[English]
This paper may not be uploaded anywhere other than https://github.com/incognito-developer/ECGanalysis and https://github.com/incognito-developer/ecgAnalysisUsingML without permission. As of October 19, 2023, this material cannot be processed or modified without the permission of incognito-developer, moonsae, blue125ml, and dleogh476. Distribution, unauthorized use, and theft without the permission of all four people mentioned above are prohibited. If it is difficult to contact all 4 people, incognito-developer will decide. If you are interested in this paper or want to use it, please leave a comment in the relevant GitHub repository as an issue or at https://incognito-developer.github.io/posts/2023-07-16-aboutEcgPaperAndInfomations.

This paper is material created during the project. This paper has not been reviewed. As a result, there may be technical errors or lack of evidence, and it may not fit the format of the journal. Therefore, this paper may be insufficient as academic material. Accordingly, we are not responsible for any problems arising from inaccurate content or inaccurate content in the paper. We are not responsible for any problems or damages arising from the use of this material.

If this paper contains problematic content, please let us know. We will then check and take action such as editing or deleting.

This paper only written by Korean, so you need to use translator If you want to see other language.

[Korean]
이 논문은 https://github.com/incognito-developer/ECGanalysis 과 https://github.com/incognito-developer/ecgAnalysisUsingML 외의 다른 곳에 허가없이 업로드를 금지합니다. 이 자료는 2023년 10월 19일 기준 incognito-developer, moonsae, blue125ml, dleogh476 모두의 허락없이 가공 및 변형을 허락하지 않습니다. 위에서 언급된 4명 모두의 허락없이 배포 및 무단 이용과 도용을 금합니다. 만약 4명 모두의 연락이 어렵다면, incognito-developer의 판단에 따릅니다. 만약 이 논문에 관심이 있거나 활용을 원하신다면, 깃허브 해당 리파지토리의 issues나 https://incognito-developer.github.io/posts/2023-07-16-aboutEcgPaperAndInfomations 의 댓글로 남겨주시길 바랍니다.

이 논문은 프로젝트를 진행하면서 생성된 자료입니다. 이 논문은 검토를 받지 않았습니다. 이에 기술적인 오류나 근거가 부족할 수 있으며, 학회지의 양식에 맞지 않을 수 있습니다. 이에 이 논문은 학술적 자료로서 부족할 수 있습니다. 이에 부정확한 내용이나 논문의 정확하지 않은 내용 등으로 발생한 모든 문제는 책임지지 않습니다. 이 자료를 이용함으로 인해 발생한 어떠한 문제나 피해도 책임지지 않습니다.

만약 이 논문에 문제되는 내용이 포함되어 있다면, 알려주십시오. 그러면 확인 후 수정이나 삭제 등의 조치를 하겠습니다.

이 논문은 한국어로만 제공되니, 다른 언어로 보고 싶으면 번역기를 사용하셔야 합니다.

