HAM10000 보고서

학과 : 소프트웨어학과

학번 : 20202020827

학년 : 4학년

이름 : 김경민

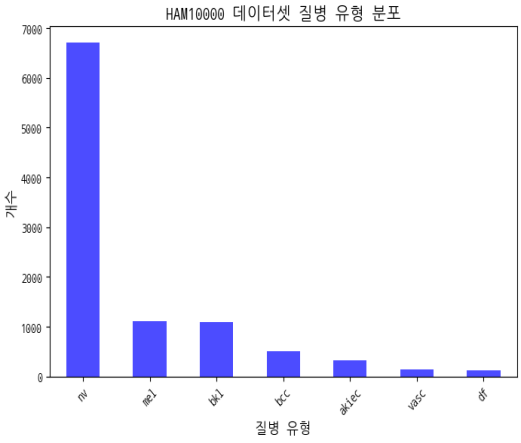
파일 구조

HAM10000/
|— data/
|　　├— ham10000_images_part_1/
|　　├— ham10000_images_part_2/
|　　├— HAM10000_metadata.csv
|　　├— hmnist_8_8_L.csv
|　　├— hmnist_8_8_RGB.csv
|　　├— hmnist_28_28_L.csv
|　　├— hmnist_28_28_RGB.csv
|— ham.ipynb

주요 파일 설명

파일명	설명	이미지 크기	색상	데이터 형태
HAM10000_metadata.csv	HAM10000 메타데이터	–	–	CSV (텍스트)
hmnist_8_8_L.csv	8x8 흑백 이미지	8x8	Grayscale	64 개 값 + 라벨
hmnist_8_8_RGB.csv	8x8 컬러 이미지	8x8	RGB	192 개 값 + 라벨
hmnist_28_28_L.csv	28x28 흑백 이미지	28x28	Grayscale	784 개 값 + 라벨
hmnist_28_28_RGB.csv	28x28 컬러 이미지	28x28	RGB	2352 개 값 + 라벨

Metadata.csv – 코드 질병 유형 분포



akiec	광선각화증 및 표피 내 암종
bcc	기저세포암종
bkl	양성 각화증성 병변
df	피부 섬유종
mel	흑색종
nv	모반(점)
vasc	혈관 병변

딥러닝 기반 피부 병변 분류 보고서

1. 개요

본 프로젝트는 **HAM10000** 데이터셋을 활용하여 피부 병변 이미지를 분류하는 모델을 구현하고, 여러 사전 학습(Pretrained) 모델을 비교·분석 했습니다. 프로젝트 전 과정에서 데이터 로딩 및 전처리, PyTorch 를 이용한 사용자 정의 Dataset 구현, 다양한 이미지 증강(Augmentation) 기법 적용, 모델 학습 및 평가, 성능 지표 시각화 등을 모두 직접 수행하며, 딥러닝 실습 과정을 학습했습니다.

---

2. 데이터 전처리

2.1. 메타데이터 로딩 및 결측치 처리

- **학습한 내용**
  - pandas 라이브러리를 이용하여 HAM10000_metadata.csv 파일을 불러오는 방법을 배웠습니다.
  - isnull()과 fillna() 메서드를 활용하여 **결측치**(특히 나이(age) 컬럼)를 평균값으로 대체하는 과정을 익혔습니다.
  - map() 함수를 사용해 병변 유형(dx)을 실제 병변명(예: 'mel' -> 'Melanoma')으로 매핑해, 분석 및 시각화 시 사람이 이해하기 쉽게 바꾸는 방법을 배웠습니다.

2.2. 사용자 정의 Dataset 및 DataLoader

- **이미지 경로 처리**
  - os.path.exists() 등을 통해 여러 폴더에서 이미지를 찾고, 없는 경우 에러를 발생시키도록 구현함으로써 **파일 입출력** 에러 처리 방식을 학습했습니다.
- **데이터셋 분할**
  - torch.utils.data.random_split 을 사용해 학습(80%), 검증(10%), 테스트(10%) 세트로 분할했습니다.
  - 이 과정을 통해 **학습, 검증, 테스트 세트의 필요성**과 각각의 역할(모델 훈련, 하이퍼파라미터 튜닝, 최종 성능 평가)에 대해 이해했습니다.

