**심장질환 환자 ECG 데이터 분석을 위한 딥러닝 기법 설계**

**-중간보고서**



**<Deep Heart 조>**

**201824447 김정무**

**201824455 김지윤**

**201824601 천효승**

**목차**

1. **요구조건 및 제약 사항 분석에 대한 수정사항**…………………………………………..………………3

1-1. 요구조건

1-2 데이터 변경사항

1. **모델 상세화 및 변경 내역**….…………………………………………………………………….………..………4

2-1 모델 설계

2-2 LSTM

2-3 ResNet(CNN)

1. **보고 시점까지의 과제 수행 내용 및 중간 결과**…..………………………………...…………………16
2. **갱신된 과제 추진 계획**……………………………………………………………………………………………...17
3. **구성원별 진척도**…………..…………………………………………………………………………………….……..18
4. **요구조건 및 제약 사항 분석에 대한 수정사항**

**1-1. 요구조건**

PTB-XL ECG dataset을 통해 딥러닝 기법을 설계한다. 딥러닝 모델을 통하여 환자의 현 상태를 대입하면, 환자가 어떤 질병을 가지고 있는지 추론할 수 있는 모델을 만든다. LSTM(RNN), ResNet(CNN) 등 다양한 방법의 머신러닝 모델을 제작하고, 정확도 및 적합성을 바탕으로 특정 방법을 채택하기로 하였다. 목표는 모델의 정확도를 최대화하는 것이다.

**1-2 데이터 변경사항**

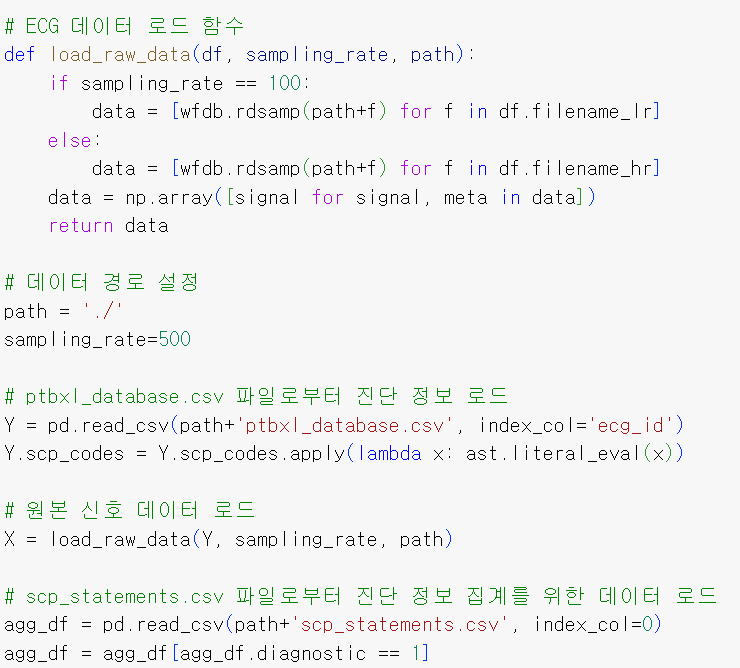
데이터가 23,000명이 넘는 환자들의 ECG 데이터를 포함하고 있기 때문에 Colab에서 프로그램을 돌릴 때 데이터를 불러오는 과정에서 과도하게 많은 시간이 소요되었다. 따라서 데이터의 사이즈를 줄여서 학습과 테스트를 진행하였다. 데이터 셋의 크기가 과도하게 작으면 딥러닝이 제대로 이루어지지 않기 때문에, 데이터의 크기를 500부터 2000까지 키워 가며 실행하고, 모델의 적합성을 판단했다. 축소된 데이터를 이용하여 Colab 환경에서 테스트한 뒤, 원본 데이터를 학교 연구실 서버의 Jupyter Lab 환경으로 실행하였다.

