**Pusan National University Computer Science and Engineering Technical Report 2023-10**

**심장질환 환자 ECG 데이터 분석을 위한 딥러닝 기법 설계**



201824447 김정무

201824455 김지윤

201824601 천효승

지도교수 송길태

목 차

[1. 과제 개요 1](#_Toc41926412)

[1.1. 과제 배경 1](#_Toc41926413)

[1.2. 과제 목표 2](#_Toc41926414)

[2. 설계과정 및 배경지식 3](#_Toc41926416)

[2.1. 요구 조건 3](#_Toc41926417)

[2.2. 머신 러닝 모델(Random Forest) 4](#_Toc41926418)

[2.3. 딥 러닝 모델(RNN) 6](#_Toc41926418)

[2.4. 딥 러닝 모델(CNN) 8](#_Toc41926418)

[2.5. Pytorch 11](#_Toc41926418)

[2.6. ROC AUC/ROC Curve 11](#_Toc41926418)

[2.7. ECG(Electrocardiogram) 12](#_Toc41926418)

[2.8. 기존 제약사항에 대한 수정사항 및 한계점 14](#_Toc41926418)

[3. 연구내용 및 결과 16](#_Toc41926419)

[3.1. RNN 16](#_Toc41926423)

[3.2. ResNet(CNN) 26](#_Toc41926424)

[4. 결론 및 향후 연구 방향 32](#_Toc41926427)

[5. 개발 일정 및 역할 분담 36](#_Toc41926428)

[6. 참고 문헌 37](#_Toc41926429)

# 과제 개요

## 과제 배경

* 통계청의 2017년 사망원인 통계에 따르면 과거에 비해 심장 질환에 의한 사망률이 높아졌다. 여성과 남성 모두 심장질환이 악성 신생물에 이어 두 번째로 높은 사망 원인으로 추정하고 있다. 또한, 심장 질환은 전체 사망 원인의 10.8%를 차지하고 있는데, 이러한 심장질환의 예시로는 부정맥, 심근경색, 심낭 압전, 대동맥 파열, 폐동맥 색전증이 있다. 심장질환에 의한 돌연사, 급사의 90%를 차지하는 것은 부정맥이다.

 최근 보편화된 스마트 웨어러블 기기를 통해 많은 정보를 얻을 수 있는데, 그중 하나가 심전도(ECG, electrocardiogram) 값을 얻는 것이다. 심전도 정보를 통하여 심장의 비정상적인 리듬을 감지하고, 각종 심장병을 진단할 수 있다.

 부정맥의 진단은 심전도 데이터를 측정해서 할 수 있다. 표준 12유도 심전도 검사법과 같은 심전도 검사를 통해 얻은 심전도 기록을 기반으로 전문의가 진단을 내리는 것이 일반적인 부정맥의 진단 방법이다. 다만, 부정맥이 만성으로 진행되기 전에 불규칙적으로 나타나는 경우가 있어, 병원에 내원해 측정한 심전도 기록에서 부정맥을 진단하지 못하게 되는 문제가 발생할 수 있다.

 바이오인포매틱스가 큰 주목을 받는 현재, 인력 비용을 최소화하면서 동시에 대용량의 심전도 데이터에 대한 정확한 진단이 가능한 인공지능의 개발이 필수적이다. 심전도는 심장의 신호를 알려주는 굉장히 정확한 지표이다. 이를 사용하여 부정맥을 정확하게 진단해 낼 수 있는 인공지능을 개발한다면 추후 사용되는 인력을 대폭 줄일 수 있을 것이고, 디지털 의료기기를 한 단계 더 발전시킬 수 있을 것이며, 의료 보험에도 긍정적으로 작용할 것이다.

텍스트, 컴퓨터, 전자제품, 스크린샷이(가) 표시된 사진

자동 생성된 설명

그림 1. 심전도 그래프

## 과제 목표

* 우리 조는 방대하게 쌓인 심전도 데이터를 바탕으로 환자의 현 심장 상태를 진단할 수 있는 머신 러닝 및 딥 러닝 모델을 다양한 기법을 통해 제작하고, 이를 고도화시켜 최대한 높은 정확도를 얻기 위한 목표를 가지고 이 과제에 임하게 되었다. Tree 기반의 머신 러닝 분류 모델, RNN(Recurrent Neural Network) 기반의 딥 러닝 모델, CNN(Convolutional Neural Network) 기반의 딥 러닝 모델 총 세 가지의 기법으로 모델을 생성한 뒤 각 모델의 실행 시간, 정확도(Accuracy), ROC AUC(Receiver Operating Characteristic Area Under the Curve) 등을 시각화하고 비교하여 상황에 따라 어떤 모델이 가장 적합한지 판단하는 방향성을 설정하였다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

