요약

GitHub Repo URL

- https://github.com/Gyuni/deep-learning-project-1

개요 및 문제 설명

안구 기저부 이미지와 해당 안구 기저부 이미지가 녹내장 진단을 받은 환자의 것인지 결과 값이 1,020 세트 존재합니다. 이 프로젝트에선 주어진 이미지를 학습해 어떠한 이미지가 주어졌을 때 해당 안구 기저부 이미지가 녹내장 환자의 것인지 추측하는 모델을 작성합니다.

방법론

주어진 이미지를 800장의 Training Set과 200장의 Test Set으로 분류합니다. Training Set을 이용해 학습을 진행하며 layer별 weight와 bias를 적절히 조정합니다. Test Set을 이용해 해당 모델이 얼마나 잘 학습되었는지 측정합니다.

발견 사항 및 결론

적은 Data Set으로도 약 70%에 달하는 정확도를 확보할 수 있었습니다. 더 많은 Data Set이 주어지거나, 주어진 Data Set 안에서도 Crop 된 형태의 Data나 Mask, NerveRemoved Data를 제대로 활용한다면 더 높은 정확도를 얻어낼 수 있을 것이라 기대합니다.

서론

Data Set에 대한 설명

주어진 Data Set에는 세 가지 데이터 세트의 안저 이미지와 해당 시신경 디스크/컵 세그먼트 및 녹내장 진단 정보가 포함되어 있습니다.

하고자 하는 일

주어진 Data Set을 학습해 새로운 이미지가 주어졌을 때 해당 이미지 속 안구가 녹내장에 해당하는 지 판단해야 합니다. 이를 위해 신경망을 구성하고 학습시킵니다.

방법

Data 선정

전처리 과정을 단축하기 위해 해상도가 동일한 square 형태의 이미지 Set만을 사용합니다.

Training Set과 Test Set으로 분류

Training Set과 Test Set은 8:2의 비율이 적절합니다. 주어진 Data는 총 1,020개인데 이미지의 number를 기준으로 sorting 한 후 앞의 800개를 Training Set으로, 뒤의 220개를 Test Set으로 사용합니다.

이를 서로 다른 directory를 생성해 분류했습니다. 또한 각 set이 녹내장 진단 여부를 받았는지 여부를 test_label.csv와 train_label.csv 파일에 나누어 기록합니다.

Data 로드 및 리사이징

load_glaucoma() 함수를 이용해 Data Set을 Load 합니다. 이때 형태는 numpy의 ndarray를 따르도록 합니다.

이미지

- 원본 해상도인 512*512를 그대로 load 한 결과 랩탑에서 실행 불가능 할 정도로 메모리를 소모하는 문제가 있었습니다. 따라서 해상도를 64*64 로 조절하여 load 합니다.

- 컬러 이미지이기 때문에 r, g, b 3개 채널의 정보가 독립되어 있습니다. 따라서 이미지 파일 하나는 (3, 64, 64) 차원의 array로 나타낼 수 있습니다.

Label

- CSV 파일을 읽어들여 1차원 array로 나타낼 수 있습니다.

신경망 구성

```
conv - relu - conv- relu - pool -
        conv - relu - conv- relu - pool -
        conv - relu - conv- relu - pool -
        affine - relu - dropout - affine - dropout - softmax
```

위와 같이 구성되어 있습니다.

epoch, batch size, evaluate_sample_num_per_epoch 등 설정

mnist data set을 통해 신경망을 학습한 예제를 참고해 값을 조정했습니다.

- 해당 Data set에선 training data가 60,000개 test data가 10,000개였습니다. 이때 epochs가 20, mini_batch_size가 100, evaluate_sample_num_per_epoch가 1000 이었습니다.
- 주어진 Data Set은 training data가 800개, test data가 200개 이므로 mini_batch_size를 50, evaluate_sample_num_per_epoch를 50으로 조절했습니다.

epoch는 20으로 시작했으나 그 이상 값을 올려도 정확도가 개선되지 않아 20을 유지했습니다.

