 컨테이너 이미지는 Docker hub의 Pytorch:2.0.1-cuda11.7-cudnn8-devel를 빌드하였다.

2. 강의자료 모델 실험 계획 및 결과

2.1 실험 계획

본 실험에서는 일반화 검증을 진행하지 않는다. 즉 일반적인 실험이라면 Train, Validation, Test로 데이터 셋을 분리하여 Train과 Validation으로 학습을 진행하고 Testset으로 일반화 검증을 진행하지만, 이번 실험에서는 일반화 검증을 하지 않기 때문에 Train과 Test(Validation)으로 분리하여 실험을 계획하였다. 재현성을 위해 랜덤시드는 23으로 고정하였다. 강의자료의 코드를 활용하여 모델을 구성하였으며, 각 모델의 성능을 비교하기 위해 데이터 전처리는 정규화만 진행하였으며, 각 실험의 배치크기는 64, 에포크는 100으로 통일하였다. 또한 옵티마이저로는 SGD를 적용하였으며, learning rate은 0.1, momentum은 0.9, weight decay는 0.0005를 주었다.

2.2. 실험 결과

2.2.1. DNN

DNN의 경우 최고 성능이 에포크 17일 때 정확도 52.64%와 1.3459의 오차를 보였다. Fig 1의 (A)를 보면 에포크 17이후 발산하는 모습을 보이면 저조한 성능을 보였다.

2.2.2. CNN

Fig 1의 (B)를 보면 학습 오차와 테스트 오차가 시간이 지날수록 감소하는 것을 통해 학습이 잘 진행되었음을 확인할 수 있으며, 에포크 67일 때 정확도 77.14%와 0.6787의 오차를 보였다.

2.2.3. ResNet

에포크 11일 때 정확도가 80.85%와 0.5801의 오차를 보이면 모델중 가장 좋은 성능을 보였지만, 에포크 11이후 테스트 오차가 발산하며 과적합 양상을 보였다.

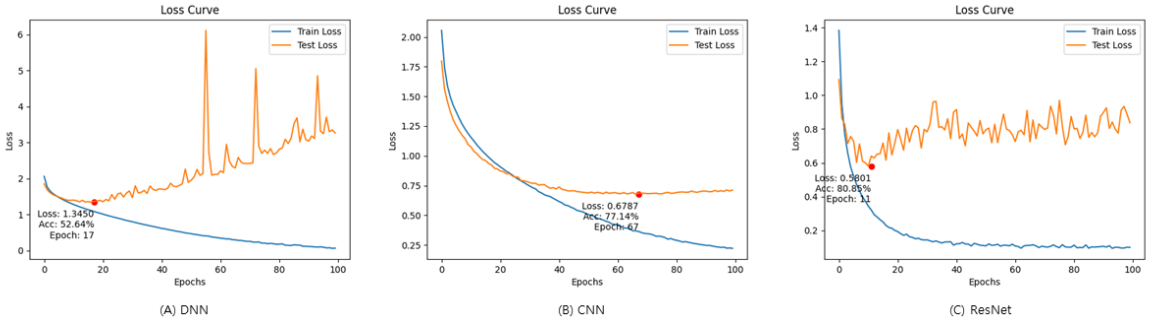

Fig 1. 모델별 Loss Curve

3. Custom ResNet

3.1. 모델 아키텍처

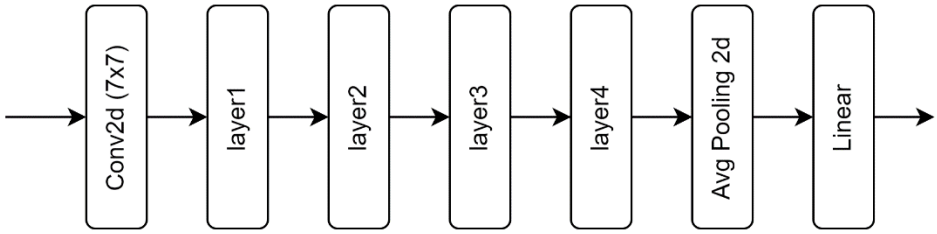

Fig 2. Custom ResNet 구조

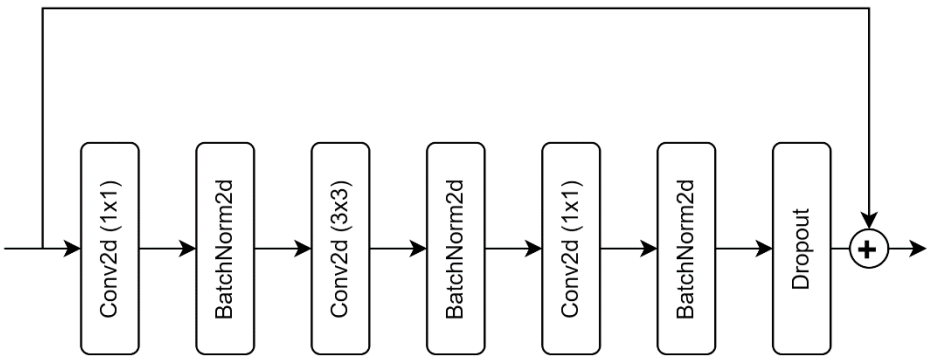

Fig 3. Bottleneck Block 구조

강의자료의 ResNet 구조에서 Block 구조를 가지는 layer를 1층 더 추가하여 Fig2와 같이 설계하였다. 또한 Basic Block을 Fig 3과 같은 Bottleneck Block구조를 변경하였다.