그림 2. 과제 목표

# 설계과정 및 배경지식

## 요구 조건

* PTB-XL ECG dataset[1]을 통해 Pytorch 기반의 머신 러닝 및 딥 러닝 모델을 제작한다. Random Forest, RNN, ResNet(CNN) 등 다양한 방법의 머신 러닝 및 딥 러닝 모델을 제작하고, 모델 간의  실행 시간, 정확도, ROC AUC를 시각화하여 비교하기로 하였다. 주 목표는 모델을 고도화시켜 진단 능력을 최대화하는 것이다. 데이터가 23,000명이 넘는 환자들의 ECG 데이터를 포함하고 있고, 데이터의 크기도 약 3GB로 매우 큰 용량을 차지하기 때문에 google Colab에서 프로그램을 돌릴 때 google Drive에서 원본 데이터를 불러오는 것이 불가능하였다. 따라서 모델 제작 및 하이퍼파라미터 튜닝을 할 때 데이터의 크기를 줄여서 학습과 테스트를 진행하였다. 데이터의 크기가 과도하게 작으면 학습이 제대로 이루어지지 않기 때문에, 데이터의 크기를 환자 500명분부터 2000명분까지 키워 가며 실행하고, 모델의 적합성을 판단했다. 축소된 데이터를 이용하여 Colab 환경에서 모델 제작 및 하이퍼파라미터 튜닝을 한 뒤, 원본 데이터와 각 모델을 학교 연구실 서버의 Jupyter Lab 환경으로 실행하였다.

폰트, 로고, 그래픽, 상징이(가) 표시된 사진

자동 생성된 설명

폰트, 로고, 그래픽, 텍스트이(가) 표시된 사진

자동 생성된 설명

그림 3. 실행 환경

## 머신 러닝 모델(Random Forest)

**스크린샷, 원, 다채로움, 그래픽이(가) 표시된 사진

자동 생성된 설명**

그림 4. Random Forest 구조(Tree)

텍스트, 스크린샷, 원, 폰트이(가) 표시된 사진

자동 생성된 설명

그림 5. Random Forest 실행 과정

* 머신 러닝 기반 분류 모델은 데이터를 기반으로 패턴을 학습하고, 주어진 입력 데이터를 여러 클래스 또는 범주 중 하나로 분류하는 모델이다. 그 중, Random Forest[2] 모델은 머신 러닝 기법 중 하나인 앙상블 학습(Ensemble Learning)의 대표적인 예시 중 하나로, 여러 개의 결정 트리(Decision Tree)를 결합하여 분류 또는 회귀 모델을 구축하는 방법이다. Random Forest는 데이터의 특성과 레이블 사이의 복잡한 관계를 모델링하고 예측하는 데 사용되며, 각 결정 트리는 데이터의 subset을 기반으로 학습된다. 이후 이러한 결정 트리의 결과를 조합하여 최종 예측을 수행하게 된다.

텍스트, 폰트, 화이트, 라인이(가) 표시된 사진

자동 생성된 설명

그림 6. 각 트리의 특성 중요도를 구하는 수식

* 각 트리의 특성 중요도는 그림 6.의 식과 같이 일반화된다.
* 여기서 RFfii란, Random Forest 모델로 얻은 모든 트리로 계산된 i 특성의 중요도이고,
* normfiij란, i 특성의 일반화된 특성 중요도를 의미한다.

## 딥 러닝 모델(RNN)

스크린샷, 도표이(가) 표시된 사진

자동 생성된 설명

그림 7. RNN 구조

* 데이터가 심전도의 시계열 데이터이기 때문에, 시계열 데이터 처리에 유용한 RNN 기반의 모델이 적합하다고 판단되어 RNN 모델을 제작하였다.
* RNN[3](Recurrent Neural Network)은 순차적인 데이터, 시계열 데이터 또는 자연어와 같은 순차적인 데이터를 처리하기 위한 인공 신경망의 한 유형이다. RNN은 순차 데이터를 처리하는 데 적합하며, 이전 타임 스텝의 출력이 다음 타임 스텝의 입력으로 사용된다. 이를 통해 순차 데이터에서 나타나는 시간적 또는 순서적 패턴을 파악할 수 있다. 또, RNN은 내부가 순환(재귀) 구조로 되어 있으며, 이전 타임 스텝에서 계산된 정보를 다음 타임 스텝으로 전달하게 된다. 이러한 재귀 구조를 통해 RNN은 이전 정보를 "기억"하고 현재 입력과 결합하여 출력을 생성한다.

타이포그래피, 폰트, 서예, 텍스트이(가) 표시된 사진

자동 생성된 설명

그림 8. hidden state의 계산 과정

* RNN에는 hidden state라는 핵심적인 변수가 존재한다(ht). hidden state ht란, t라는 시간에서의 상태를 의미한다. hidden state는 이전 hidden state와 input에 weight를 곱한 값을 더해서 계산되며, RNN내부의 메모리 역할을 하여 이전 정보를 현재 시간 스텝으로 전달하고 새로운 정보를 포함하는 역할을 한다.
* RNN의 변형에는 LSTM(Long Short Term Memory)과 GRU(Gated Recurrent Unit)의 두 가지가 있는데, 두 모델 모두 RNN의 장기 의존성 문제와 소실 문제를 극복하기 위해 나온 모델이다.

**스크린샷, 만화 영화, 원이(가) 표시된 사진

자동 생성된 설명**그림 9. GRU unit

* 먼저 GRU모델은 LSTM에 비하여 게이트 수가 적은 것이 특징인데, 리셋 게이트(Reset Gate)와, 업데이트 게이트(Update Gate)의 두 가지 주요한 게이트들로 구성된다. 구조가 간단하여, 빠른 학습 시간을 가진 것이 특징이다.

