Project

: Image classification model의 이해

과 목 딥러닝입문

전 공 컴퓨터공학전공

학 번 -

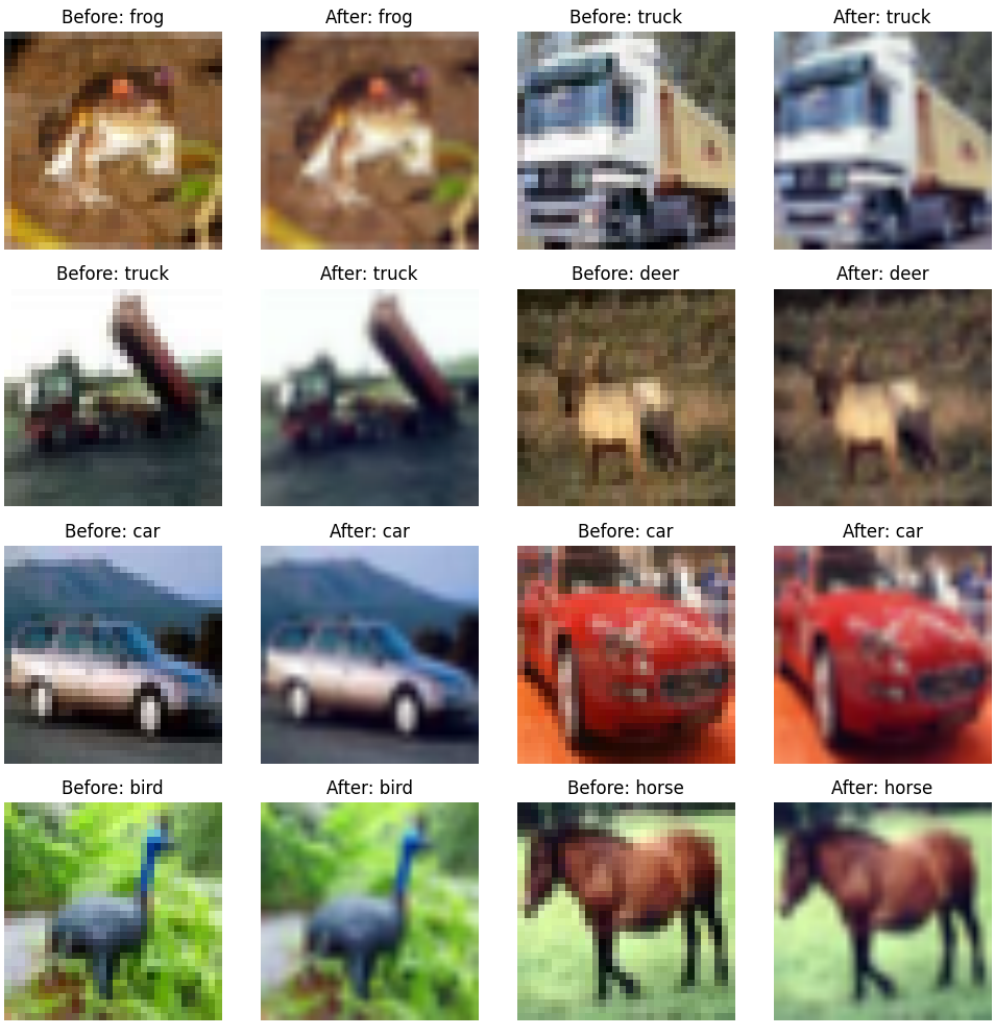
이 름 정수채

Ⅰ. 데이터 분석

이번 프로젝트에서 사용할 데이터는 CIFAR-10 데이터셋으로 32*32 픽셀의 컬러 이미지를 갖는다. 학습 데이터는 50,000개, 테스트 데이터는 10,000개로 구성되어 있으며 분류해야 하는 클래스는 10가지(비행기, 자동차, 새, 고양이, 사슴, 개, 개구리, 말, 배, 트럭) 이다.

데이터를 학습에 사용하기 위해 transforms.Compose()를 사용해 픽셀 크기를 224*224로 키우고, 표준화를 통해 전처리를 진행한다. 이때 전처리로 인한 시각적 변화를 확인하기 위해 data/data_load.py에 data_visualize_save() 함수를 정의했다.

[그림 1]을 통해 전처리를 거치기 전과 후의 차이를 확인할 수 있다. 전처리 전에는 해상도가 낮아 픽셀 블럭이 보이고 거친 느낌이었지만, 픽셀 크기를 키우니 부드럽게 표현되는 것을 볼 수 있다. 또한 표준화로 인해 전체적인 색감 차이가 줄어들었음을 확인할 수 있다.