머신러닝으로 심전도 데이터를 이용한 부정맥 분석

{incognito-developer[O], moonsae, blue125ml, dleogh476} at GitHub

Arrhythmia analysis using ECG data with Machine Learning

GitHub username{incognito-developer[0], moonsae, blue125ml, dleogh476}

요 약

심전도(ECG, electrocardiogram) 분석은 사람의 심장 상태 분석을 위해 매우 중요하다. 사람이 심전도를 분석할 시 시간이 오래 걸리고, 판독자의 피로도, 숙련도 등으로 인해 검사 결과에 영향을 받을 수 있다. 사람이 심전도를 분석하는 것을 대신하기 위해 컴퓨터가 자동으로 심전도(ECG, electrocardiogram) 분석을 하기 위한 방법은 매우 다양하다. 하지만 심전도의 복잡한 패턴으로 인해 분석에 한계가 존재하고, 오경보가 잦다. 이번 연구에서는 training 과 test 에 Kaggle 의 ECG Heartbeat Categorization Dataset[1]을 사용했다. CNN-LSTM, dropout, ensemble, transformer 등 다양한 방법을 적용해 ECG 데이터 분석 가능한 모델을 만들고, 각 모델 별 성능측정 비교를 했다. 이번 실험에 사용한 모델 중 최대 약 97%의 accuracy 를 보이는 CNN-LSTM 기반 모델이 가장 성능이 좋았다. 이렇게 만들어진 모델을 웹 기반으로 연동시켜, 일반인들까지 접근성을 향상시켰다. 논문과 관련된 프로젝트는 https://incognito-developer.github.io/posts/2023-07-16-aboutEcgPaperAndInfomations 에서 확인 가능하다.

1. 서론

부정맥은 심장이 정상적으로 뛰지 않는 것을 의미하며, 불규칙한 맥박뿐만 아니라 빠르거나 느린 것을 총칭하며, 다양한 증상을 나타낼 수 있다. 이러한 부정맥 증상을 조기에 감지하기 위해 심전도 검사 등 다양한 검사를 활용하며, 특히 24 시간 심전도 모니터링인 Holter monitoring 은 부정맥의 갑작스러운 발작을 탐지하는 데 사용된다. 그러나 이 방법은 많은 시간과 인력을 필요로 하며, 대량의 데이터 분석에도 한계가 있다.

빠른 인공지능의 발전으로 심장 파형을 의사가 판독하는 대신 인공지능을 이용하여 분석을 시도하고 있다. 심전도 분석을 인공지능이 대체하기 위해 여러 연구가 진행, 개선되고 있지만, 복잡한 패턴으로 인한 성능 문제 이외에도 법률적인 문제가 존재한다. 그래서 아직까지 의사의 판단을 대체하기 보단 판단 보조의 용도로 주로 쓰인다. 본 연구는 기계 학습을 활용하여 부정맥을 탐지하는 여러가지 방법을 시도하고, 성능지표들로 분석한다. 환자의 심전도 데이터를 기반으로 사전에 학습된 모델을 활용하여 부정맥을 식별함으로써, 부정맥의 정확한 진단을 도와준다. 기계 학습 기반의 부정맥 탐지 방법은 판독자의 검사 숙련도 차이로 인한 오진을 줄일 수 있으며 검사 결과 대기 시간을 줄이고 빠른 처치나 치료를 할 수 있어 부정맥 환자들에게 많은 도움이 될 것이다

2. 관련 연구

2.1 기존 연구와의 비교

기존에도 부정맥을 탐지할 수 있는 다양한 기법들이 연구되고 현재에도 딥러닝 등의 기법을 사용해 발전하는 분야이다. 한 예로, 중환자실 내 위급한 부정맥을 탐지하려고 한 연구[2]를 소개하겠다. CNN을 활용해 PhysioNet/CinC challenge 2015 데이터셋을 사용한 결과, Accuracy 88.67%, Sensitivity 89.47%, Specificity 88.03%의 성과를 이뤘다고 한다. 이번 연구는 심전도 데이터를 분석하여 부정맥을 탐지할 수 있는 모델들을 다양한 기법으로 제작하고 성능을 측정하는 것을 목적으로 한다. 이 모델은 사용자가 웹 사이트와 웹 서버를 통해 간편하게 데이터를 업로드하고 결과를 받을 수 있는 것이 목표이다. 이를 통해 병원에서의 분석 시간 감소와 의사의 피로도 감소를 이루고 환자들은 신속한 판독 결과를 통해 적절한 조치를 취할 수 있다.