**스크린샷, 만화 영화, 원, 텍스트이(가) 표시된 사진

자동 생성된 설명**그림 10. LSTM unit

* LSTM모델은 셀 상태(Cell state), 입력 게이트(Input Gate), 망각 게이트(Forget Gate), 출력 게이트(Output Gate)의 구성요소로 이루어져 있으며, GRU모델에 비하여 비교적 복잡한 모델이다. 따라서 보다 더 많은 파라미터를 이용할 수 있으며, 의존성을 잘 학습하고 이를 표현하는 표현력 또한 뛰어나다.

## 딥 러닝 모델(CNN)

**도표, 라인, 스크린샷, 그래프이(가) 표시된 사진

자동 생성된 설명**그림 11. CNN 구조

* 기존 CNN으로는 만족스러운 결과를 얻을 수가 없어서 기존 CNN을 개선한 ResNet을 기반으로 모델을 제작하였다.
* CNN(Convoluted Neural Network)은 딥러닝의 신경망 아키텍처이다. CNN은 영상에서 객체, 클래스, 범주 인식을 위한 패턴을 찾을 때 유용하고, 오디오, 시계열 및 신호 데이터를 분류하는 데도 매우 효과적이다.

 시계열 데이터는 주로 시간에 따라 변하는 데이터이다. CNN을 적용할 때, 이미지의 공간적인 차원 대신 시간 차원을 고려해야 한다. 이를 위해 입력 데이터는 1D 또는 2D 형태로 구성되는데, 1D 시계열 데이터의 경우, 각 스텝이 입력의 열로 표현될 수 있으며, 2D로 변환하려면 여러 시계열 데이터 샘플을 열로 스택 할 수 있다.

 CNN은 합성곱 연산을 사용하여 데이터의 지역적 패턴을 추출하는 데 사용된다. 시계열 데이터에 CNN을 적용할 때는 시간 차원에서 작동하는 1D 합성곱 필터를 사용한다. 이러한 1D 합성곱 필터는 시간적 패턴을 감지하고 각 스텝에서의 특징을 추출한다.

 시계열 데이터에도 풀링 레이어를 적용할 수 있는데, 주로 최대 풀링(Max Pooling)이 사용되며, 시계열의 시간 차원을 다운샘플링하여 계산량을 줄인다. CNN은 다층 신경망 구조를 가질 수 있으며, 여러 개의 합성곱 레이어와 풀링 레이어를 쌓아 사용한다. 이를 통해 모델은 시계열 데이터의 복잡한 패턴을 학습하고 추상화할 수 있다.

 CNN은 파라미터 공유를 통해 시간 차원에서도 효과적으로 특징을 학습한다. 이전 타임 스텝의 정보를 현재 타임 스텝에서 재사용하므로 시계열 데이터의 지역적 패턴을 인식하는 데 도움이 된다. 심전도 데이터는 시간에 따라 변하는 심장박동을 기록한 시계열 데이터이며, CNN을 사용하여 각 스텝에서의 심전도 패턴을 분석하는 데 활용할 수 있다.

* ResNet[4](Residual Network)은 딥 러닝에서 사용되는 신경망 아키텍처 중 하나로, 딥러닝 모델의 깊이를 키우면서 발생하는 그래디언트 소실 문제를 해결하기 위해 개발되었다.

 ResNet은 Microsoft Research에서 개발되었으며, 2015년 ImageNet 이미지 분류 대회에서 우승한 아키텍처이다. ResNet의 핵심 아이디어는 "잔차 학습" 또는 "스킵 연결"이다. 일반적인 딥 러닝 모델에서는 입력 데이터가 각 레이어를 거치면서 변환되고, 중간 레이어에서 학습된 표현이 최종 출력 레이어로 전달된다. 그러나 깊은 네트워크에서는 그래디언트가 역전파되는 동안 중간 레이어에서 소실되거나 폭발하는 문제가 발생할 수 있다. ResNet은 이 문제를 해결하기 위해 스킵 연결을 도입하게 된다. 각 레이어는 이전 레이어의 출력에 현재 레이어의 출력을 더하는 형태로 연결된다. 이를 통해 그래디언트가 레이어를 거치면서 소실되지 않고 흐를 수 있다. 이러한 스킵 연결은 잔차(Residual)라고도 불린다.

 ResNet의 주요 특징 및 이점은 다음과 같다:

* 깊은 네트워크 구조: ResNet은 매우 깊은 신경망을 구축할 수 있으며, 이를 통해 더 복잡한 패턴과 특징을 학습할 수 있다.
* 그래디언트 소실 문제 해결: 잔차 연결로 인해 그래디언트가 원활하게 전달되므로, 더 깊은 네트워크에서도 학습이 가능해진다.
* 종단 간 추론 (End-to-End Inference): 스킵 연결로 인해 입력에서 출력까지의 경로가 짧아지므로 종단 간 추론이 용이해진다.