1. **설계 상세화 및 변경 내역**

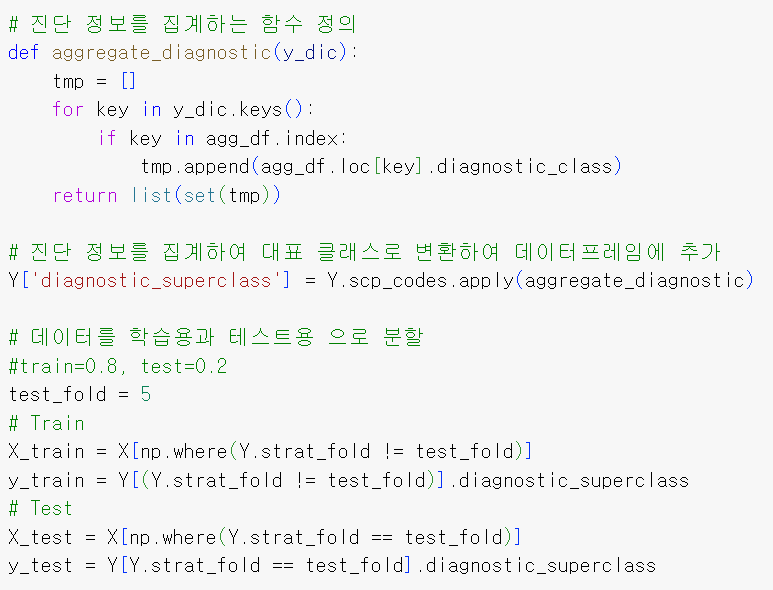
**2-1 모델설계**

머신러닝 모델을 만드는 데 사용되는 기법은 여러 가지가 있는데, 데이터가 sequential 하므로 LSTM(RNN), ResNet(CNN)의 종류를 사용하는 것이 좋을 것 같다고 판단하였고, 이를 통해 두 가지의 서로 다른 머신러닝 모델을 만들어 보기로 하였다.

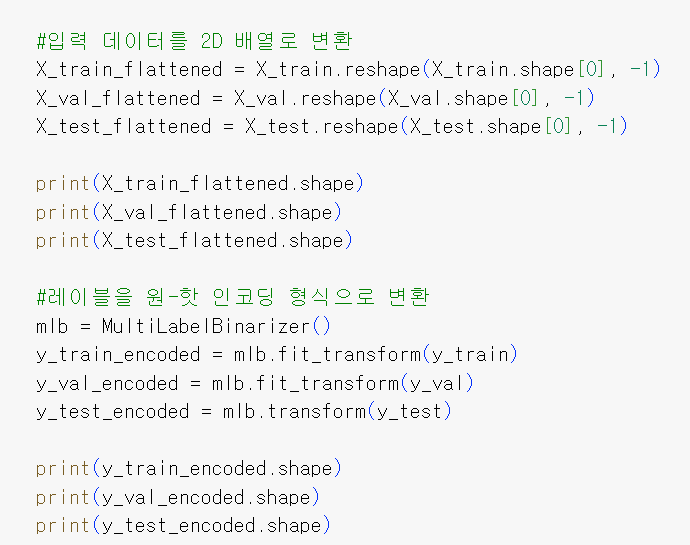
**2-2 LSTM 모델**

****

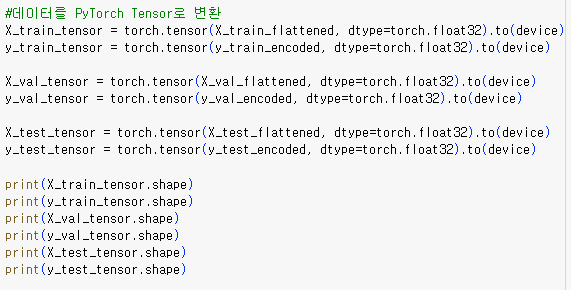
* 데이터를 로드한다.



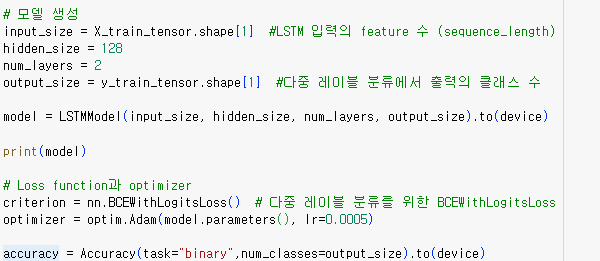
* 데이터를 학습용과 테스트용으로 분할하고, 그 비율은 8 대 2가 되도록 한다.