원래 연구 계획은 일반인의 접근성을 높이기 위해 스마트 워치의 ECG 분석 기능을 사용하려 했다. 스마트 워치는 심전도 분석을 위해 lead-1 방식을 사용한다. Lead-1 방식과 Lead-2 방식은 그림 1 처럼 다른 양상을 보여 구분이 필요하다. Lead-1 방식의 심전도 데이터를 구하기 어려워, lead-2 방식 데이터로 모델 학습 진행 후 lead-2 방식 심전도 데이터를 분석하는 서비스를 만들었다. CNN, dropout, ensemble, transformer 등 다양한 기술을 적용한 모델들을 활용하여 부정맥 탐지 accuracy 를 최대 약 97%로 높였다. 이로써 본 연구는 심전도 데이터 분석 모델의 의료 현장 적용 가능성도 보일 수 있다.

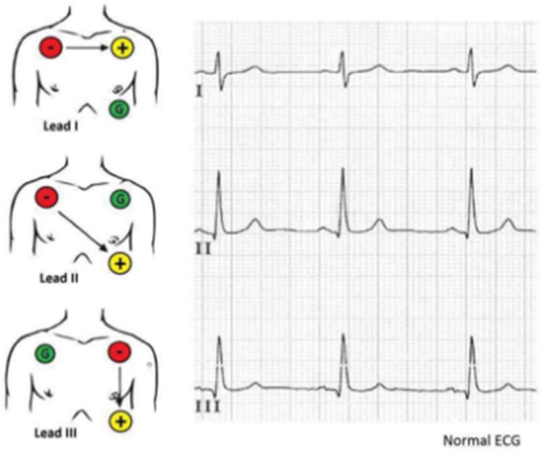


그림 1. Lead 1 2 3 별 검사 방법과 파형 예시[3].

2.2 문제 및 해결방안

심전도 분석 학습시에 발생한 문제점들은 다음과 같다. 불규칙한 기저선은 심방과 심실의 탈 분극 및 재분극에 의해 발생한 전기적 활동이 감지되지 않을 때 나타나는 평평한 직선을 의미한다. 기저선은 뚜렷할 수록 심전도의 전기적 활동이 정확하게 측정된다. 이는 심장의 상태를 간접적으로 확인하는데 중요한 역할을 한다. 하지만 심전도 측정 시 여러 잡음들이 발생할 수 있다. 잡음이 포함되게 되면, 기저선은 불규칙하게 나타나게 된다. 이 잡음들은 1Hz 미만으로 인한 변동도 발생할 수 있으며 전극이 부착된 부위의 피부 건조, 부착이 제대로 되지 않는 경우 등 다양한 상황에서 불규칙하게 나타날 수 있다. 기저선이 뚜렷하게 나타날수록 심장의 상태를 간접적으로 확인하기 좋지만, 이번 실험에 사용한 데이터 셋에는 기저선이 뚜렷하지 않았다. 기저선에 포함된 노이즈를 제거하기 위한 여러가지 방법들[4]도 있지만, 이번 연구 주제 상 노이즈 제거를 노력하기 보단 다양한 모델을 활용한 분석에 초점을 두었기에 노이즈 감소 기법을 별도로 적용하지 않았다.

다른 문제점은 데이터셋[1]에서 부정맥이라고 판단하는 기준의 모호함이 있다. 원본 PTBDB 데이터셋에는 Myocardial infarction, Cardiomyopathy/Heart failure, Bundle branch block, Dysrhythmia, Myocardial hypertrophy, Valvular heart disease, Myocarditis, Miscellaneous, Healthy controls 등 총 9 가지 label 이 있다. 그러나 정상인 healthy controls 와 9 가지 category 에 포함되는지 분석이 어려운 Miscellaneous 를 제외하면 실제로 부정맥인 경우는 Bundle branch block(다발갈래차단)과 Dysrhythmia(율동 부정) 두 가지만 해당된다. 다른 병명들은 질병으로 인해 부정맥이 발생하는 경우가 많아서 판단 기준이 모호하다. 즉, 부정맥 자체를 판별하는 것인지, 다른 질병 때문에 부정맥 현상이 나타나는 것인지 알기 어렵다. 부정맥 자체인지, 다른 질병이 원인으로 부정맥이 나타나는지는 중요한 차이가 있다. 한 예로, 심근 경색(Myocardial infarction)이 있다. 심근 경색이 있으면 심장 근육에 문제가 생기기에 부정맥이 발생할 수밖에 없다. 하지만, 심장 근육이 살아있는 시기에 따라 얼마든지 다른 파형이 나올 수 있다[5].