**<Residual Block architecture in ResNet>**

텍스트, 도표, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

그림 12. Residual Block의 구조

* Resnet 의 핵심 아이디어 중 하나는 잔여 함수를 도입하여 스킵 연결을 사용한다는 것인데, 여기서 잔여 함수는 R(x) = F(x) - x 라고 표현할 수 있다. 여기서 x는 입력, F(x)는 변환된 값인데, n개의 layer을 거치면서 합성곱 연산을 통해 Fn(x)를 생성하고, 이를 스킵 연결을 통해 계산하여 그래디언트 소실 문제를 피하고 학습을 용이하게 한다.

## Pytorch[5]

* Python을 위한 오픈소스 머신 러닝 라이브러리로, cuda 명령어를 통하여 GPU를 사용할 수 있기 때문에 보다 빠른 정보 처리가 가능하며, tensor 계산을 수행한다. 동적 계산 그래프를 기반으로 하는 특징을 가지고 있어, 그래프를 구축하고 구성하기가 직관적이고 유연하다. 다양한 도구와 라이브러리들이 지원되어 확장성이 높다.

텍스트, 스크린샷, 도표이(가) 표시된 사진

자동 생성된 설명

그림 13. Pytorch 구조

## ROC AUC / ROC Curve

* ROC 곡선은 이진 분류 모델의 성능을 시각적으로 평가하는 데 사용되는 표준 도구이다. 그래프의 x축은 FPR(False Positive Rate)을 의미하고, y축은 TPR(True Positive Rate)를 의미한다.
* ROC AUC는 ROC곡선의 아래 영역을 의미하고,  모델의 성능을 단일 숫자로 나타내는 데 사용한다. ROC AUC가 1에 가까울수록 이상적인 모델이라고 할 수 있다.

도표, 라인, 텍스트, 그래프이(가) 표시된 사진

자동 생성된 설명

그림 14. ROC AUC

## ECG(Electrocardiogram)

**2.7.1. 표준 12유도 심전도 기법**

텍스트, 폰트, 흑백, 친필이(가) 표시된 사진

자동 생성된 설명

그림 15. 12유도 기법의 각 전압 정보

* 12유도 심전도는 가슴에 여러 개의 전극을 부착하여 심장의 전기 활동을 다양한 각도에서 측정하는 방법이다. 12유도 심전도는 총 10개의 전극을 사용하여 몸의 특정 위치에 부착하고, 심장의 전기적 신호를 측정하고 각 리드의 값을 계산한다.

**2.7.2. 진단 클래스**

* 진단 클래스에는 크게 NORM, CD, STTC, MI, HYP의 5가지 종류가 있다.

**NORM** : 정상적인 심전도를 나타내는 클래스로 일반적인 심박수와 심장 리듬이 관측됨.

**CD** : 전도장애를 나타내는 클래스. 전기적 전달이 느리거나 심장리듬의 변화가 관측됨.

**STTC** : 심장의 신장성 빈맥이 빠르게 발생하여 심박수가 높아진 상태.

**MI** : 심근경색을 나타내는 클래스. 혈액공급이 감소하여 심장 근육의 일부분이 손상된 상태.

**HYP** : 심근비대를 나타내는 클래스. 심장 근육이 비정상적으로 두꺼워지는 상태.

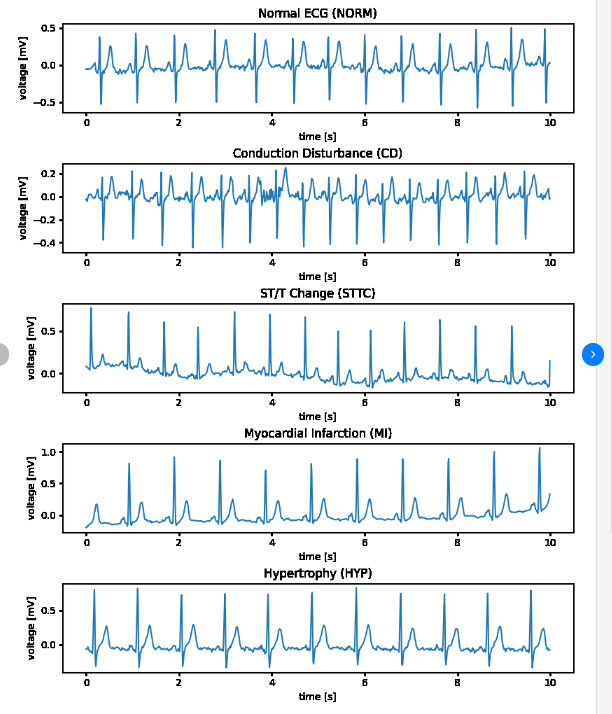


그림 16. 진단 클래스의 종류

## 기존 제약사항에 대한 수정사항 및 한계점

**2-8-1. google Colab 환경에서의 데이터 불러오기**

* google Colab 환경에서 Pytorch를 사용하게 되면 데이터를 google Drive에서부터 가져오게 되는데, 전체 데이터의 크기가 너무 커서 불러올 수 없는 문제가 있어 작은 데이터셋을 불러온 뒤 Colab 환경에서 테스트하고, 전체 데이터는 연구실의 jupyterLab 환경으로 돌리게 되었다.