[그림 1] CIFAR-10 데이터 전처리 전/후 비교

Ⅱ. 학습

Ⅱ.1. 모델 정의

학습에 사용할 모델은 모두 model/ 디렉토리에 위치한다. 모델은 총 세 가지로, alexnet.py에 AlexNet, alexnet_with_residual.py에 레지듀얼을 사용한 AlexNet(이후 AlexNet-R), alexnet_with_classifier.py에 선형 분류기를 사용한 AlexNet(이후 AlexNet-C)을 정의했다.

AlexNet은 교안의 코드를 참고하였고, classifier 부분을 embedding과 classifier로 분할하여 사용하였다. AlexNet-R은 기존 AlexNet의 features를 5가지 부분으로 분할한 후, 2개의 Skip-connection을 추가하였다. AlexNet-C는는 AlexNet 모델과 분류기 모델을 각각 입력받아 AlexNet의 embedding까지만 수행한 뒤 최종 분류는 분류기 모델을 사용하도록 구현하였다.

Ⅱ.2. 트레이너 정의

학습을 위한 CustomTrainer 클래스를 model/trainer.py에 정의했다. 해당 클래스는 [그림 2]와 같은 파라미터를 입력으로 받아 저장하고, fit() 함수를 호출하면 학습을 수행한다. 하이퍼파라미터에 따라 검증 에포크 수와 저장 에포크 수 등을 변경하여 동작하도록 구현했다.

```
class CustomTrainer():
    # 모델, 하이퍼파라미터, 손실 함수, 옵티마이저 정의
    def __init__(
        self,
        model: nn.Module,
        train_loader: torch.utils.data.DataLoader,
        test_loader: torch.utils.data.DataLoader,
        batch_size: int=64,
        epoch: int=50,
        learning_rate: float=1e-4,
        criterion: str="cross-entropy",
        optimizer: str="adam",
        logging_epoch: int=1,
        eval_epoch: int=1,
        save_epoch: int=1,
        save_dir: str='outputs',
        save_best_model: bool=True,
        device: str='cpu'
    ):
```