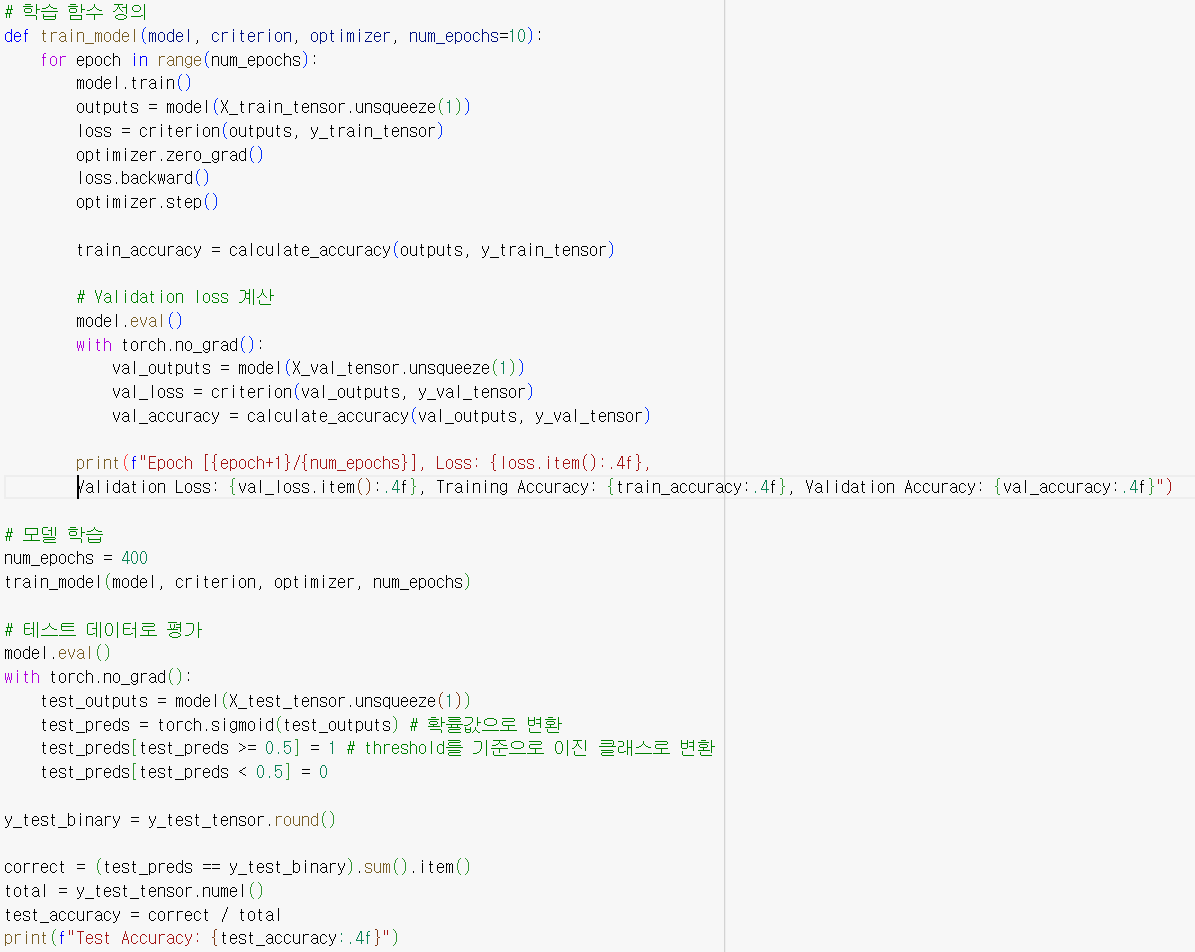
* 입력 데이터를 2D 배열로 변환하고, 레이블은 one-hot-encoding 방식으로 변환한다.