텍스트, 전자제품, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**2-8-2. 추가적인 모델 설계**

* RNN과 CNN같은 딥 러닝 외의 머신 러닝 모델과도 비교하기 위해서 XGBoost, LightGBM, SGD 등의 여러 모델들을 설계해본 후 가장 정확도가 잘 나온 Random Forest 모델을 추가하였다.

**2-8-3. 모델의 크기 문제**

* Random Forest 모델과 RNN 모델의 경우 모델의 크기가 서버 RTX3090의 24GB를 크게 상회하는 모습을 보여 GPU 환경에서 프로그램을 돌리지 못하였고, 따라서 CPU 환경에서 돌리게 되었다.



* 또, RNN 모델의 경우 sampling rate가 500일 때도 용량 문제로 인해 CPU 환경에서 돌릴 수 없어 sampling rate를 100으로 낮춘 뒤 실행하였다.

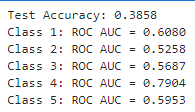
**2-8-4. ResNet Convolutional layer 추가**

* 중간 보고서에서 제작했던 ResNet의 convolutional layer는 1부터 4까지 있었지만, 정확도가 생각보다 잘 나오지 않아 5번째 convolutional layer를 추가시켜 모델을 더욱 고도화하였다.

****

\* 기존 ResNet에서는 주석처리 되어 있었다.

**2-8-5. Random Forest 모델의 한계**

* Random Forest 모델은 적은 데이터셋으로는 꽤나 만족스러운 결과가 나왔으나, 전체 20000여명 분의 데이터를 대입했을 때, overfitting의 문제 때문에 예상보다 매우 낮은 결과가 나오게 되어 딥 러닝 모델의 고도화에 더욱 집중하기로 하였다.

\* 기존 Random Forest를 전체 데이터로 돌릴 경우 test 결과가 매우 낮았다.

# 연구내용 및 결과

## RNN

**<파일 불러오기 및 기초 전처리>**

****

* Colab 환경에서 진행하기 위해서 파일을 불러오고, 기본적인 전처리를 진행하는 과정이다.

**<Training set 분리>**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

* Train 및 validation, test set을 정해진 비율로 나눈다. strat\_fold가 10인 부분을 test data로 사용하기로 했고, validation set은 기존 train set의 0.25의 비율로 설정해 주었다.

**<데이터 변환 및 인코딩>**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 폰트, 스크린샷, 타이포그래피이(가) 표시된 사진

자동 생성된 설명**

* 학습 이후 성능을 측정하기 위해 MultiLabelBinarizer 라이브러리를 이용하여 원-핫 인코딩을 진행한다. 기존 데이터는 바이너리 형식으로 매핑된다.

(ex. STTC, CD → [1][0][0][0][1])

* train, validation, test의 비율은 각각 약 0.67, 0.23, 0.10으로 분할된 것을 확인할 수 있다.

**<데이터를 tensor 형태로 변환>**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 폰트, 친필, 서예이(가) 표시된 사진

자동 생성된 설명**

* Pytorch의 입력 형태에 맞게 데이터를 PyTorch Tensor의 형태로 변환하는 과정을 거친다. 원 데이터 형태 그대로 float 형식을 사용하였고, 모델에 입력하기 위해 채널과 시퀀스 길이의 데이터 위치를 변환하였다.

**<fast.ai 라이브러리 import>**

****

* Pytorch 기반의 딥 러닝 모델을 사용하기 위해 fastai 라이브러리를 import 한다.

**<입력 데이터 정보 추출>**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

* 풀링 연산을 수행하여 입력 데이터의 정보를 추출하고, 결과를 결합하여 출력한다.

**<RNN모델 구축>**

**텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명**

* RNN1d클래스를 정의하고, RNN모델을 구축한다. 여러 레이어의 기능을 추가하고, 데이터의 차원 조정 및 풀링을 수행한다.

**<모델 선언>**

**텍스트, 폰트, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명**

* 입력 데이터의 채널 수는 표준 12유도 심전도 기법에서 사용되는 총 12가지의 전압 데이터( I, II, III, aVR, aVL, aVF, V1, V2, V3, V4, V5, V6)이다. 클래스는 다섯 가지의 진단인 STTC, MI, HYP, NORM, CD이다.
* 모델 선언은 GRU보다 LSTM이 더 좋은 성능을 보였기 때문에 LSTM을 사용하였고, activation function의 경우 ReLU보다 ELU가 더 좋은 성능을 보였기 때문에(ELU의 경우 음수 부분을 버리지 않음) ELU를 선택하였다. bidirectional의 여부 역시 bidirectional = True일 때 loss가 더 적게 학습되어 bidirectional을 선택하였다.

**<Loss function 및 optimizer>**

**텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명**

* 모델이 다중 레이블을 분류하므로 다중 레이블 분류에 사용되는 BCEWithLogitsLoss를 선택했고, optimizer의 경우 널리 사용되는 Adam의 단점을 보완한 AdamW를 learning rate = 0.005로 사용하였다.

**<Accuracy 계산 함수>**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

* accuracy를 계산하는 함수이다.

**<모델 학습>**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

* 모델을 학습하는 함수이다. epoch만큼 과정을 거치며 각 epoch의 train과 validation의 loss 및 accuracy를 계산하며 출력한다.

