Introduction to Deep Neural Networks (Spring 2021)

Final Project Report

Student ID 2019311195

Name kimjiyu

1. 사용된 방법 설명

main 함수 초반에 설정하는 파라미터 중 **train_data_mode**를 'labeled_train'으로 설정하면 data augmentation + supervised learning 방법으로 모델을 학습시키고, 'semi_supervised'로 설정하면 data augmentation + semi-supervised learning 방법으로 모델을 학습시킨다.

- train_data_mode = 'labeled_train'인 경우

  1st prepare labeled train data, unlabeled test data

  supervised learning 방법을 사용하기 때문에 라벨이 존재하는 학습 데이터와 라벨이 존재하지 않는 테스트 데이터를 가져온다.

  2nd data augmentation

  가져온 데이터에 resize, random horizontal flip, color jitter, random rotation, normalize를 적용하여 data augmentation을 해준다. 그 결과, 라벨이 존재하는 학습 데이터의 개수가 5000개에서 25000개로 늘었다.

  3rd train model

```
model = CNN().to(device)
```

```
trainer._train(labeled_trainloader, labeled_validloader)
```

  모델은 model.py에 정의되어 있던 CNN을 사용하였다. (다양한 모델을 시도해본 결과, 모델끼리 성능 차이가 크지 않았다.)

  4th test model

```
pred, num_epochs, train_acc, valid_acc = trainer._test("Test", labeled_testloader)
```

  10000개의 테스트 데이터에 대한 예측 결과를 submission.csv 파일로 저장한다.

- train_data_mode = 'semi_supervised'인 경우

  1st prepare labeled train/unlabeled train dataset, unlabeled test dataset

  Semi-supervised learning 방법을 사용하기 때문에 라벨이 존재하는 학습 데이터, 라벨이 존재하지 않은 학습 데이터, 라벨이 존재하지 않은 테스트 데이터를 가져온다.

  2nd data augmentation

  가져온 데이터에 resize, random horizontal flip, color jitter, random rotation, normalize를 적용하여 data augmentation을 해준다. 그 결과, 라벨이 존재하는 학습 데이터의 개수가 5000개에서 25000개로 늘었고, 라벨이 존재하지 않는 학습 데이터의 개수가 35,551개에서 177,755개로 늘었다.

  3rd train model

```
model = CNN().to(device)
trainer._train(labeled_trainloader, labeled_validloader, unlabeled_trainloader)
```

  모델은 model.py에 정의되어 있던 CNN을 사용하였다. (다양한 모델을 시도해본 결과, 모델끼리 성능 차이는 크지 않았다.)

  4th test model

```
pred, num_epochs, train_acc, valid_acc = trainer._test("Test", labeled_testloader)
```

  10000개의 테스트 데이터에 대한 예측 결과를 submission.csv 파일로 저장한다.

2. 실험 설정

   하이퍼 파라미터 중 learning rate, train_batch가 성능에 가장 큰 영향을 미쳤다. 0.01, 0.005, 0.003, 0.001의 learning rate를 시도해본 결과, 학습 속도가 느려도 안전한 0.001이 적당했다. 32, 64, 128, 256을 시도해본 결과, train_batch가 64일 때가 validation accuracy가 가장 빨리 올랐다.

3. 여러 시도

   - Data augmentation + supervised learning

     처음에 시도한 방법은 data augmentation과 supervised learning이다. 인터넷이 찾아본 결과, data augmentation 방법이 매우 다양하였다. 교안에 제시된 crop, rotate, horizontal flip 외에도 다양한 data augmentation 방법을 사용해 보았다.

하지만 다양한 data augmentation 방법을 사용하는 것이 모델의 성능을 크게 높여주지는 못했다. 오히려, 데이터에 대한 정규화를 진행하는 것이 모델의 성능을 향상시키는데 가장 큰 기여를 했다. data augmentation을 진행한 후, 라벨이 붙어 있는 데이터를 이용하여 **supervised learning**을 진행한 결과, leaderboard 기준으로 **83**%의 정확도를 갖는 모델을 만들 수 있었다.

- Data augmentation +semi-supervised learning(pseudo labeling)

  다음으로 시도한 방법은 **data augmentation**과 **unsupervised learning**이다. Unsupervised learning에는 라벨이 붙어 있지 않은 데이터가 사용되기 때문에 이 데이터에 라벨을 붙여주는 **pseudo labeling** 또한 사용되었다.

  - Pseudo labeling without selecting high confidence predictions

    처음 시도한 방법에서 사용된 data augmentation을 동일하게 진행한 후, 라벨이 존재하는 학습 데이터를 사용하여 모델을 학습시켰다. 학습된 모델을 이용하여 라벨이 존재하지 않는 데이터의 라벨을 예측하는 pseudo labeling을 진행하였다. 기존에 라벨이 존재한 데이터와 pseudo labeling 과정을 거쳐 라벨이 존재하는 모든 데이터를 사용하여 모델을 다시 학습시켰다. 위의 과정을 진행한 결과, leaderboard 기준으로 **81**%의 정확도를 갖는 모델을 만들 수 있었다.

  - Pseudo labeling with selecting high confidence predictions

    이전 시도에서 Semi-supervised learning을 진행한 모델의 정확도가 오히려 supervised learning을 진행한 모델의 정확도보다 안 좋은 결과가 나왔다. pseudo labeling 과정에 문제가 있다는 것을 파악하여 해당 과정을 보완한 후, 다시 semi-supervised learning을 시도했다. 라벨이 존재하는 데이터를 사용하여 학습시킨 모델로 pseudo labeling을 진행할 때 최고 예측 확률이 threshold를 넘지 못하는 데이터는 이후에 사용될 학습 데이터에서 제외시켰다. 위의 과정을 진행한 결과, leaderboard 기준으로 **83**%의 정확도를 갖는 모델을 만들 수 있었다.

4. 결과