2.3. 이미지 증강(Augmentation)

- **학습한 내용**
  - transforms.Compose()를 통해 RandomResizedCrop, RandomHorizontalFlip, RandomRotation, ColorJitter 등을 조합해 **이미지 증강** 기법을 적용함으로써 모델의 **일반화 성능**을 높이는 방법을 배웠습니다.
  - 검증/테스트 데이터에는 **Resize** 와 **정규화**만을 적용하여, 평가 시 데이터 분포가 학습 시와 일관되도록 유지하는 원칙을 익혔습니다.

---

3. 모델 구성 및 학습 과정

3.1. 사전 학습(Pretrained) 모델 활용

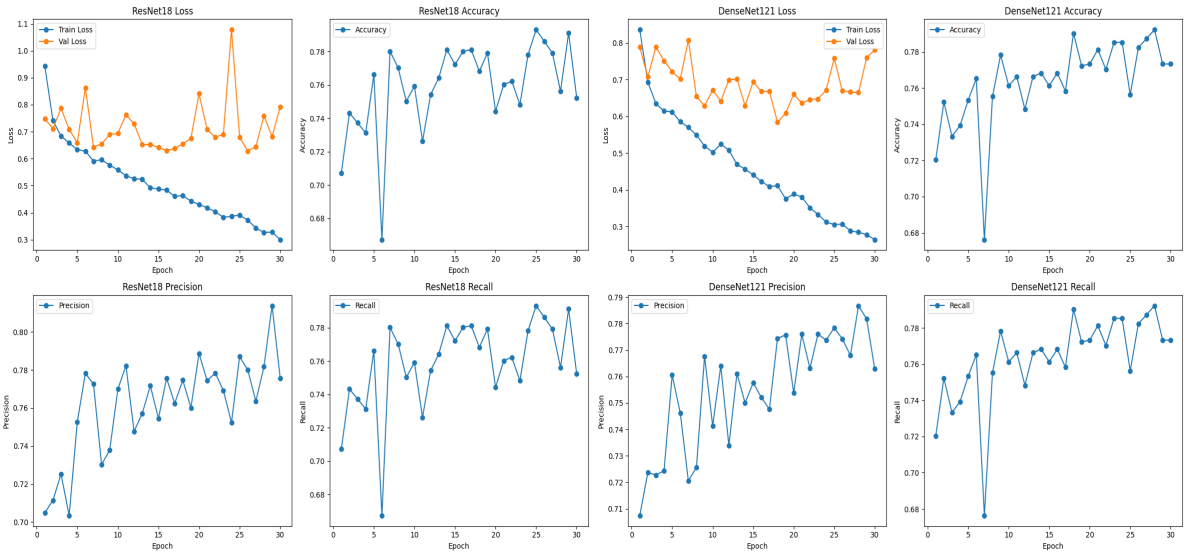
- **학습한 내용**
  - PyTorch 의 `torchvision.models` 에서 제공하는 **ResNet18**, **DenseNet121**, **MobileNetV2**, **EfficientNet-B0** 를 불러오는 방법과, 마지막 레이어(FC Layer)를 현재 프로젝트의 클래스 개수(7 개 병변)에 맞게 변경하는 방법을 익혔습니다.
  - 사전 학습된 가중치를 사용함으로써 **전이 학습(Transfer Learning)** 개념을 배웠으며, 비교적 적은 데이터로도 높은 성능을 낼 수 있다는 장점을 확인했습니다.