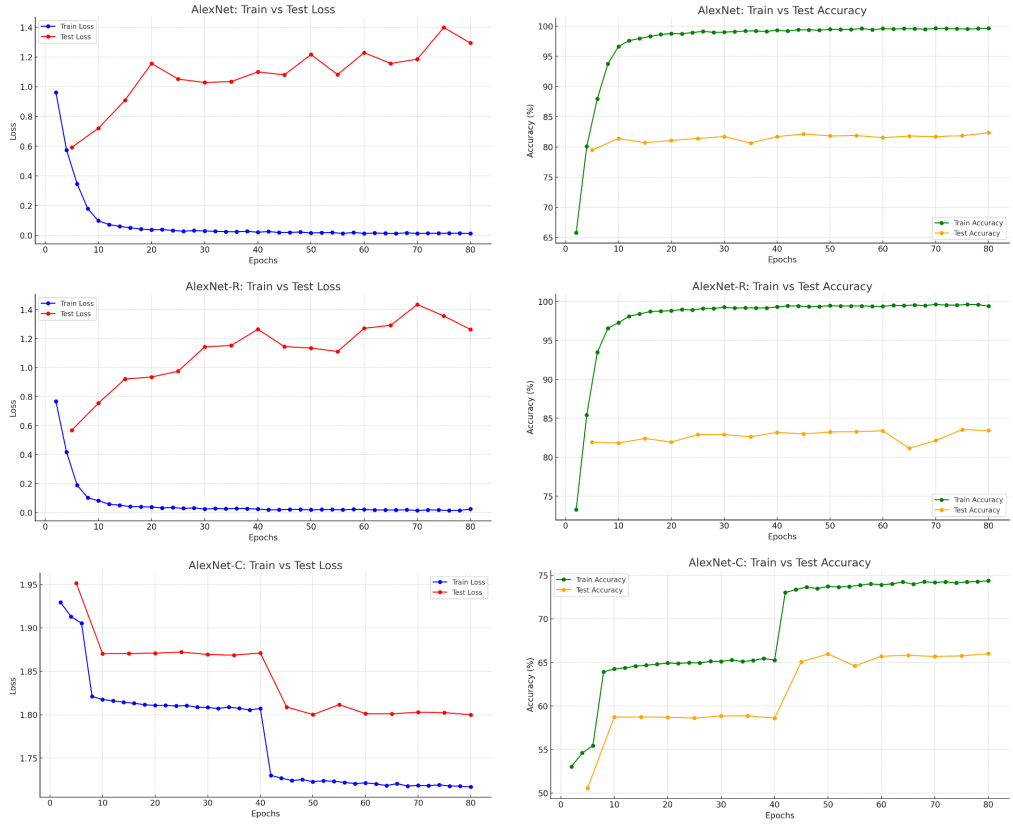
[그림 2] CustomTraininer() 생성자 파라미터

Ⅱ.3. 학습 코드

main/training.py에 Ⅱ.1과 Ⅱ.2에서 정의한 모델, 트레이너를 사용해 학습을 수행하는 코드를 구현했다. trainin.py 스크립트는 실행시 인자를 통해 '모델', '배치 크기', '에포크 수', '학습율'을 입력받고, 모델을 생성한 뒤 CustomTrainer를 생성해 학습하는 동작을 수행한다. AlexNet과 AlexNet-R은 바로 모델 클래스를 생성하지만, AlexNet-C를 학습할 때는 사전 학습된 AlexNet을 불러와 파라미터를 고정하고, 선형 회귀 모델을 생성하여 하나의 클래스로 만든다.

Ⅱ.4. 학습 수행

모든 모델을 학습할 때 하이퍼파라미터는 '배치 크기 64', '에포크 수 80', '학습율 1e-4'로 고정하였다. optimizer는 adam을 사용했고, 데이터 증강은 수행하지 않았다. 또한 학습 과정에서 과적합이 발생하는지 확인하기 위해 5번의 에포크마다 validation을 수행하였고, 동일하게 5번의 에포크마다 모델의 파라미터를 /outputs/{model}/에 저장했다. 그리고 save_best_model 파라미터를 통해 test accuracy가 가장 높은 모델을 /outputs/{model}/best_model/ 디렉토리에 저장했다.



(a)

(b)

(c)

[그림 3] 학습 중 train dataset과 test dataset의 Loss와 Accuracy

[그림 3]의 (a), (b), (c)는 각각 AlexNet, AlexNet-R, AlexNet-C의 학습과정에서 Loss와 Accuracy 변화를 보여준다. AlexNet과 AlexNet-R의 경우 20번째 에포크에서 train loss가 0에 수렴하고, train accuracy가 100%에 가까워지는 것을 확인할 수 있다. 그러나 5번째 에포크 이후로 test loss가 증가하고 test accuracy가 변화하지 않음을 통해 과적합 되고 있음을 알 수 있다.

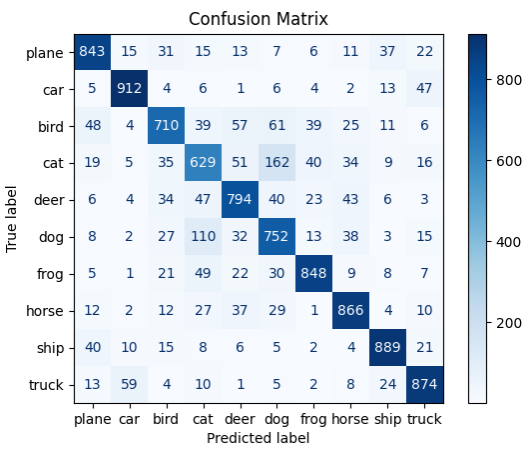
AlexNet-C의 학습에서는 손실 감소와 정확도 상승이 지속적으로 나타나는 것이 아니라 계단형식으로 떨어지는 것을 확인할 수 있었다. 이는 학습율을 높게 제공하여 안정적으로 수렴하지 못하고 특정 조건에서 크게 변화했거나, 선형 회귀

모델이 단순하여 Feature extractor에서 추출한 특징을 분류하는 방법을 적절히 학습하지 못함을 의미한다.

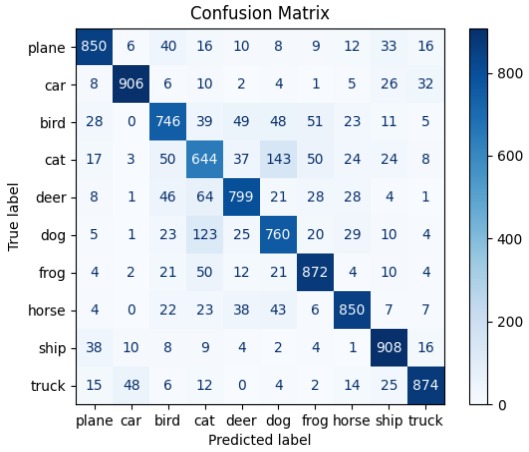
test data에 대한 모델의 정확도 측면에서도 AlexNet과 AlexNet-R은 82.34%, 83.55%의 성능을 보였지만 AlexNet-C는 66.01%에 그쳤다. 이를 통해 classifier 부분을 선형회귀 모델로 단순화 하였을 때 성능이 떨어짐을 알 수 있고, 기존 모델의 feature extractor 파라미터를 고정했다는 것을 고려했을 때 특징 추출이 충분히 이루어지지 않고 있음을 의미한다.