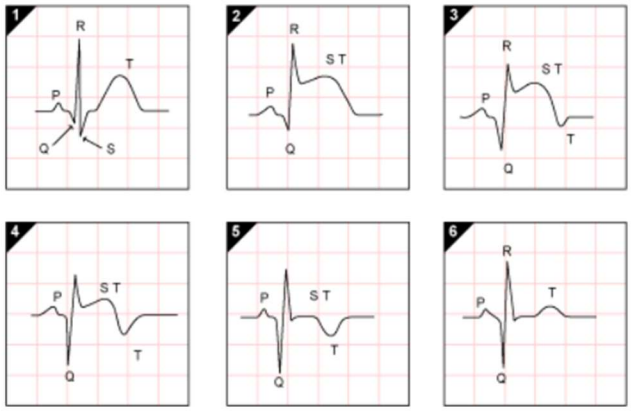


그림 2. 심근 경색의 진행에 따른 심전도 파형 변화 과정[5].

이러한 문제들을 해결하기 위해 상세 label 설명 없이 단순히 abnormal 과 normal 로 구분되어서 데이터셋[1]이 제공하는 PTBDB 데이터셋은 적합하지 않다고 생각했다. 대신 train 데이터셋과 test 데이터셋이 각각 제공되고 각 파형 별 상세 category 가 제공되는 MITBIH 데이터 셋을 이번 연구에 사용했다. MITBIH 에서 상세 label 중 Normal beat 만 정상으로 구분하고, 나머지 Supraventricular premature beat, Premature ventricular contraction, Fusion of ventricular and normal beat, Unclassifiable beat 를 비정상으로 구분해 데이터셋을 수정했다. 이를 통해 이번 실험에 사용할 정상과 비정상 분류 기준을 세울 수 있었다. PTBDB 데이터셋은 나중에 설명할 Ensemble 의 학습 및 검증 데이터로 활용된다.

3. 세부 설계 및 구현

3.1 개발환경

모델 학습 서버

종류	내용
CPU	2*XEON E5-2660v4
OS	ubuntu 20.04 LTS
RAM	64GB
GPU	2080TI 11GB
기타	tensorflow: 2.4.1 cuda=11.6 CUDNN, python3.9 등 다수

웹서버 및 모델 학습 및 테스트

종류	내용
CPU	Ryzen 3600
OS	ubuntu 20.04 LTS
RAM	32GB
GPU	4070TI
기타	tensorflow : 2.12.0 cudnn : 8.9.1 cuda : 18.8 Numpy: 1.24.3 Flask 등 다수

그림 3. 개발 환경.

본 논문의 개발 환경은 모델 학습 서버와, 웹서버 겸 학습 겸 테스트 PC 둘 다 사용하였다. ubuntu 20.04 LTS 에서 개발되었으며, python3.9 와 Flask 를 사용하였다. 그림 3 에 자세히 적었다.

3.2 시스템 구성

이 시스템은 학습된 모델을 활용하여 예측 결과를 제공한다. 정적 웹 페이지의 경우 NGINX 를 통해 즉시 접근할 수 있다. 그러나 예측 결과를 동적 웹 페이지에서 받기 위해서는 딥러닝 모델 서버가 실행되어야 한다. 이에 따른 속도 저하, 보안 문제 및 병렬 처리의 고려로, WSGI 서버로 gunicorn 을 도입하여 웹 페이지의 동적인 요청을 처리한다. 또한, Flask 프레임워크를 활용하여 학습된 모델을 호출하고 예측 결과를 다시 웹페이지로 반환한다. 이를 통해 웹페이지 내에서 예측 결과를 효율적으로 받을 수 있으며, 더 빠른 속도와 보안 강화, 병렬 처리를 통한 원활한 서비스 제공이 가능하다. 그림 4 를 통하여 한눈에 알아볼 수 있다.