3.2. 학습 파이프라인

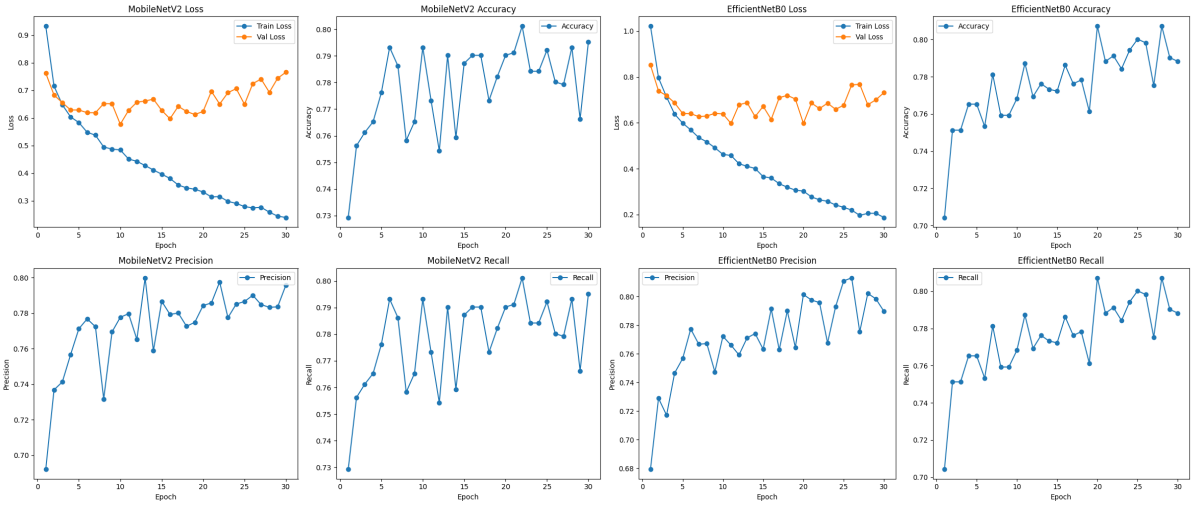
- **Train/Eval 함수 구현**
  - `train_model` 함수를 통해 각 에포크(epoch)마다 학습 단계(모델 파라미터 업데이트)와 검증 단계를 수행하고, **손실(loss)**, **정확도(Accuracy)**, **정밀도(Precision)**, **재현율(Recall)**을 계산했습니다.
  - `evaluate_model` 함수를 별도로 만들어 **테스트 세트**에서의 최종 성능을 확인했습니다.
  - 이 과정을 통해 **역전파(Backpropagation)**와 **옵티마이저(Adam, lr=0.001)** 사용법, 배치 단위 학습, 에포크별 성능 모니터링 방법 등을 학습했습니다.
- **학습률 스케줄러(Scheduler)**
  - 필요 시 `scheduler.step()`을 통해 검증 손실에 따라 학습률을 조정하는 방법도 배웠습니다.

3.3. 성능 지표 시각화

ResNet18 DenseNet121



MobileNetV2 EfficientNetB0



- plot_metrics 함수를 작성하여,
  - 에포크별 Train Loss, Validation Loss, Accuracy, Precision, Recall 을 그래프로 시각화했습니다.
- 이를 통해 학습이 **오버피팅/언더피팅**되는지 여부, 모델이 수렴하는 양상 등을 직관적으로 파악하는 방법을 익혔습니다.

---

4. 모델별 결과 및 해석

4.1. 최종 결과 (표)

아래 표는 테스트 세트에서의 최종 성능을 요약한 것입니다.
(Precision, Recall 은 멀티클래스 분류에서의 가중 평균(weighted average) 기준)

Model	Test Loss	Accuracy (%)	Precision	Recall
ResNet18	0.7818	74.65	0.7687	0.7465
DenseNet121	0.6609	79.74	0.7836	0.7974
MobileNetV2	0.7103	80.34	0.8006	0.8034
EfficientNetB0	0.7099	81.04	0.8023	0.8104

4.2. 결과 해석

1. ResNet18

- **정확도(74.65%)**: ResNet18 은 다른 모델들에 비해 상대적으로 낮은 정확도를 보였습니다. 하지만, **학습 속도가 빠르고, 모델 크기가 작아서 실험하기 용이**했습니다. 이 모델은 빠르게 학습이 진행되지만, **훈련 데이터에 과도하게 맞춰지는 오버피팅**이 상대적으로 빠르게 나타나는 경향이 있었습니다.

2. DenseNet121

- **정확도(80.24%)**: DenseNet121 은 정확도에서 좋은 성능을 보였으며, 특히 **학습 후반부에 안정적으로 높은 성능**을 유지했습니다. DenseNet 은 **각 계층에서 학습된 특징을 잘 전달**해주기 때문에, **중간 레이어에서도 중요한 특징들을 효과적으로 학습**할 수 있었습니다. 이는 모델이 더 많은 정보를 활용할 수 있게 도와주며, 성능 향상에 기여했습니다.

3. MobileNetV2

- **정확도(80.34%)**: MobileNetV2 는 DenseNet121 과 비슷한 수준의 성능(정확도)을 보였으나, **모델이 경량화되어** 연산량이 적고 파라미터가 적습니다.