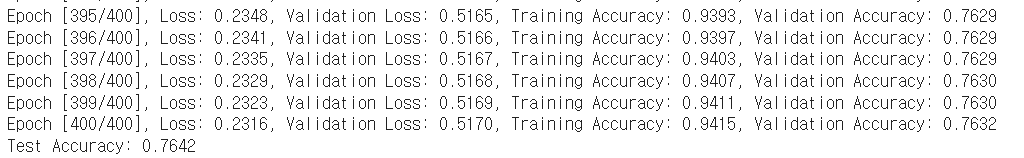
* 데이터 변환 후, 정확도를 나타내기 위한 함수를 설정한다.



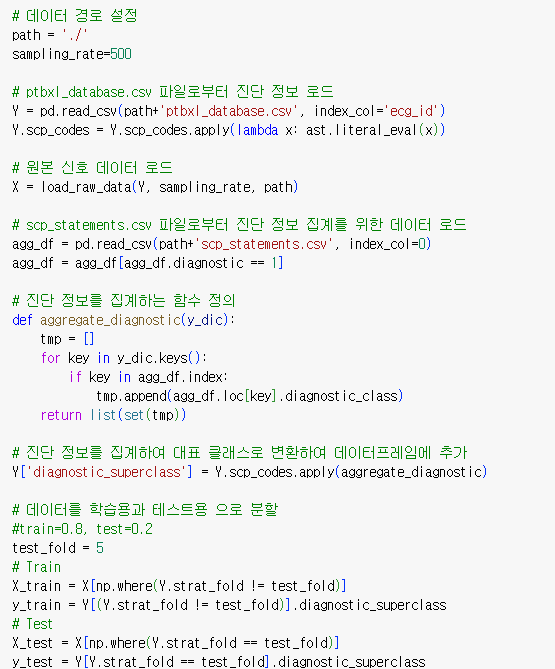
* 사용할 LSTM 모델을 정의하고, Layer 개수, Hidden\_size, Output\_size를 정의하고, Validation loss를 나타내기 위한 Loss function, Optimizer를 설정한다.



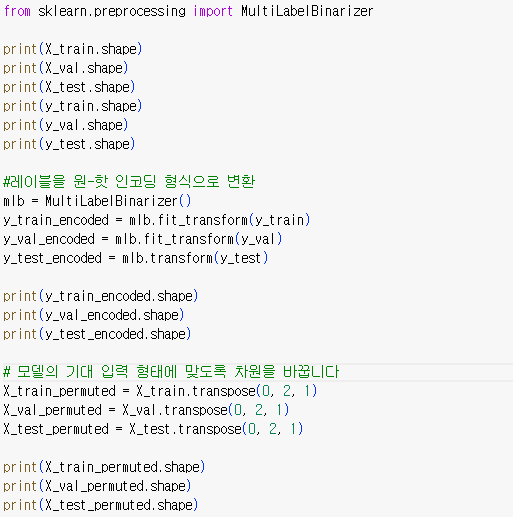
* 학습 함수를 정의하고, epoch를 400으로 설정하고 정확도와 loss를 나타낼 수 있도록 설정한다.