Ⅲ. 혼동 행렬

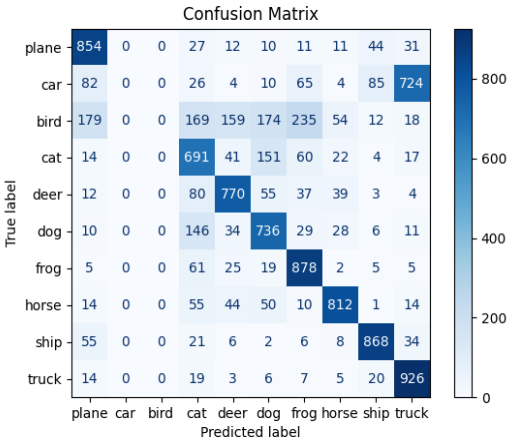
각 모델의 라벨 분류 정확도를 확인하기 위해 main/test.py에 test data의 예측 결과를 실제 라벨과 비교하여 혼동 행렬을 생성하도록 구현했다. 10000개의 test data를 모두 사용하여 행렬을 생성했고, 모델별 결과는 outputs/pictures/2_confusion_matrix_{model}.png에서 확인할 수 있다.



(a) AlexNet



(b) AlexNet-R



(c) AlexNet-C

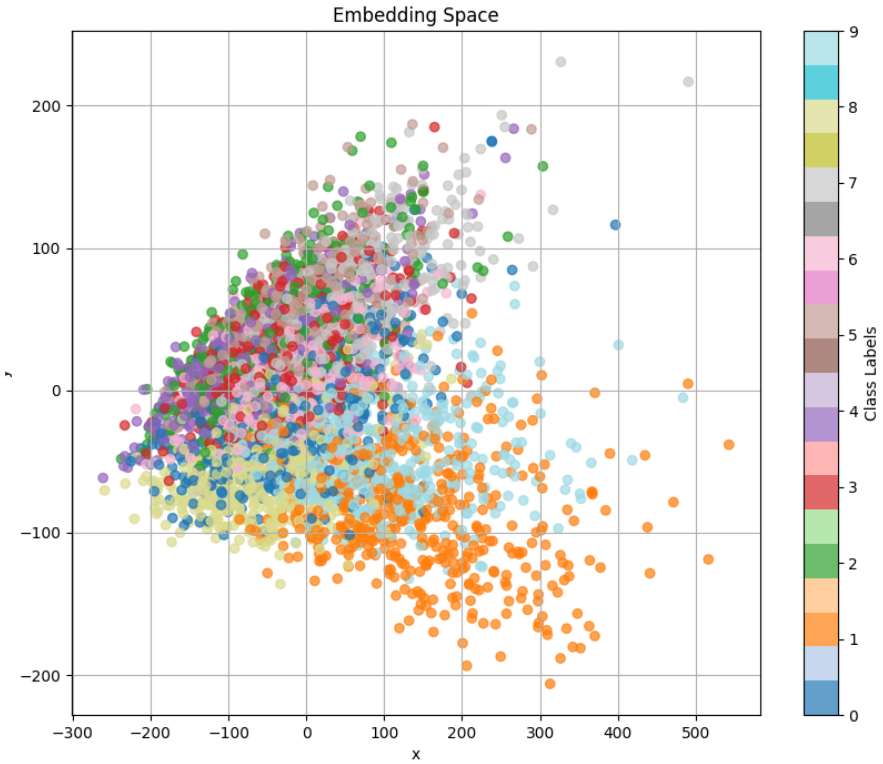
[그림 4] 각 모델의 혼동 행렬

[그림 4]의 결과를 보면 AlexNet과 AlexNet-R에서는 적절히 학습이 이루어져 대부분 정답을 맞추는 것을 확인할 수 있다. 흥미로운 점은 bird, cat, deer, dog, fog인 동물 클래스는 같은 동물 클래스로 잘못 예측하는 경우가 많았고 특히 cat-dog에서는 오답 중 40%를 상대 클래스로 예측했다는 것이다. 이를 통해 사람이 보기에도 유사한 객체일 수록 모델도 헷갈려함을 알 수 있다,

AlexNet-C에서는 대부분의 클래스가 제대로 예측됐지만, car와 bird 클래스는 단 한 번도 예측되지 못했다. 이를 통해 선형 회귀 모델이 클래스를 제대로 분류하지 못하고 있음을 알 수 있었고, 이는 간단한 모델로 분류할만큼 충분히 쉬운 특징이 추출되지 못했기 때문임을 추측할 수 있다. 특히 car의 경우 truck으로 오분류 되었는데, 추출된 특징에서 car와 truck이 유사함을 유추할 수 있다.