결과 저장

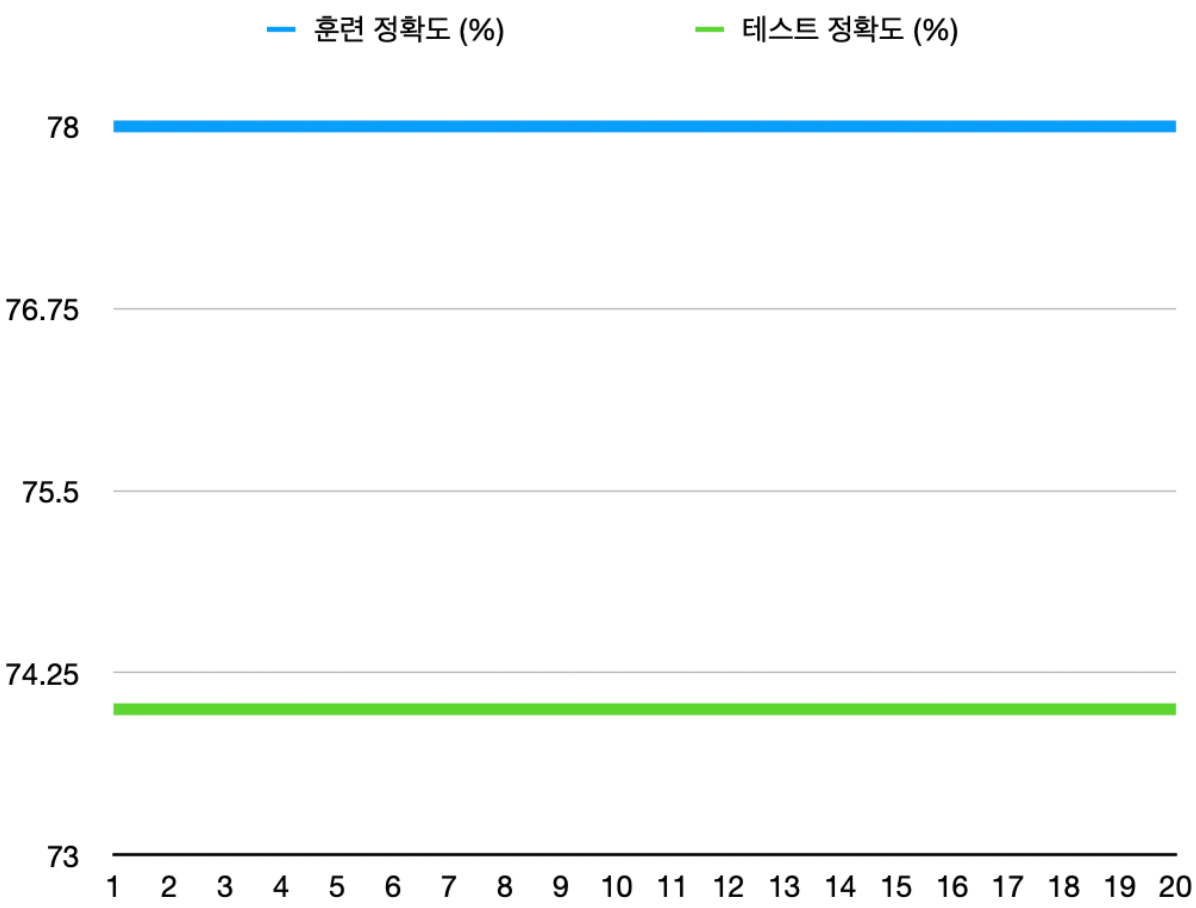
- training 결과인 각 layer별 weight, bias 등은 deep_convert_params.pkl 파일에 저장했습니다.
- 이후 해당 모델을 사용하고자 할 때 신경망을 새로 학습할 필요 없이 기존에 저장해둔 param 값을 load 해 그대로 사용할 수 있습니다.

결과

훈련 및 테스트 정확도 변화

epoch별 훈련 및 테스트 정확도

아래 표와 그래프는 각 epoch마다의 훈련 정확도와 테스트 정확도를 나타냅니다.