**<학습 시간 측정>**

**텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명**

* epoch은 하이퍼파라미터 튜닝 결과 120이 가장 적합하다고 판단되어 120으로 설정하였다.(120 epoch 이후부터는 overfitting 발생)
* 모델의 학습에 걸리는 총 실행 시간을 계산하기 위해 time 라이브러리를 사용하여 총 학습 시간을 측정하였다.

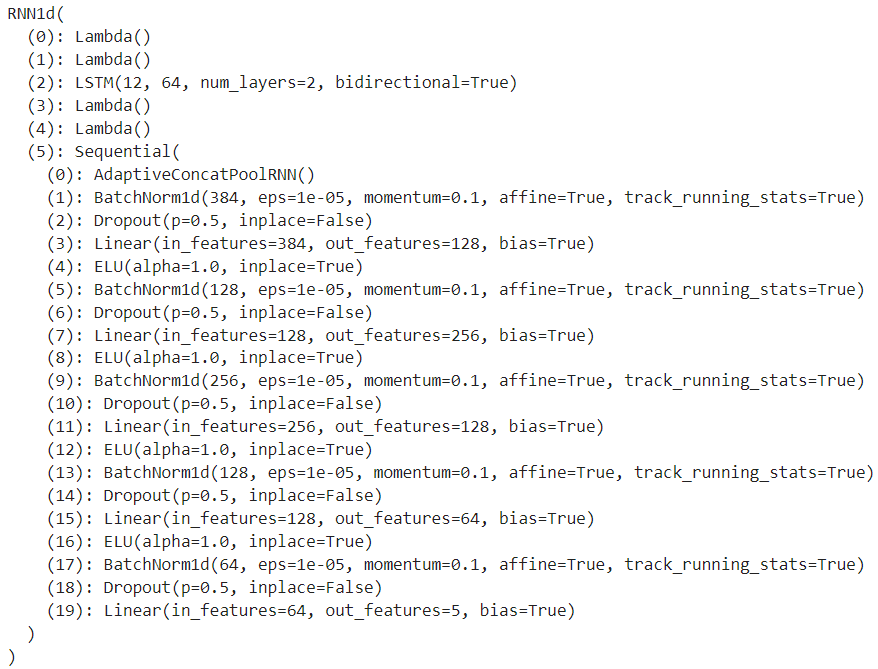
**<모델 평가>**

**텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명**

* 모델의 학습이 끝난 후, test data를 이용해 모델이 얼마나 잘 학습되었는지 평가한다.

**<모델 구조>**



* 모델의 구조는 다음과 같다. 128, 256, 128, 64의 레이어 크기 순서를 따르는 bidirectional LSTM 구조이다.

**<학습 및 평가 결과>**

텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명

* RNN모델의 test accuracy는 0.8755가 나왔고, 실행 시간은 8145초가 나왔다.

**<ROC AUC 출력>**

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

* 클래스 별로 ROC AUC를 출력하는 함수이다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* ROC Curve를 그리기 위해 각 클래스 별로 tpr(True Positive Rate), fpr(False Positive Rate)를 구하는 함수이다.

**<ROC Curve 출력>**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

* 하나의 그래프에 plt를 이용하여 각 클래스 별로 ROC Curve를 그려 시각화한다.

**텍스트, 스크린샷, 라인, 그래프이(가) 표시된 사진

자동 생성된 설명**

* RNN모델은 이와 같은 ROC AUC 결과를 얻을 수 있었다.

## ResNet(CNN)

**<cuda 사용>**

텍스트, 폰트, 라인, 화이트이(가) 표시된 사진

자동 생성된 설명



* ResNet은 다른 모델들에 비해 차지하는 용량이 비교적 적어 cuda(GPU) 환경에서 돌릴 수 있어 첫 부분에 cuda를 enable해 주는 코드이다. 그래픽카드(RTX 3090)의 총 사용 가능한 용량은 24576 MB로 나온다.

**<데이터 전처리>**

폰트, 텍스트, 화이트, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

폰트, 텍스트, 타이포그래피, 디자인이(가) 표시된 사진

자동 생성된 설명

* 데이터의 전처리 부분은 RNN과 동일하다. 전처리 이후 x와 y의 형태는 위와 같다.

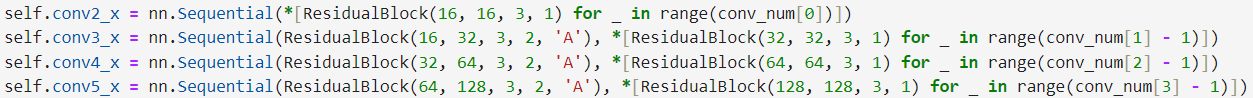
**<모델 - ResNet>**

****

* ResNet의 구조는 다음과 같다. 5개의 convolutional layer 단계를 거치며, average pooling을 적용하는 구조이다. 각 Residual Block의 형태는







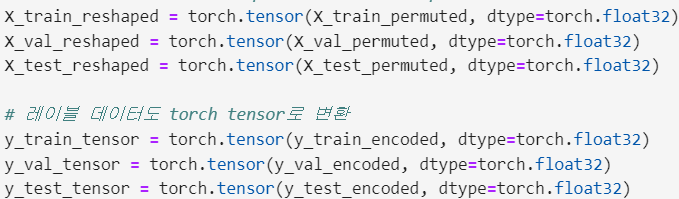
* 다음과 같이 12, 16, 32, 64, 128 순으로 설정하였다.