그림 4. 시스템 구성도.

3.3 심전도 분석 모델

심전도 데이터 분석 모델은 총 4 가지를 준비했다. 각각의 모델마다 구조가 달라 성능에 차이가 발생할 수밖에 없다. 그렇기에 이용자가 각 모델 별 성능지표를 통해 여러 모델을 선택할 수 있도록 준비했다.

학습에 사용한 데이터셋[1]은 P, Q, R, S, T 파가 반복되는 심박(heartbeat) 마다 잘라서 제공된다. 이렇게 자르는 방식의 원리는 논문[6]을 참고하면 된다.

이번 실험에서 공통적으로 사용한 데이터 전처리(data preprocessing) 과정은 크게 두가지이다. 데이터 레이블을 기준을 세워 normal, abnormal 로 변경한 것과, 데이터 셋 안의 클래스 간 데이터 개수가 불균형(class imbalance) 한 문제를 해결하기 위해 가장 작은 클래스의 개수에 맞춰 나머지 데이터들을 삭제했다. 데이터 전처리의 과정 중 class 에 관한 기준은 '2.2 문제 및 해결방안'에 설명되어 있다.

Train 과정에 사용한 데이터 비율은 다음과 같다. 데이터 전처리 과정을 거친 데이터에서, 0.1 비율을 Train 과정을 마친 후 Test 데이터로 사용했다. 나머지 0.9 의 데이터 중, 0.7 을 training 에 사용했고, 0.3 을 validation 데이터로 사용했다. 즉, Train 에 사용된 데이터셋은 전처리를 거친 데이터 셋 중 0.63 만 사용되었고, validation 에 사용된 데이터는 전체의 0.27 이다. Train 을 마친 후 바로 Test 를 한 데이터는 전체의 0.1 이고, 이때의 데이터 개수는 3017 개이다. 각 모델별 Train result 는 3017 개의 데이터로 평가한 성능지표를 의미한다.

Test 과정에 사용한 데이터는 제공된 데이터셋 중 Test 데이터 셋 전체를 사용했다. 전체 데이터 개수는 21892 개이다. 각 모델 별 Test result 는 21892 개의 데이터로 평가한 성능지표를 의미한다. 데이터의 양이 많지 않기에, normal 과 abnormal 의 데이터 개수를 맞추지 않고 전체 데이터를 사용하고, 객관성을 나타내기 위해 matrix 로 True Positive, True Negative, False Positive, False Negative 의 값을 전부 보여주었다.