Ⅳ. 임베딩 공간 분석

모델이 만들어내는 임베딩 공간을 분석할 수 있도록 main/embedding_space_analysis.py에서 test data의 임베딩 값을 모아 산점도를 그린 뒤, outputs/pictures/3_embedding_space.png에 저장하도록 구현했다. 또한 4096차원의 임베딩 공간을 2차원에 표현하기 위한 PCA 모델도 models/pca.py에 구현했다. 임베딩 벡터는 AlexNet에서 test data에 대한 정확도가 가장 높았던 모델을 사용하여 계산했다.



[그림 5] AlexNet으로 만들어낸 CIFAR-10 데이터의 임베딩 시각화

[그림 5]의 클래스는 순서대로 (0) plane, (1) car, (2) bird, (3) cat, (4) deer, (5) dog, (6) frog, (7) horse, (8) ship, (9) truck이다. 혼동 행렬에서 보았던 특징대로 혼동하던 동물 클래스인 (2), (3), (4), (5), (6)의 임베딩 벡터가 매우 비슷한 위치에 겹쳐있는 것을 확인할 수 있다. 또한 선형 회귀 모델이 분류하지 못하던 car와 truck의 경우 (1)과 (9)의 임베딩 벡터가 동일한 위치에 분포되어 있는 것을 확인할 수 있었고, 이에 따라 두 클래스를 구분하는 방법을 학습하는 것이 어려웠음을 유추할 수 있다. (2) bird의 임베딩 벡터의 경우에도 전반적으로 분포되어 있어 다른 클래스와 구분하는 방법을 배우지 못했음을 알 수 있다.

전체 4096차원을 2차원으로 줄였다는 것과, 분포 기반의 PCA 방법만 사용했음을 감안하더라도 각 클래스들의 임베딩 벡터가 섞여 있음을 확인할 수 있고, 이를 통해 feature extractor 부분에서 충분한 특징 추출이 이뤄지지 않음을 다시 한 번 알 수 있다.

Ⅴ. 코드 실행

프로젝트에서 여러 모델을 사용하여 학습을 수행하고, 또 겹치는 코드가 많았기에 이를 모듈별로 분할하여 구현했다. 그리고, 학습을 수행할 때 항상 AlexNet을 AlexNet-C보다 먼저 학습해서 사전학습된 파라미터를 준비해야 했기에, 순서를 신경써야 했다. 따라서 추후 사용시 이를 간편화하기 위해 순서와 인자를 작성해 run_code.sh 파일에 작성하였다. 따라서 전체 코드를 실행하기 위해서는 아래의 2단계만 수행하면 된다.