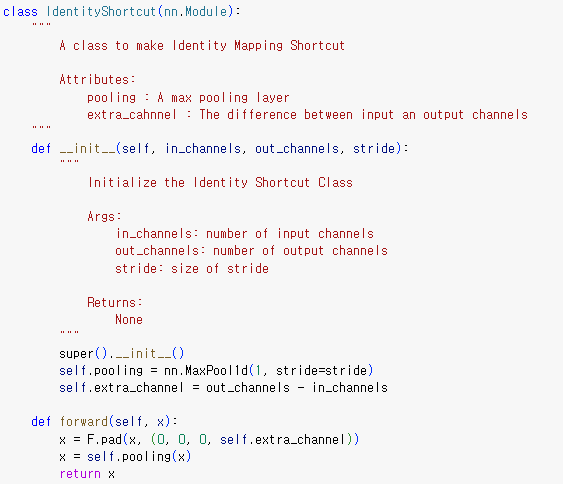
**2-3 ResNet(CNN) 모델**



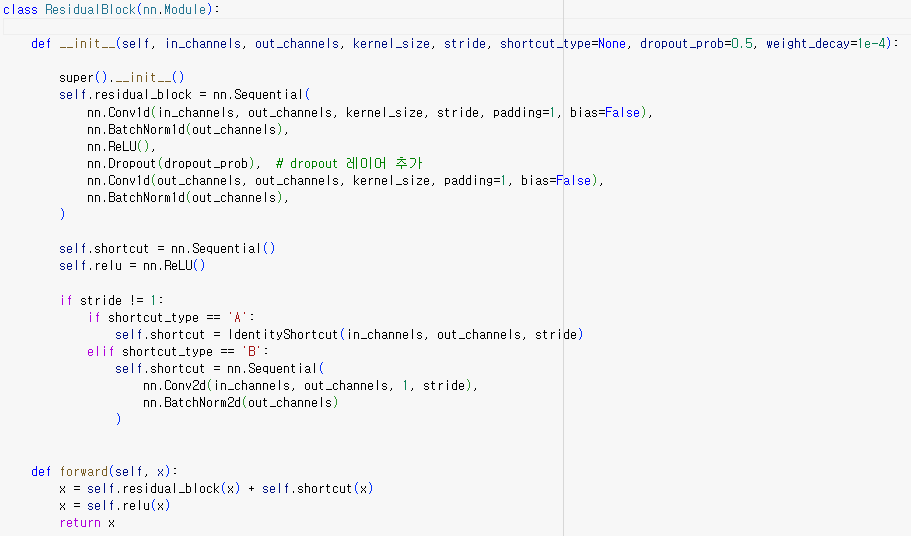
* 먼저 데이터를 업로드한다.



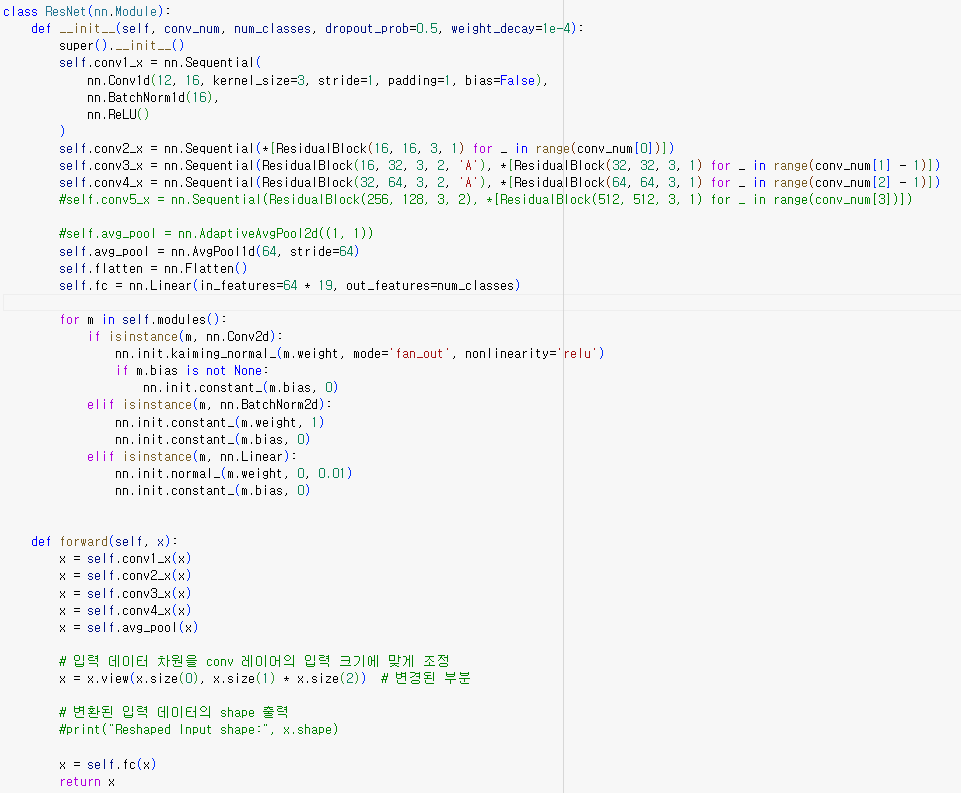
* 레이블을 원 핫 인코딩 방식으로 변환한 뒤, 데이터를 모델이 처리할 수 있는 형태로 맞추기 위하여 차원의 순서를 조정한다. 데이터의 차원을 조정함으로써, 모델의 입력 형태에 맞추어 데이터를 변환할 수 있게 된다.