4. EfficientNet-B0

- **정확도(81.04%)**: EfficientNet-B0 는 모든 모델 중에서 **가장 높은 정확도**를 보였습니다. 이 모델은 **복합적 스케일링(compound scaling)** 방법을 통해, **적은 파라미터로도 높은 성능을** 이끌어냈습니다.

---

5. 결론 및 느낀 점

이번 프로젝트를 통해 **데이터 전처리부터 모델 구축, 학습, 평가, 시각화**에 이르는 전 과정을 구현해 보며, 다음과 같은 점을 배웠습니다.

- **데이터 전처리의 중요성**
  - 결측치 처리, 병변 유형 매핑 등 데이터 자체가 깨끗해야 모델 학습이 원활하게 진행됨을 실감했습니다.
- **전이 학습(Transfer Learning)의 효과**
  - 사전 학습된 모델을 사용함으로써 적은 데이터로도 높은 성능을 달성할 수 있었으며, 다양한 모델을 비교해보는 경험을 통해 모델 구조가 성능에 큰 영향을 미침을 배웠습니다.
- **평가 지표 활용**
  - Accuracy 이외에도 Precision, Recall 등 다양한 지표를 살펴보며 클래스 불균형 상황에서 성능을 해석하는 방법을 배웠습니다.
- **오버피팅, 언더피팅, 기울기 손실에 대한 이해**
  - 오버피팅(Overfitting)과 언더피팅(Underfitting)의 개념을 명확히 이해했습니다. 오버피팅은 모델이 학습 데이터에 너무 맞춰져서 테스트 데이터에 대한 일반화 능력이 떨어지는 현상이고, 언더피팅은 모델이 학습 데이터의 패턴을 충분히 학습하지 못하는 경우입니다.

    - 또한, 기울기 손실(Gradient Loss)에 대해 이해를 했습니다. 기울기 손실은 역전파 과정에서 파라미터 업데이트가 제대로 이루어지지 않거나, 학습이 제대로 진행되지 않는 경우 발생할 수 있는 문제입니다. 이를 해결하기 위해 학습률을 적절히 설정하고, 옵티마이저를 잘 선택하는 것이 중요함을 배웠습니다.

- **모델 내부 구조와 학습 과정의 이해 부족**

    - 모델을 가져와서 사용해본 후, 각 모델의 내부 구조에 대한 명확한 이해가 부족하다는 것을 느꼈습니다. 향후 더 깊은 학습을 통해 각 모델이 어떻게 작동하는지, 특히 네트워크 아키텍처와 레이어의 연결 방식에 대한 많은 공부를 해야겠다고 생각했습니다.

- **역전파(Backpropagation)와 옵티마이저, 스케줄러에 대한 학습 필요**

    - 역전파와 옵티마이저(Adam 등) 사용법을 공부하는 과정에서, 각 모델의 파라미터 업데이트 방식과 학습률 조정 방법에 대해 더 깊이 학습해야겠다는 필요성을 느꼈습니다. 특히, 학습률 조정 방법(스케줄러)에 대해 더 이해하고 활용법을 개선할 필요성을 느꼈습니다. 이는 모델 학습의 안정성 및 최적화에 중요한 영향을 미침을 알게 되었습니다.

참고자료

- YH Kim. "ResNet-18 논문 리뷰," https://yhkim4504.tistory.com/3.
- Jaehoon Go. "Residual Learning의 이해와 ResNet-18 구현," https://velog.io/@jaehoon_go/Residual-Learning%EC%9D%98-%EC%9D%B4%ED%95%B4%EC%99%80-ResNet-18-%EA%B5%AC%ED%98%84.
- Channel AI. "잔차 연결(Residual Connections)," https://channelai.tistory.com/2.
- Wikidocs. "기울기 손실 및 과적합," https://wikidocs.net/61375.
- Sikmulation. "오버피팅과 언더피팅," https://sikmulation.tistory.com/50.
- Wkddmswh99. "Train, Validation, Test 데이터셋 나누기," https://wkddmswh99.tistory.com/10.
- Sid321axn. "Step-wise Approach to CNN Model (77.0344% Accuracy)," https://www.kaggle.com/code/sid321axn/step-wise-approach-cnn-model-77-0344-accuracy.
- Hunkim. "인프런 강의," https://hunkim.github.io/ml/.
- "Underfitting and Overfitting," https://ko.d2l.ai/chapter_deep-learning-basics/underfit-overfit.html.

- Wikidocs. "과적합과 과소적합," https://wikidocs.net/37406.