1. 실행 권한 부여: chmod +x run_code.sh
2. 쉘 스크립트 실행: ./run_code.sh

정상적으로 실행되면 outputs 디렉토리에 각 모델의 학습 과정 파라미터와 best model 파라미터가 저장되고, outputs/pictures 디렉토리에 데이터 분석, 혼동 행렬, 임베딩 공간을 시각화한 표가 저장된다. run_code.sh의 실행 로그는 [그림 6]에서 확인할 수 있다.

```
nohup: ignoring input
Files already downloaded and verified
Files already downloaded and verified
Image saved to outputs/pictures/1_data_visualize

[Using AlexNet model.]
Epoch # 2 | Train Loss: 0.9622 | Train Accuracy: 65.79% |
Epoch # 4 | Train Loss: 0.5743 | Train Accuracy: 80.09% |
Test Loss: 0.5924 | Test Accuracy: 79.49%
Epoch # 6 | Train Loss: 0.3463 | Train Accuracy: 87.98% |
Epoch # 8 | Train Loss: 0.1793 | Train Accuracy: 93.76% |
Epoch #10 | Train Loss: 0.0979 | Train Accuracy: 96.64% |
Test Loss: 0.7209 | Test Accuracy: 81.39%
Epoch #12 | Train Loss: 0.0729 | Train Accuracy: 97.59% |
Epoch #14 | Train Loss: 0.0602 | Train Accuracy: 97.96% |
Test Loss: 0.9090 | Test Accuracy: 80.70%
Epoch #16 | Train Loss: 0.0511 | Train Accuracy: 98.30% |
Epoch #18 | Train Loss: 0.0422 | Train Accuracy: 98.62% |
Epoch #20 | Train Loss: 0.0377 | Train Accuracy: 98.78% |
Test Loss: 1.1566 | Test Accuracy: 81.08%
Epoch #22 | Train Loss: 0.0387 | Train Accuracy: 98.73% |
Epoch #24 | Train Loss: 0.0333 | Train Accuracy: 98.94% |
Test Loss: 1.0517 | Test Accuracy: 81.41%
Epoch #26 | Train Loss: 0.0280 | Train Accuracy: 99.14% |
Epoch #28 | Train Loss: 0.0323 | Train Accuracy: 98.97% |
Epoch #30 | Train Loss: 0.0295 | Train Accuracy: 99.00% |
Test Loss: 1.0286 | Test Accuracy: 81.72%
Epoch #32 | Train Loss: 0.0272 | Train Accuracy: 99.10% |
Epoch #34 | Train Loss: 0.0249 | Train Accuracy: 99.19% |
Test Loss: 1.0349 | Test Accuracy: 80.64%
Epoch #36 | Train Loss: 0.0242 | Train Accuracy: 99.20% |
Epoch #38 | Train Loss: 0.0274 | Train Accuracy: 99.12% |
Epoch #40 | Train Loss: 0.0208 | Train Accuracy: 99.32% |
Test Loss: 1.0998 | Test Accuracy: 81.71%
Epoch #42 | Train Loss: 0.0261 | Train Accuracy: 99.19% |
Epoch #44 | Train Loss: 0.0190 | Train Accuracy: 99.40% |
Test Loss: 1.0800 | Test Accuracy: 82.15%
Epoch #46 | Train Loss: 0.0203 | Train Accuracy: 99.38% |
Epoch #48 | Train Loss: 0.0223 | Train Accuracy: 99.32% |
Epoch #50 | Train Loss: 0.0166 | Train Accuracy: 99.50% |
Test Loss: 1.2166 | Test Accuracy: 81.83%
Epoch #52 | Train Loss: 0.0183 | Train Accuracy: 99.45% |
Epoch #54 | Train Loss: 0.0190 | Train Accuracy: 99.44% |
Test Loss: 1.0820 | Test Accuracy: 81.88%
Epoch #56 | Train Loss: 0.0129 | Train Accuracy: 99.60% |
Epoch #58 | Train Loss: 0.0197 | Train Accuracy: 99.40% |
Epoch #60 | Train Loss: 0.0134 | Train Accuracy: 99.59% |
Test Loss: 1.2294 | Test Accuracy: 81.54%
Epoch #62 | Train Loss: 0.0155 | Train Accuracy: 99.51% |
Epoch #64 | Train Loss: 0.0139 | Train Accuracy: 99.59% |
Test Loss: 1.1565 | Test Accuracy: 81.80%
Epoch #66 | Train Loss: 0.0130 | Train Accuracy: 99.58% |
Epoch #68 | Train Loss: 0.0169 | Train Accuracy: 99.47% |
Epoch #70 | Train Loss: 0.0129 | Train Accuracy: 99.64% |
Test Loss: 1.1859 | Test Accuracy: 81.72%
Epoch #72 | Train Loss: 0.0145 | Train Accuracy: 99.59% |
Epoch #74 | Train Loss: 0.0144 | Train Accuracy: 99.57% |
Test Loss: 1.3986 | Test Accuracy: 81.86%
Epoch #76 | Train Loss: 0.0145 | Train Accuracy: 99.51% |
Epoch #78 | Train Loss: 0.0145 | Train Accuracy: 99.59% |
Epoch #80 | Train Loss: 0.0126 | Train Accuracy: 99.64% |
Test Loss: 1.2941 | Test Accuracy: 82.34%
```