* 이 클래스는 주어진 입력과 출력의 차원이 같은 경우, 입력을 그대로 출력으로 매핑해주는 역할을 한다. nn.MaxPool1d를 이용하여 1D Max Pooling 레이어를 만들며, 입력과 출력의 채널 차원이 달라지는 경우에는 영행렬을 패딩해 차원을 일치시켜 준다.



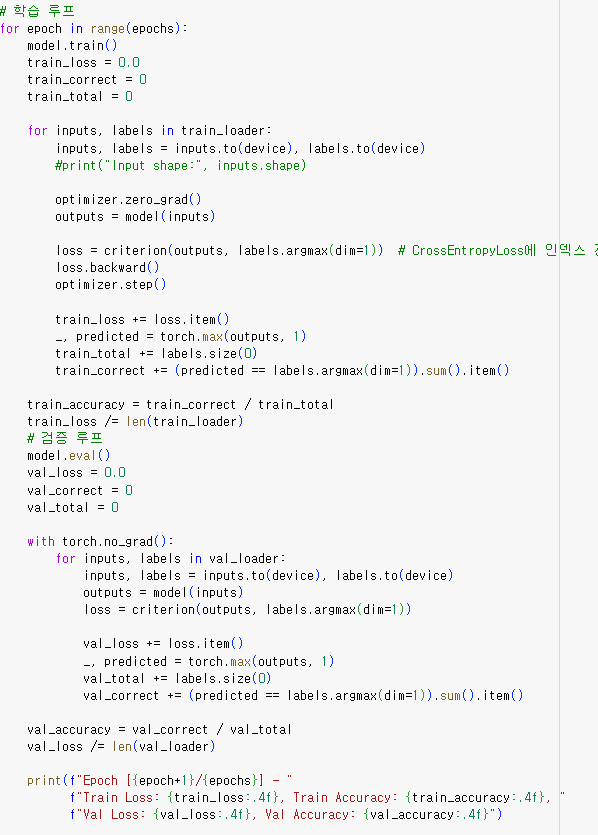
* 이 클래스는 Residual Block을 정의하는 역할로, 입력값을 그대로 출력값에 더하는 Residual connection을 구현한다. nn.Sequential 컨테이너를 사용하여 순차적으로 레이어를 구성하며, 입력과 출력의 차원을 고려한 Convolution, 배치 정규화, 활성화 함수, 드롭아웃을 포함한다.



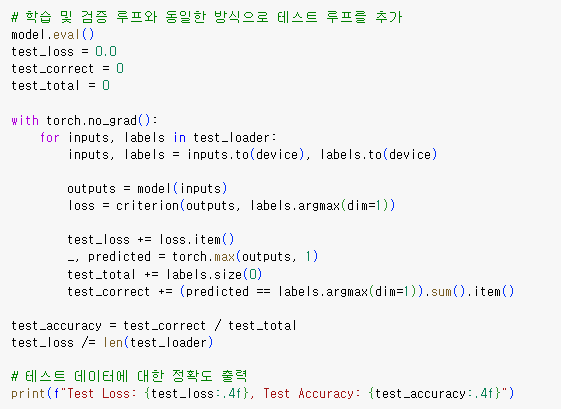
* 이 클래스는 전체 Residual Network를 구성하는 역할을 하며, 여러 Residual Block을 조합한 깊은 신경망 구조이다. conv1\_x, conv2\_x, conv3\_x, conv4\_x 등의 변수로 여러 층의 residual 블록을 구성하고, 마지막에 평균 풀링 레이어와 완전 연결 레이어를 추가하여 최종 분류 결과를 얻는다. 모델을 초기화할 때 가중치 초기화 등을 수행하며, forward 함수를 통해 실제 데이터가 모델을 통과하며 결과를 출력한다.



* 학습률, batch\_size, epoch를 포함하는 하이퍼 파라미터를 설정한 뒤, 데이터를 reshape 한다.
* ResNet 클래스를 사용하여 모델을 초기화하고 필요한 하이퍼 파라미터를 설정하여 생성하고, 손실 함수와 옵티마이저를 정의한다.