3.3.1 CNN-LSTM

CNN(Convolution Neural Network)과 LSTM(Long Shot-Term Memory)을 결합한, 이번 실험에서 기본(base)모델로 사용했다. CNN 부분은, 이 모델에서 1D-CNN 을 의미한다. 시계열 데이터의 지역적인 패턴을 추출하는데 특화되었다. 지역적인 패턴이란, 데이터의 특정 구간에서 나타나는 패턴을 의미한다. Convolution 연산을 진행하면서 필터를 이용하여 입력 데이터의 특징을 검출하고, MaxPooling 연산을 통해 데이터의 길이를 줄여 벡터를 축소시킨다. 데이터의 길이를 줄여주는 것은 학습 파라미터의 수를 줄여주고 overfitting 을 방지하는데 도움이 된다. 이 모델은 Convolution 연산과 MaxPooling 연산 반복을 통해 점점 학습 데이터의 공간(spatial size)를 축소해주는 방향으로 설계했다. CNN 의 출력을 LSTM 의 입력으로 사용된다. LSTM 은 시계열 데이터의 전체적 시간적 흐름을 고려해 장기적인 의존성(long-term dependency)을 모델링하는데 특화되었다. 장기적인 의존성이란, 이전 데이터가 미래의 데이터에 미치는 영향을 의미한다. 즉, 시계열 데이터의 앞 뒤 관계를 예측하는데 도움이 된다. CNN 모델과 LSTM 을 결합하면, 시계열 데이터의 지역적인 패턴과 장기적인 의존성을 모두 고려해 더 정확한 예측을 제공할 수 있다. 중간에 있는 Maxpooling layer 는 입력 데이터의 차원을 줄여 모델의 복잡도를 낮추고 중요한 정보만을 유지하는데 도움을 준다. 이로 인해 모델의 과적합(overfitting) 방지, 일반화(generalization) 성능 향상, 모델의 복잡도 감소로 인한 계산 비용 감소를 얻을 수 있다. 단점은 정보 손실로 인한 모델 성능 저하이므로, 적절하게 사용해야 한다. 그림 5 와 그림 6 은 CNN 연산을 지나 LSTM 을 적용하는 모델의 구조이다.