[Using AlexNetWithResidual model.]
Epoch # 2 | Train Loss: 0.7677 | Train Accuracy: 73.27% |
Epoch # 4 | Train Loss: 0.4174 | Train Accuracy: 85.43% |
Test Loss: 0.5687 | Test Accuracy: 81.93%
Epoch # 6 | Train Loss: 0.1883 | Train Accuracy: 93.48% |
Epoch # 8 | Train Loss: 0.1016 | Train Accuracy: 96.57% |
Epoch #10 | Train Loss: 0.0817 | Train Accuracy: 97.28% |
Test Loss: 0.7554 | Test Accuracy: 81.82%
Epoch #12 | Train Loss: 0.0576 | Train Accuracy: 98.11% |
Epoch #14 | Train Loss: 0.0507 | Train Accuracy: 98.40% |
Test Loss: 0.9217 | Test Accuracy: 82.41%
Epoch #16 | Train Loss: 0.0411 | Train Accuracy: 98.71% |
Epoch #18 | Train Loss: 0.0404 | Train Accuracy: 98.77% |
Epoch #20 | Train Loss: 0.0377 | Train Accuracy: 98.81% |
Test Loss: 0.9354 | Test Accuracy: 81.95%
Epoch #22 | Train Loss: 0.0320 | Train Accuracy: 98.98% |
Epoch #24 | Train Loss: 0.0344 | Train Accuracy: 98.91% |
Test Loss: 0.9756 | Test Accuracy: 82.92%
Epoch #26 | Train Loss: 0.0288 | Train Accuracy: 99.11% |
Epoch #28 | Train Loss: 0.0320 | Train Accuracy: 99.11% |
Epoch #30 | Train Loss: 0.0240 | Train Accuracy: 99.28% |
Test Loss: 1.1435 | Test Accuracy: 82.90%
Epoch #32 | Train Loss: 0.0277 | Train Accuracy: 99.15% |
Epoch #34 | Train Loss: 0.0261 | Train Accuracy: 99.20% |
Test Loss: 1.1535 | Test Accuracy: 82.63%
Epoch #36 | Train Loss: 0.0280 | Train Accuracy: 99.17% |
Epoch #38 | Train Loss: 0.0269 | Train Accuracy: 99.18% |
Epoch #40 | Train Loss: 0.0240 | Train Accuracy: 99.31% |
Test Loss: 1.2652 | Test Accuracy: 83.19%
Epoch #42 | Train Loss: 0.0191 | Train Accuracy: 99.44% |
Epoch #44 | Train Loss: 0.0194 | Train Accuracy: 99.43% |
Test Loss: 1.1452 | Test Accuracy: 82.99%
Epoch #46 | Train Loss: 0.0219 | Train Accuracy: 99.34% |
Epoch #48 | Train Loss: 0.0211 | Train Accuracy: 99.36% |
Epoch #50 | Train Loss: 0.0188 | Train Accuracy: 99.47% |
Test Loss: 1.1355 | Test Accuracy: 83.24%
Epoch #52 | Train Loss: 0.0206 | Train Accuracy: 99.42% |
Epoch #54 | Train Loss: 0.0207 | Train Accuracy: 99.43% |
Test Loss: 1.1117 | Test Accuracy: 83.29%
Epoch #56 | Train Loss: 0.0189 | Train Accuracy: 99.43% |
Epoch #58 | Train Loss: 0.0221 | Train Accuracy: 99.39% |
Epoch #60 | Train Loss: 0.0214 | Train Accuracy: 99.38% |
Test Loss: 1.2716 | Test Accuracy: 83.39%
Epoch #62 | Train Loss: 0.0178 | Train Accuracy: 99.52% |
Epoch #64 | Train Loss: 0.0177 | Train Accuracy: 99.49% |
Test Loss: 1.2924 | Test Accuracy: 81.15%
Epoch #66 | Train Loss: 0.0171 | Train Accuracy: 99.55% |
Epoch #68 | Train Loss: 0.0188 | Train Accuracy: 99.48% |
Epoch #70 | Train Loss: 0.0142 | Train Accuracy: 99.63% |
Test Loss: 1.4361 | Test Accuracy: 82.15%
Epoch #72 | Train Loss: 0.0180 | Train Accuracy: 99.54% |
Epoch #74 | Train Loss: 0.0172 | Train Accuracy: 99.53% |
Test Loss: 1.3568 | Test Accuracy: 83.55%
Epoch #76 | Train Loss: 0.0141 | Train Accuracy: 99.63% |
Epoch #78 | Train Loss: 0.0149 | Train Accuracy: 99.60% |
Epoch #80 | Train Loss: 0.0237 | Train Accuracy: 99.40% |
Test Loss: 1.2644 | Test Accuracy: 83.43%

```
[Using AlexNetWithClassifier model.]
Epoch # 2 | Train Loss: 1.9297 | Train Accuracy: 53.03% |
Epoch # 4 | Train Loss: 1.9131 | Train Accuracy: 54.61% |
