머신러닝

[HW03 – Implementation of ResNet using Pytorch]

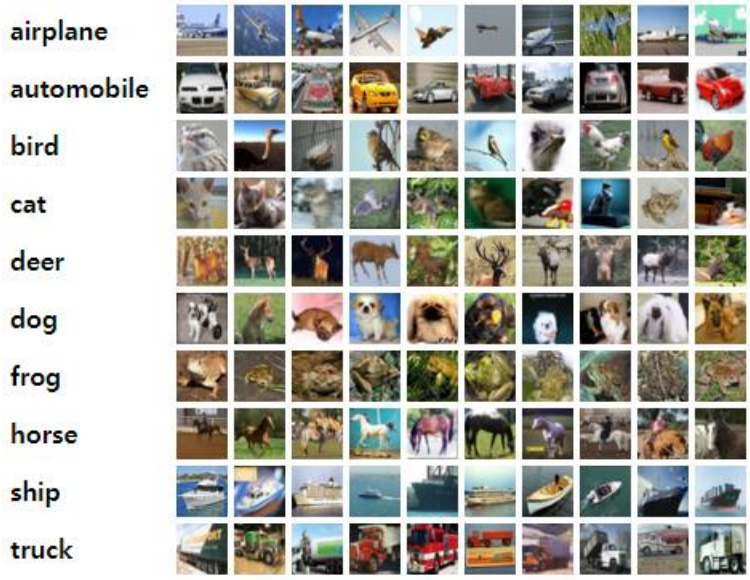
담당교수 : 이혁준 교수님

학 번 : 2021202058

성 명 : 송채영

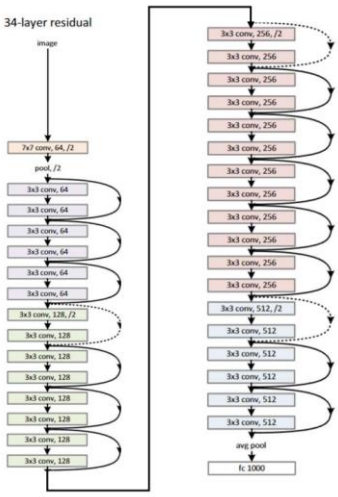
[Data]

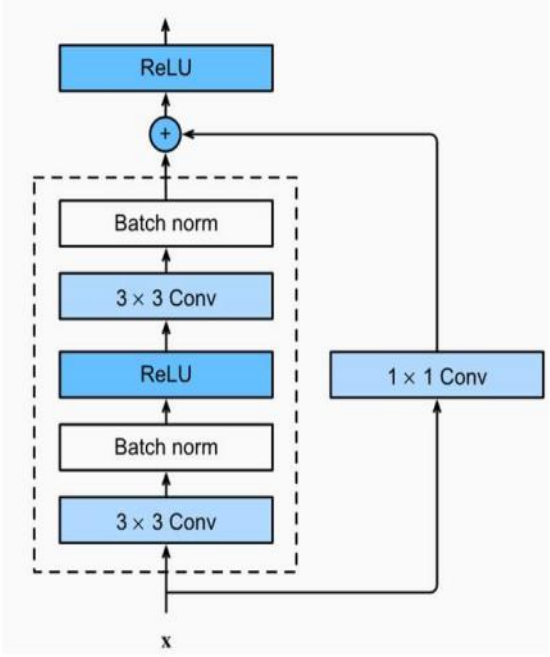
이번 프로젝트에서의 dataset은 CIFAR-10을 사용한다. 32*32 픽셀의 컬러 이미지로 train data 50,000개와 test data 10,000개로 구성되어 있고, 10개의 class(airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck)로 labeling 되어 있다.



[Network Architecture]

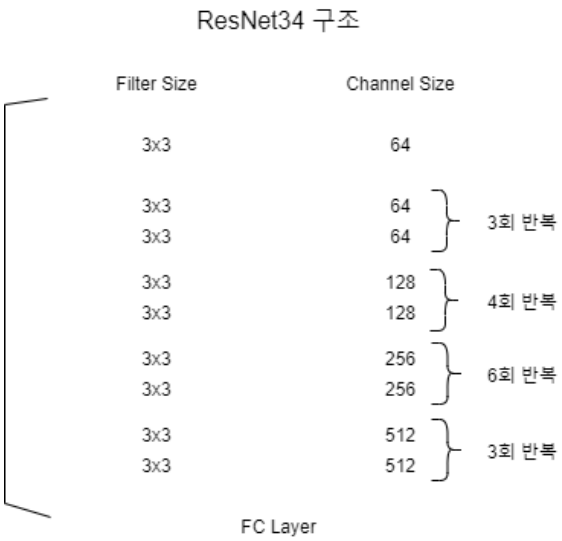
ResNet은 Residual Network의 줄임말로 딥러닝 모델의 한 종류이다. ResNet의 핵심은 Residual Block 방식이며 기존의 신경망은 입력 값 x를 타겟 값 y로 mapping 하는 함수 H(x)를 얻는 것이 목적이었다면, Residual Block 방식은 F(x) + x를 최소화하는 것을 목적으로 한다. Resnet 18, Resnet34, Resnet50, Resnet101, Resnet152가 있지만 이번 프로젝트에서는 그 중 Resnet34를 구현하였다.