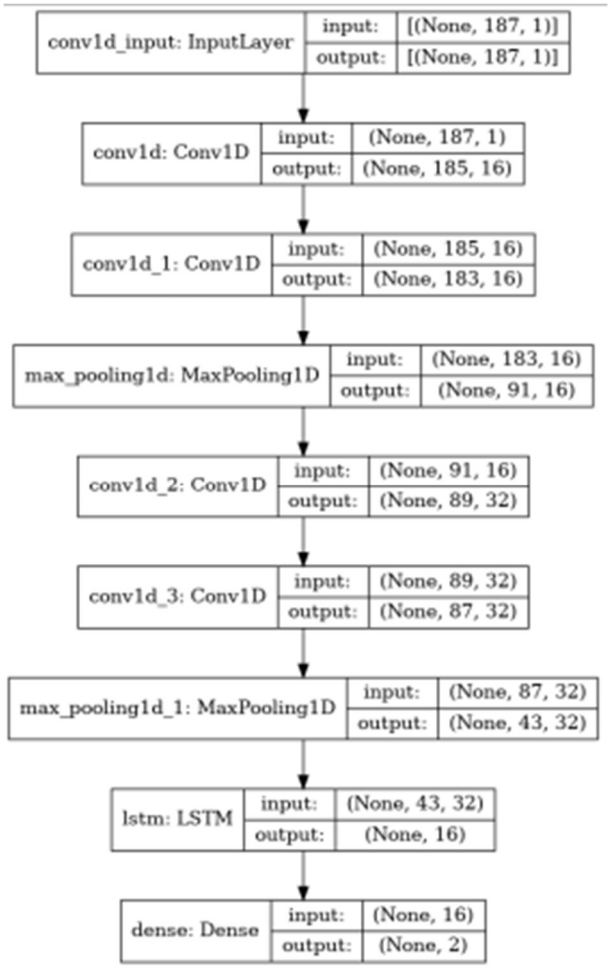


그림 5. 이번 실험에서 기본(base) 모델로 사용한 CNN-LSTM 모델 구조.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 185, 16)           64
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 183, 16)           784
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 91, 16)            0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 89, 32)            1568
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 87, 32)            3104
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 43, 32)            0
_________________________________________________________________
lstm (LSTM)                  (None, 16)                3136
_________________________________________________________________
dense (Dense)                (None, 2)                 34
=================================================================
Total params: 8,690
Trainable params: 8,690
Non-trainable params: 0
_________________________________________________________________
```

그림 6. model.summary() 로 출력한 정보.

이 모델의 학습 과정을 시각화하기 위해, loss vs. epoch 그래프와 accuracy vs. epoch 그래프인 그림 7 을 추가했다. Loss vs. epoch 그래프는 Epoch 과 Loss 간의 관계를 나타내고, accuracy vs. epoch 그래프는 epoch 과 accuracy 간의 관계를 나타낸다. 이 두 그래프를 동시에 확인하면, 두 그래프를 동시에 확인함으로써 얻을 수 있는 정보들은 다음과 같다.

1. 모델의 학습 상태: loss vs. epoch 그래프를 통해 모델이 얼마나 잘 학습되고 있는지, loss가 감소하는지, overfitting이 발생하는지 등을 확인할 수 있다. Loss가 초반엔 빠르게 감소하고 후반엔 느리게 감소하는 것이 좋다.
2. 모델의 예측 성능: accuracy vs. epoch 그래프를 통해 모델의 정확도가 얼마나 높은지, 정확도가 증가하는지, overfitting이 발생하는지 등을 확인할 수 있다. Accuracy가 초반엔 빠르게 증가하고 뒤로 후반엔 느리게 증가하는 것이 좋다.
3. 과적합(overfitting) 여부: loss가 감소하지만 accuracy가 증가하지 않는 경우에는 overfitting이 발생할 가능성이 있다. 이 경우에는 모델이 training data에만 과도하게 적응하여 test data에서의 성능이 저하될 수 있다. 이 경우에는 early stopping이나 Regularization, pruning, ensembling을 활용해서 과적합을 막거나, 데이터의 양을 늘려 더 많은 학습이 이뤄지도록 데이터를 더 많이 수집하거나 data augmentation기법을 이용해야 한다.
4. 학습 속도: loss가 빠르게 감소하고 accuracy가 빠르게 증가하는 경우에는 모델이 빠르게 학습되고 있다는 것을 알 수 있듯, 두 그래프를 함께 분석하면 모델의 학습 속도를 파악할 수 있다.

성능 지표를 나타내는 Train Result 와 Test Result 를 표 1 에서 확인할 수 있다. Train Result 와 Train Result 와 관련된 데이터 정보들은 '3.3 심전도 분석 모델'에 나와있다.

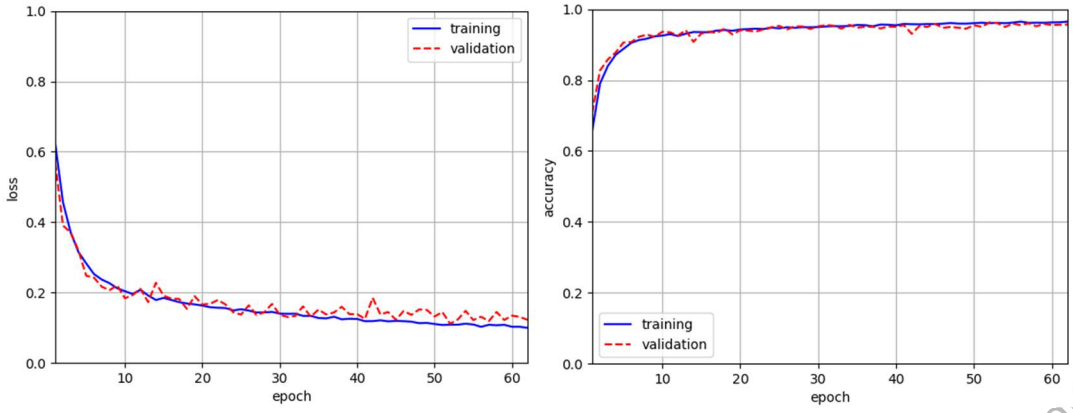


그림 7. 기본(base) 모델로 사용한 CNN-LSTM 모델의 loss vs. epoch 그래프(좌), accuracy vs. epoch 그래프(우).

	Train Result				Test Result			
Accuracy	0.963871				0.973095			
Precision	0.975543				0.894281			
Recall	0.951624				0.957075			
F1 score	0.963435				0.924613			
Cohens kappa	0.927743				0.908261			
ROC AUC	0.963875				0.966753			
matrix			Predictive Values				Predictive Values	
			Positive	Negative			Positive	Negative
	Actual Values	Positive	1472	36	Actual Values	Positive	17691	427
		Negative	73	1436		Negative	162	3612

표 1. 기본(base) 모델로 사용한 CNN-LSTM 모델의 Train Result와 Test Result.

3.3.2 CNN-LSTM, Dropout

CNN 과 LSTM 을 사용한 기본 모델에 dropout 을 적용했다. Dropout 은 훈련 데이터를 과도하게 학습함으로 노이즈와 일반화가 잘 되지 않는 세부사항까지 학습하여 발생할 수 있는 과적합(overfitting)을 방지하기 위해 사용한다. Dropout 은 각 훈련 반복(epoch)마다 미리 설정된 비율만큼 무작위로 뉴런을 선택해 비활성화하는 방법이다. Dropout 을 사용하기 위해 output 의 출력차원을 늘려야 했고, 그로 인해 convolution layer 의 filter 의 개수를 기본 모델보다 2 배씩 늘렸다. 이 모델은 CNN 연산과 CNN 연산 사이 dropout 기법을 적용시키고, CNN 을 지나 LSTM 을 적용하는 모델이다. 모델의 구조는 그림 8 과 그림 9 를 통해 확인 가능하다.

이 모델의 학습 과정과 성능을 측정할 수 있는 지표를 그림 10 과 표 2 에 첨부한다.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 185, 32)           128
_________________________________________________________________
dropout (Dropout)            (None, 185, 32)           0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 183, 32)           3104
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 91, 32)            0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 89, 64)            6208
_________________________________________________________________
dropout_1 (Dropout)          (None, 89, 64)            0
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 87, 64)            12352
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 43, 64)            0
_________________________________________________________________
lstm (LSTM)                  (None, 16)                5184
_________________________________________________________________
dense (Dense)                (None, 2)                 34
=================================================================
Total params: 27,010
Trainable params: 27,010
Non-trainable params: 0
```