* 학습 루프를 통해 주어진 epochs 만큼 반복하여 모델을 학습하고, 검증 루프를 통해 손실과 정확도를 계산하여 출력한다.

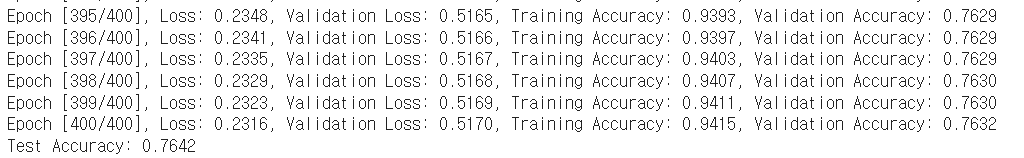


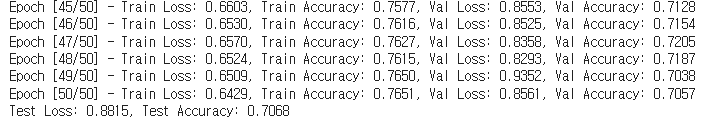
* 모델을 평가 모드로 설정하고 테스트 데이터 미니 배치를 반복하여 예측을 수행하고 손실을 계산한다. 이를 통해 평균 손실과 정확도를 계산하고 출력합니다. 평가 모드에서는 그레디언트 계산이 비활성화되며, 모델의 실제 성능을 확인할 수 있다.

1. **보고 시점까지의 과제 수행 내용 및 중간 결과**

LSTM, ResNet의 2가지 모델을 시도해 보았고, 각각의 test accuracy는

LSTM -약 76퍼센트/ ResNet - 약 70퍼센트

(LSTM)



(ResNet)

결론적으로 ResNet보다 LSTM이 더 높은 정확도를 보여주었다. 하지만 LSTM과 ResNet 모두 만족할 만한 정확도를 보여주지 못했는데, 이를 보완하기 위해 앞으로 하이퍼 파라미터 튜닝 및 모델 아키텍처 수정, 오버 피팅 방지에 힘을 쓸 계획이다.

1. **갱신된 과제 추진 계획**

위 2가지 모델을 통해 주어진 ECG 데이터를 학습시킨 결과, LSTM이 가장 높은 정확도를 보여주었고, 적합한 방법이라고 판단되었다. 하지만 지금의 모델은 만족스러운 결과를 가져오지 못했는데, 이를 보완하기 위해 몇 가지의 계획을 세우게 되었다:

1. 먼저 기본적으로 Accuracy가 0.76 부근에서 더 이상 증가하지 않는 모습을 보였다. hyper parameter을 조정하였음에도, 유의미하게 증가하는 경향을 보이지 않았기 때문에 다른 방법을 찾아 봐야 한다.
2. Validation loss가 너무 크다. epoch 값이 증가함에 따라, validation loss는 U자형 그래프를 나타내게 되는데, overfitting이 일어나기 시작하는 지점에서의 loss 값이 0.5를 상회한다. optimizer를 바꾸는 등의 시도를 해보았으나, 오히려 loss 값이 늘어나거나, accuracy가 비정상적으로 낮아지는 등의 문제가 발생하였다. 이는 loss function을 저장한 criterion 값이 accuracy를 측정하는 함수에 영향을 주기 때문이라 생각된다. 따라서 loss function을 바꾸는 쪽으로 방향성을 설정하였다.
3. LSTM 모델의 특징은 Layer의 개수를 바꾸어 원하는 기능을 작동하게 하는 것인데, Layer의 수를 단순히 증가시키는 것만으로는 더 나은 결과물을 도출하지 못하였다. Layer 수를 늘림과 동시에, 함수를 추가하거나, 특정 값을 변경함으로써 좋은 결과물을 도출하는 방향으로 계획하였다.
4. ResNet(CNN) 모델의 경우 모델의 복잡도가 부족하여 모델의 전체적인 Accuracy가 잘 나오지 않는데, 이 부분은 Convolutional Layer를 추가하여 모델의 복잡도를 증가시키고, 다양한 하이퍼 파라미터를 수정하여 Accuracy를 높일 계획이다
5. **구성원별 진척도**