ResNetBlock은 왼쪽의 강의 자료 그림을 참고하여 구현하였다. 두 개의 Convolution Layer와 Batch Normalization Layer로 구성하였으며 conv1, bn1, relu, conv2, bn2의 순서로 forward에 필요한 layer를 정의해주었다. stride가 1이 아니거나 입력 채널 수와 출력 채널 수가 다를 경우 downsampling을 하였다. Downsampling을 해야 하는 이유는 Featuremap의 형상이 축소되는 지점에서 pooling이 되기 때문에 형상을 맞춰주기 위해 필요하다. 또한 코드에서 out += identity로 구현한 부분은 Identity Mapping에 해당한다. Identity Mapping은 입력으로 들어간 값 x가 어떠한 함수를 통과해도 x가 나와야 한다.

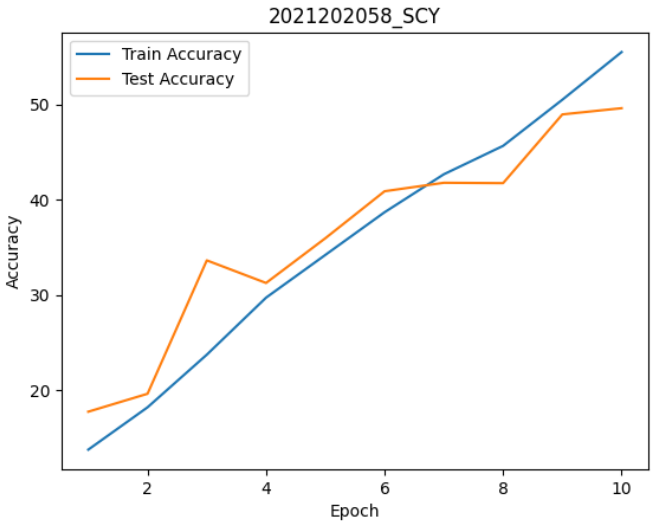
이어서 ResNet 역시 강의 자료 그림을 참고하여 구현하였다. 아래 사진은 구현한 ResNet34의 구조이다. 위에서 구현한 Block을 여러 개를 쌓아 만든 구조이다. 필터의 개수는 각 block을 거치며 2배씩 늘어나게 된다. (64 -> 128 -> 256 -> 512) 모든 block들을 거친 후에는 출력 이미지의 크기를 고정하기 위해 AdaptiveAgvPool2d를 적용하여 tensor로 만든 후 fc layer로 연결해주었다.



[Result]

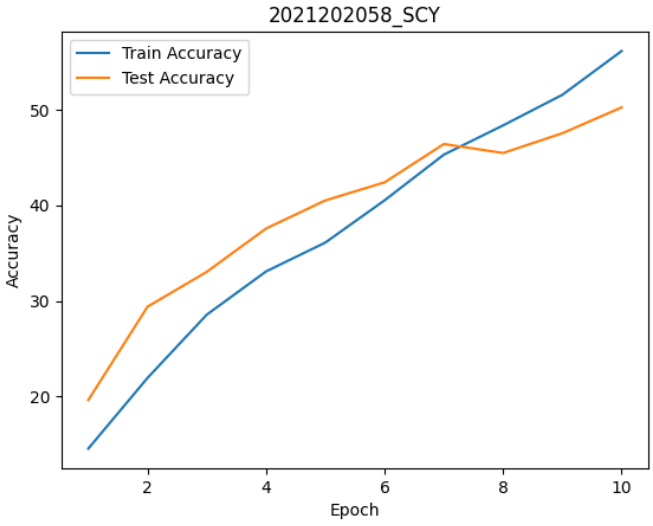
output 결과에 대해 설명하겠다. 우선 plot시 학번_이름으로 설정해야 하는데 한글로 넣을 경우 깨지는 것이 확인되어 이름 부분은 영어로 넣었다. CIFER-10의 사진 크기가 32x32이고, ResNet 34의 input image size는 224x224이므로 transforms.Resize를 여러 개로 실험해보았는데 128x128일

때의 크기가 가장 좋았다. 또한 learning rate 역시 기본으로 설정되어 있던 0.9 이외에도 여러 값으로 진행해보았는데 0.01일 때 가장 좋게 나와 0.01로 설정하였다. Director.py의 코드는 pytorch guide를 참고해서 구현하였다.