Test Loss: 1.9516 | Test Accuracy: 50.55%
Epoch # 6 | Train Loss: 1.9054 | Train Accuracy: 55.43% |
Epoch # 8 | Train Loss: 1.8211 | Train Accuracy: 63.92% |
Epoch #10 | Train Loss: 1.8176 | Train Accuracy: 64.25% |
Test Loss: 1.8704 | Test Accuracy: 58.72%
Epoch #12 | Train Loss: 1.8160 | Train Accuracy: 64.38% |
Epoch #14 | Train Loss: 1.8145 | Train Accuracy: 64.58% |
Test Loss: 1.8705 | Test Accuracy: 58.73%
Epoch #16 | Train Loss: 1.8132 | Train Accuracy: 64.68% |
Epoch #18 | Train Loss: 1.8115 | Train Accuracy: 64.80% |
Epoch #20 | Train Loss: 1.8107 | Train Accuracy: 64.93% |
Test Loss: 1.8710 | Test Accuracy: 58.71%
Epoch #22 | Train Loss: 1.8107 | Train Accuracy: 64.89% |
Epoch #24 | Train Loss: 1.8101 | Train Accuracy: 64.97% |
Test Loss: 1.8722 | Test Accuracy: 58.61%
Epoch #26 | Train Loss: 1.8105 | Train Accuracy: 64.95% |
Epoch #28 | Train Loss: 1.8087 | Train Accuracy: 65.13% |
Epoch #30 | Train Loss: 1.8084 | Train Accuracy: 65.12% |
Test Loss: 1.8694 | Test Accuracy: 58.85%
Epoch #32 | Train Loss: 1.8071 | Train Accuracy: 65.28% |
Epoch #34 | Train Loss: 1.8088 | Train Accuracy: 65.10% |
Test Loss: 1.8687 | Test Accuracy: 58.87%
Epoch #36 | Train Loss: 1.8073 | Train Accuracy: 65.23% |
Epoch #38 | Train Loss: 1.8055 | Train Accuracy: 65.46% |
Epoch #40 | Train Loss: 1.8072 | Train Accuracy: 65.26% |
Test Loss: 1.8712 | Test Accuracy: 58.60%
Epoch #42 | Train Loss: 1.7301 | Train Accuracy: 73.02% |
Epoch #44 | Train Loss: 1.7269 | Train Accuracy: 73.38% |
Test Loss: 1.8089 | Test Accuracy: 65.07%
Epoch #46 | Train Loss: 1.7242 | Train Accuracy: 73.65% |
Epoch #48 | Train Loss: 1.7252 | Train Accuracy: 73.49% |
Epoch #50 | Train Loss: 1.7228 | Train Accuracy: 73.74% |
Test Loss: 1.8002 | Test Accuracy: 65.97%
Epoch #52 | Train Loss: 1.7239 | Train Accuracy: 73.66% |
Epoch #54 | Train Loss: 1.7233 | Train Accuracy: 73.72% |
Test Loss: 1.8116 | Test Accuracy: 64.60%
Epoch #56 | Train Loss: 1.7220 | Train Accuracy: 73.88% |
Epoch #58 | Train Loss: 1.7206 | Train Accuracy: 74.03% |
Epoch #60 | Train Loss: 1.7214 | Train Accuracy: 73.92% |
Test Loss: 1.8013 | Test Accuracy: 65.71%
Epoch #62 | Train Loss: 1.7203 | Train Accuracy: 74.03% |
Test Loss: 1.8010 | Test Accuracy: 65.83%
Epoch #66 | Train Loss: 1.7206 | Train Accuracy: 73.99% |
Epoch #68 | Train Loss: 1.7178 | Train Accuracy: 74.27% |
Epoch #70 | Train Loss: 1.7185 | Train Accuracy: 74.19% |
Test Loss: 1.8030 | Test Accuracy: 65.68%
Epoch #72 | Train Loss: 1.7182 | Train Accuracy: 74.26% |
Test Loss: 1.8024 | Test Accuracy: 65.76%
Epoch #76 | Train Loss: 1.7178 | Train Accuracy: 74.26% |
Epoch #78 | Train Loss: 1.7176 | Train Accuracy: 74.29% |
Epoch #80 | Train Loss: 1.7168 | Train Accuracy: 74.39% |
Test Loss: 1.8000 | Test Accuracy: 66.01%

Using AlexNet model.
Image saved to outputs/pictures/2_confusion_matrix_alexnet
Using AlexNetWithResidual model.
Image saved to outputs/pictures/2_confusion_matrix_alexnet_with_residual
Using AlexNetWithClassifier model.
Image saved to outputs/pictures/2_confusion_matrix_alexnet_with_classifier
Image saved to outputs/pictures/3_embedding_space
완료
```

[그림 6] run_code.sh 실행 로그