**<데이터 변수 선언>**



* input\_features 변수에는 입력 시퀀스의 크기인 5000이 들어가고, num\_classes 변수에는 총 클래스의 개수인 5가 들어가게 된다.

**<Pytorch Tensor로의 변환>**

****

* 각각의 분할된 데이터를 float의 형태로 Pytorch Tensor로 변환한다.

**<DataLoader 생성>**

**텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명**

* 변환한 데이터를 원활한 학습을 위해 DataLoader의 형식으로 변환한다.

**<모델 선언 및 하이퍼파라미터>**

**텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명**

* 모델은 ResNet101모델을 사용하였다. 조금 더 고도화 된 ResNet152를 사용하려 했으나, 실행 시간이 더 긴 반면에 비슷한 결과가 나오게 되어 ResNet101을 선택하게 되었다. ResNet101은 각각의 layer의 개수가 [3, 4, 23, 3]로 정해져 있는 구조이다.
* Overfitting을 방지하기 위해 0.2의 dropout을 추가했고, optimizer의 경우 Adam의 단점을 보완한 AdamW를 사용했다.
* criterion의 경우 RNN과 동일하게 멀티 레이블 분류를 위해 BCEWithLogitsLoss를 사용했다.

텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진

자동 생성된 설명

* 하이퍼파라미터 튜닝 결과, learning\_rate는 0.001이 가장 적합했고, batch\_size는 학습 속도를 높이기 위해 기존보다 큰 128로 설정했다. epoch은 50이후로는 overfitting이 일어나게 되어 50으로 설정했다.

**<모델 학습>**

****

* 학습도 RNN과 매우 유사하나, GPU환경 사용을 위해 모든 데이터를 cuda로 옮겨 주는 .to(‘cuda’) 를 사용했다.

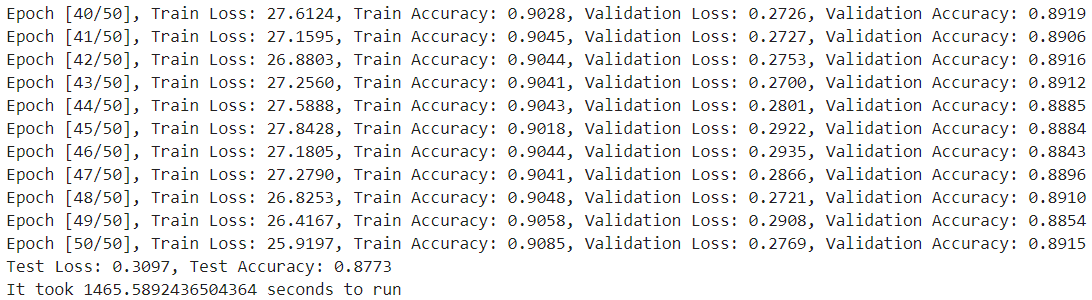
**<실행 시간 측정>**

**텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진

자동 생성된 설명**

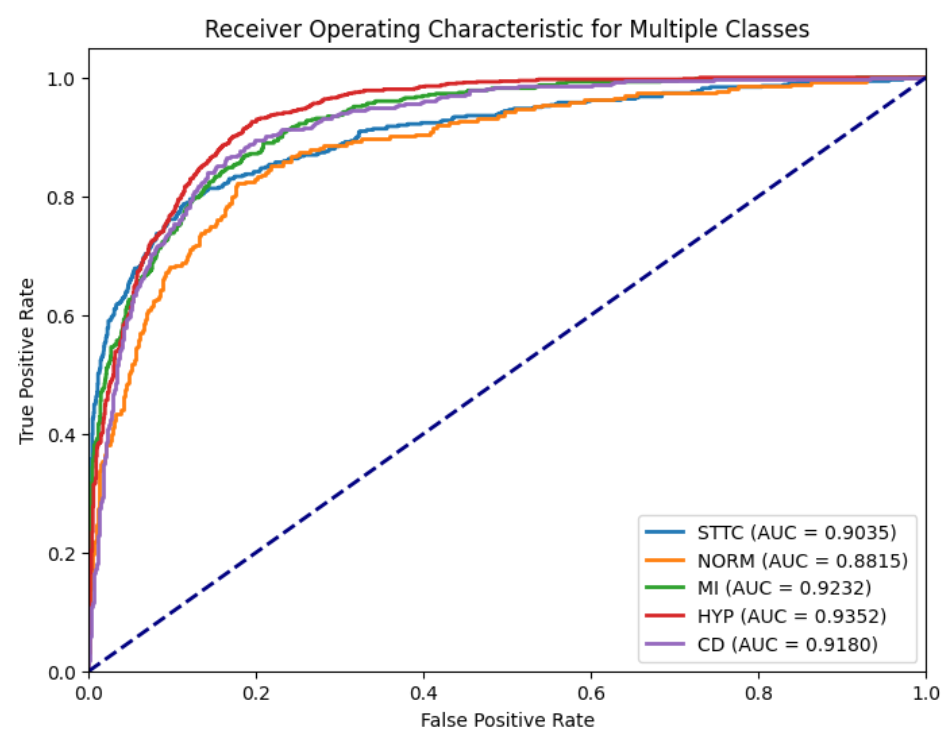
* 실행 시간의 경우에도 train을 시작하기 전과 후의 차를 구하여 학습에 걸린 시간을 측정했다.