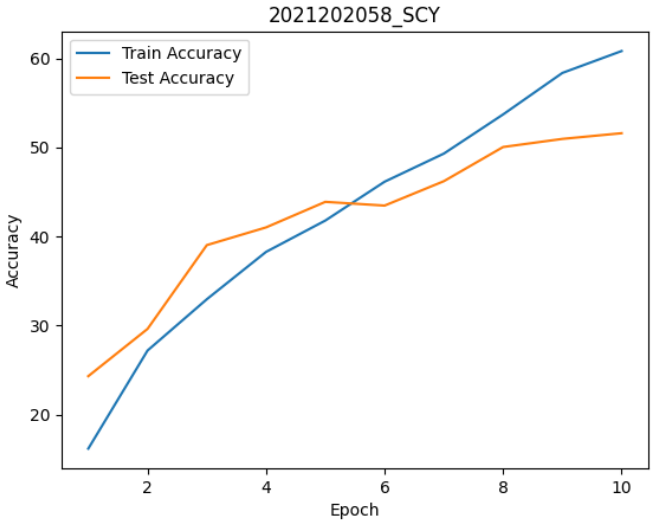

```
Epoch 1/10
Train Accuracy: 14.56 %
Test Accuracy: 19.64 %
Epoch 2/10
Train Accuracy: 21.96 %
Test Accuracy: 29.41 %
Epoch 3/10
Train Accuracy: 28.58 %
Test Accuracy: 33.06 %
Epoch 4/10
Train Accuracy: 33.1 %
Test Accuracy: 37.59 %
Epoch 5/10
Train Accuracy: 36.12 %
Test Accuracy: 40.53 %
Epoch 6/10
Train Accuracy: 40.54 %
Test Accuracy: 42.42 %
Epoch 7/10
Train Accuracy: 45.32 %
Test Accuracy: 46.43 %
Epoch 8/10
Train Accuracy: 48.38 %
Test Accuracy: 45.49 %
Epoch 9/10
Train Accuracy: 51.58 %
Test Accuracy: 47.56 %
Epoch 10/10
Train Accuracy: 56.16 %
Test Accuracy: 50.26 %
```

optimizer가 SGD 일 때의 결과이며 5번 실행해 본 결과 중 정확도가 가장 높은 것을 가져왔다.




```
Epoch 1/10
Train Accuracy: 13.8 %
Test Accuracy: 17.78 %
Epoch 2/10
Train Accuracy: 18.24 %
Test Accuracy: 19.66 %
Epoch 3/10
Train Accuracy: 23.76 %
Test Accuracy: 33.65 %
Epoch 4/10
Train Accuracy: 29.74 %
Test Accuracy: 31.28 %
Epoch 5/10
Train Accuracy: 34.22 %
Test Accuracy: 35.97 %
Epoch 6/10
Train Accuracy: 38.7 %
Test Accuracy: 40.9 %
Epoch 7/10
Train Accuracy: 42.68 %
Test Accuracy: 41.79 %
Epoch 8/10
Train Accuracy: 45.66 %
Test Accuracy: 41.75 %
Epoch 9/10
Train Accuracy: 50.5 %
Test Accuracy: 48.96 %
Epoch 10/10
Train Accuracy: 55.5 %
Test Accuracy: 49.61 %
```

optimizer가 Adam 일 때의 결과이며 5번 실행해 본 결과 중 정확도가 가장 높은 것을 가져왔다.




```
Epoch 1/10
Train Accuracy: 16.16 %
Test Accuracy: 24.3 %
Epoch 2/10
Train Accuracy: 27.18 %
Test Accuracy: 29.61 %
Epoch 3/10
Train Accuracy: 32.94 %
Test Accuracy: 39.02 %
Epoch 4/10
Train Accuracy: 38.26 %
Test Accuracy: 41.01 %
Epoch 5/10
Train Accuracy: 41.78 %
Test Accuracy: 43.88 %
Epoch 6/10
Train Accuracy: 46.14 %
Test Accuracy: 43.46 %
Epoch 7/10
Train Accuracy: 49.3 %
Test Accuracy: 46.2 %
Epoch 8/10
Train Accuracy: 53.7 %
Test Accuracy: 50.04 %
Epoch 9/10
Train Accuracy: 58.36 %
Test Accuracy: 50.95 %
Epoch 10/10
Train Accuracy: 60.82 %
Test Accuracy: 51.59 %
```

optimizer가 SGD 그리고 momentum을 0.95로 설정해주었을 때의 결과이며 5번 실행해 본 결과 중 가장 높은 것을 가져왔다.

전체적으로 optimizer는 SGD(momentum=0.9)를 사용하고, learning rate는 0.01일 때가 가장 결과가 높게 나왔으며, 전체 train data 50,000개 중 1/10인 5,000개만 사용하여 학습했기 때문에 높은 정확도를 뽑지 못했다고 생각한다.

[Reference]

- 머신러닝 Pytorch Guide pdf
- 머신러닝 강의자료
- https://docs.ultralytics.com/ko/datasets/classify/cifar10/ - CIFAR-10 dataset
- https://dlaguddnr.tistory.com/17 - ResNet
- https://haystar.tistory.com/94 - ResNet
- https://velog.io/@gibonki77/ResNetwithPyTorch - ResNet