Bottleneck Block의 구조는 기존 Basic Block보다 1개 convolution layer가 추가되었으며, convolution layer(1x1)을 2개 사용하여 입출력의 크기 변화 없이 파라미터 수를 감소시켜 연산량을 줄인다. 또한 layer가 증가함에 따라 활성화 함수가 더 증가하기 때문에 기존 대비 비선형성이 증가하였으며, 입력을 기존보다 더 다양하게 가공할 수 있게 설계되어있다. Convolution layer 다음

에는 BatchNormalize layer를 넣어 모델 내부의 그라디언트 소실 및 그라디언트 폭주 문제를 완화하였다. 또한 Block 끝단에는 Dropout layer를 추가하였다. Dropout을 통해 과적합을 방지하였으며, 입력의 민감도를 줄여 안정적으로 예측을 수행할 수 있도록 하였다.

다시 Custom ResNet 구조로 돌아오면, 32x32 이미지가 입력으로 들어와 7x7 커널 크기의 convolution layer를 거쳐 16x16으로 변하며 3채널에서 64개의 채널로 변한다. Layer1은 3개의 BottleneckBlock으로 구성되며 이미지 크기와 채널 변화는 없다. Layer2는 4개의 BottleneckBlock으로 구성되며 첫번째 Block에서 stride가 2이기 때문에 8x8로 크기가 감소하며 그 이후 Block에서는 크기 변화는 없으며 해당 층을 통과하여 64채널에서 128채널로 변화한다. Layer3는 6개의 BottleneckBlock으로 구성되며 첫번째 Block에서는 stride가 2이기 때문에 4x4로 크기가 줄어든다. 해당 층을 통과하여 128채널에서 256채널로 변화한다. Layer4는 3개의 BottleneckBlock으로 구성되며 첫번째 Block은 stride가 2이기 때문에 2x2로 감사한다. 해당 층을 통과하면 256채널에서 512채널로 변화한다. Avg Pooling층을 통해 크기는 1x1으로 변하고 최종적으로 Linear층을 거쳐 512채널이 일자로 펼쳐지고 클래스의 수인 10으로 마지막 노드가 형성된다.

3.2. 실험 계획

데이터 전처리는 정규화를 진행하였으며, 랜덤 크롭과 랜덤 수평 플립으로 데이터를 증강하였다. 옵티마이저는 기존 SGD에서 Adam으로 바꾸어 사용하였다. 배치크기는 64, 에포크는 300을 주었으며, 과적합과 학습이 길어지는 것을 방지하기 위해 조기 학습 종료 기능을 추가하였다. Patience를 15로 주어 에폭 15기간동안 Loss 개선이 없다면 학습을 종료하였다. 또한 learning rate은 0.001, weight decay는 0.0005를 주었다. 또한 스케줄러를 사용하여 30에폭마다 learning rate를 변경하도록 실험을 설계하였다.

3.3. 실험 결과

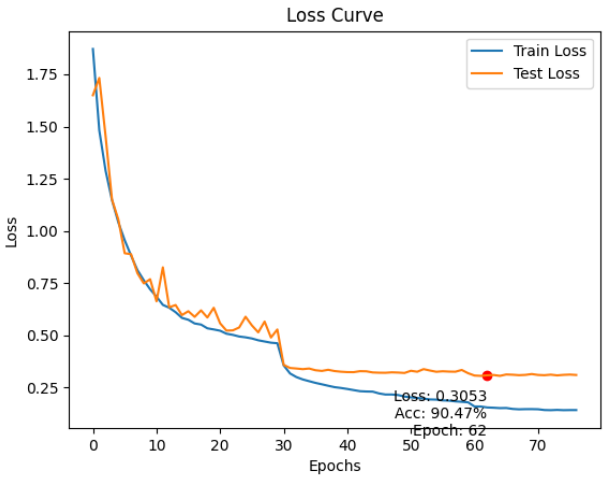

Fig 4. Custom ResNet Loss Curve

Fig 4를 보면 Train Loss와 Test Loss가 동시에 잘 감소하는 것을 통해 학습이 성공적으로 마쳤음을 확인할 수 있다. 에포크 62일 때 Loss가 0.3053으로 가장 낮았으며, 정확도는 90.47%로 가장 높았음을 확인하였다.

5. 결과 분석

Table 1. 최종 실험 결과

Model	Best Test Loss	Best Accuracy
DNN	1.3459	52.64%
CNN	0.6787	77.17%
ResNet	0.5801	80.85%
Custom ResNet	0.3053	90.47%

본 프로젝트에서는 CIFAR10 데이터셋을 활용하여 DNN, CNN, ResNet, Custom ResNet 총 4개의 모델에 대한 실험을 진행하였다. DNN보다는 Convolution 연산이 포함된 CNN의 성능이 좋았으며, CNN보다는 skip connection 개념이 도입된 ResNet의 성능이 더 좋게 나온 것을 확인할 수 있었다. 또한 최종적으로 Custom ResNet에서는 BatchNormalize Layer와 Dropout Layer 추가, Basic Block을 Bottleneck Block으로 변형하고 데이터 증강 및 옵티마이저를 SGD에서 Adam으로 변경하여 성능 개선을 하였다.