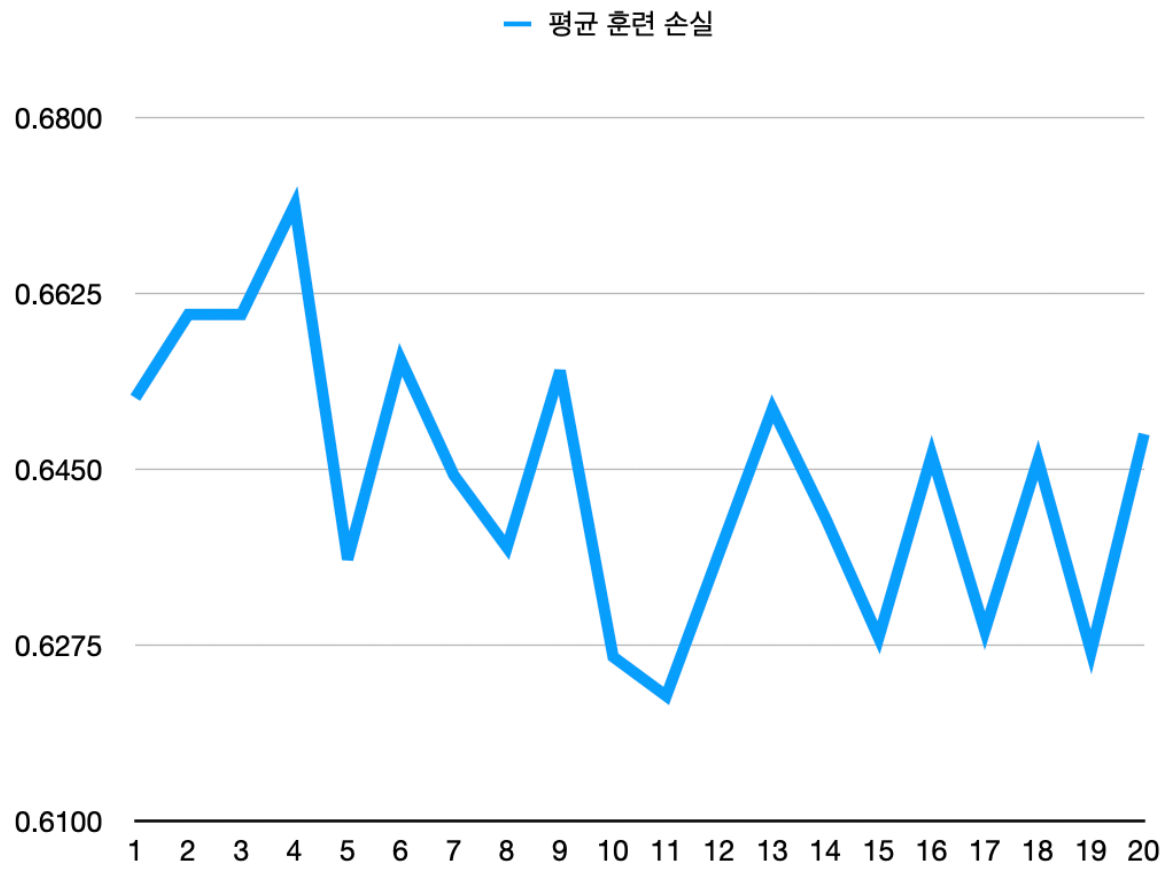
epoch	훈련 정확도 (%)	테스트 정확도 (%)
1	78	74
2	78	74
3	78	74
4	78	74
5	78	74
6	78	74
7	78	74
8	78	74
9	78	74

10	78	74
11	78	74
12	78	74
13	78	74
14	78	74
15	78	74
16	78	74
17	78	74
18	78	74
19	78	74
20	78	74

- 훈련 정확도와 테스트 정확도가 epoch 1부터 20까지 **78%**와 **74%**로 일정하게 유지되고 있습니다.

평균 훈련 손실 변화

각 epoch마다 16번의 훈련 손실 값이 기록되었으며, 이를 평균하여 epoch별 평균 훈련 손실을 계산하였습니다.



epoch	평균 훈련 손실
1	0.6521
2	0.6604
3	0.6604
4	0.6713
5	0.6360
6	0.6559
7	0.6444
8	0.6372
9	0.6548

10	0.6263
11	0.6224
12	0.6367
13	0.6510
14	0.6401
15	0.6282
16	0.6464
17	0.6289
18	0.6459
19	0.6268
20	0.6485

논의

훈련 및 테스트 정확도

- epoch 1부터 20까지 훈련 정확도는 78%, 테스트 정확도는 74% 로 일정하게 유지되었습니다.
- 이는 모델이 학습 초기부터 일정한 수준의 성능을 보였으나 추가적인 학습을 통해 성능 향상이 이루어지지 않았음을 의미합니다.

평균 훈련 손실

- 평균 훈련 손실은 epoch마다 약간의 변동이 있었지만 대체로 0.62에서 0.67 사이를 유지하였습니다.
- 손실 값의 큰 감소 없이 일정한 수준을 유지하는 것은 모델이 더 이상 학습되지 않고 있음

을 시사합니다.

최종 테스트 정확도

- 최종 테스트 정확도는 **68.64%**로 epoch 동안의 테스트 정확도인 74%보다 낮게 나타났습니다.
- 이는 모델의 일반화 성능이 기대에 미치지 못함을 나타냅니다.

결론

초기 학습 단계부터 일정한 성능을 보였지만 추가적인 학습을 통해 성능 향상이 이루어지지 않았습니다.

원인 분석

- 학습률이 부적절하여 모델이 최적의 파라미터로 수렴하지 못했을 수 있습니다
- 훈련 데이터의 양이나 다양성이 부족하여 모델이 충분한 학습을 하지 못했을 수 있습니다
- 모델의 복잡도가 데이터의 특성을 반영하기에 부족하거나 과도하여 발생하는 문제일 수 있습니다

향후 개선 방안

- 학습률, 배치 크기 등 하이퍼파라미터를 조정하여 모델의 학습 효율을 향상시킬 수 있습니다
- 데이터의 양과 다양성을 늘려 모델의 일반화 성능을 높일 수 있습니다
- 현재 모델의 구조를 분석하고 필요에 따라 더 깊은 네트워크나 적절한 정규화 기법을 적용할 수 있습니다

참고 문헌

- https://www.kaggle.com/datasets/arnavjain1/glaucoma-datasets/data
- https://github.com/ssuai/deep_learning_from_scratch