**<학습 및 평가 결과>**

****

* ResNet모델의 test accuracy는 0.8773이 나왔고, 실행 시간은 1465초가 나왔다.

**<ROC AUC 및 ROC Curve>**

****

* ROC AUC를 계산하는 부분 및 ROC 곡선을 그리는 코드는 RNN과 동일하며, ResNet은 이와 같은 결과를 얻을 수 있었다.

# 결론 및 향후 연구 방향

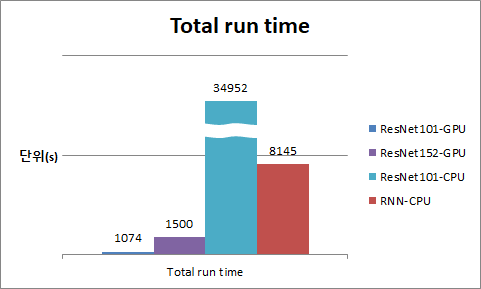
****

그림 17. 실행시간 비교

* 실행 시간의 경우 GPU환경에서 실행할 수 있는 ResNet모델이 가장 적게 걸렸으며, CPU환경에서 실행했을 때 10배 이상의 실행 시간이 걸리는 것으로 보아 머신 러닝/딥 러닝 모델은 GPU환경에서 실행하는 것이 가장 효율적임을 알 수 있었다.

**텍스트, 스크린샷, 번호, 도표이(가) 표시된 사진

자동 생성된 설명**

그림 18. 정확도 비교

**텍스트, 스크린샷, 폰트, 직사각형이(가) 표시된 사진

자동 생성된 설명**

그림 19. ROC AUC 비교(STTC)

**텍스트, 스크린샷, 폰트, 직사각형이(가) 표시된 사진

자동 생성된 설명**

그림 20. ROC AUC 비교(NORM)

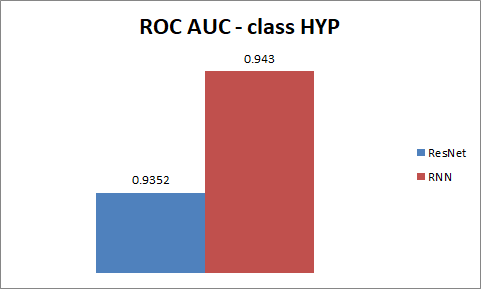


그림 21. ROC AUC 비교(MI)

텍스트, 스크린샷, 폰트, 직사각형이(가) 표시된 사진

자동 생성된 설명

그림 22. ROC AUC 비교(HYP)

텍스트, 스크린샷, 폰트, 직사각형이(가) 표시된 사진

자동 생성된 설명

그림 23. ROC AUC 비교(CD)

* 전체적인 성능은 ResNet모델이 RNN모델보다 근사하게 앞서지만, RNN 모델의 경우 데이터가 크면 클수록 모델의 크기가 감당할 수 없을 정도로 커지며, 실행 시간 역시 크게 길어진다는 단점이 있어 GPU 환경에서 실행할 수 있는 ResNet101 모델이 더 많은 데이터를 입력하거나, fine-tuning을 한다는 가정 하에 두 가지 모델 중 ECG 분석에 가장 적합해 보인다.
* 다른 기법의 모델들로도 꼭 진행해보면 좋을 것 같다. 최근 다양한 딥 러닝 분야에서 좋은 결과값을 얻는 Transformer, BERT등의 SOTA(State-of-the-art) 모델들로 모델을 설계해 본다면 보다 적은 실행 시간으로 더 좋은 결과를 얻을 수도 있을 것 같다. 또한, 만약 RNN기반 모델을 경량화하는 것이 가능하다면 GPU환경에서 실행해보는 것도 좋은 방법일 것 같다.

# 개발 일정 및 역할 분담

**텍스트, 스크린샷, 도표, 번호이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명**

# 참고 문헌

|  |
| --- |
| * [1] Patrick Wagner, Nils Strodthoff, Ralf-Dieter Bousseljot, Dieter Kreiseler, Fatima I. Lunze, Wojciech Samek & Tobias Schaeffter, “PTB-XL, a large publicly available electrocardiography dataset”, Journal of Nature : scientific data, Vol. 7 pp.154, May. 2020.(in English)   [2] Wikipedia. Randomforest - Wikipedia [Online]. Available: https://ko.wikipedia.org/wiki/%EB%9E%9C%EB%8D%A4\_%ED%8F%AC%EB%A0%88%EC%8A%A4%ED%8A%B8  [3] PyTorch RNN, [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.RNN.html  [4] PyTorch ResNet, [Online]. Available: https://pytorch.org/hub/pytorch\_vision\_resnet/  [5] PyTorch Documentation, [Online]. Available: <https://pytorch.org/docs/stable/index.html> |