그림 8. model.summary() 로 출력한 정보.

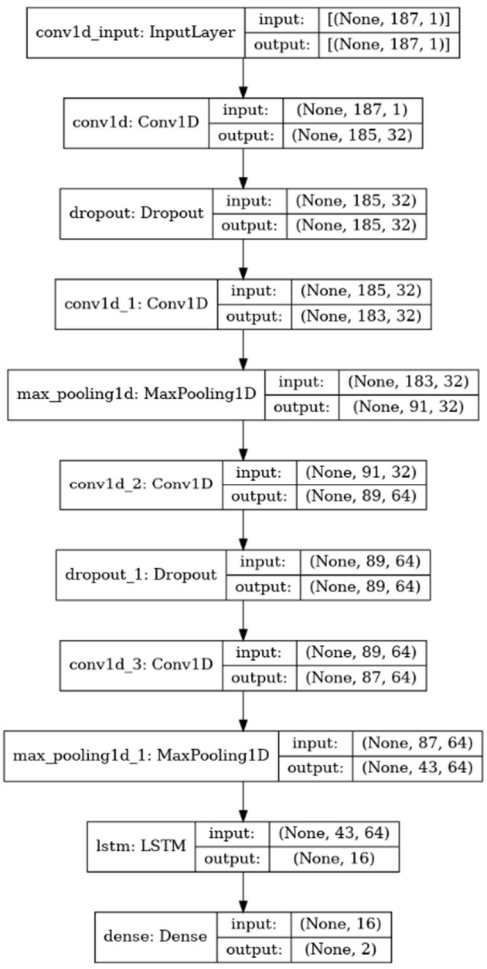


그림 9. 이번 실험에서 기본(base) 모델에 dropout 기법을 적용한 CNN-LSTM 모델 구조.

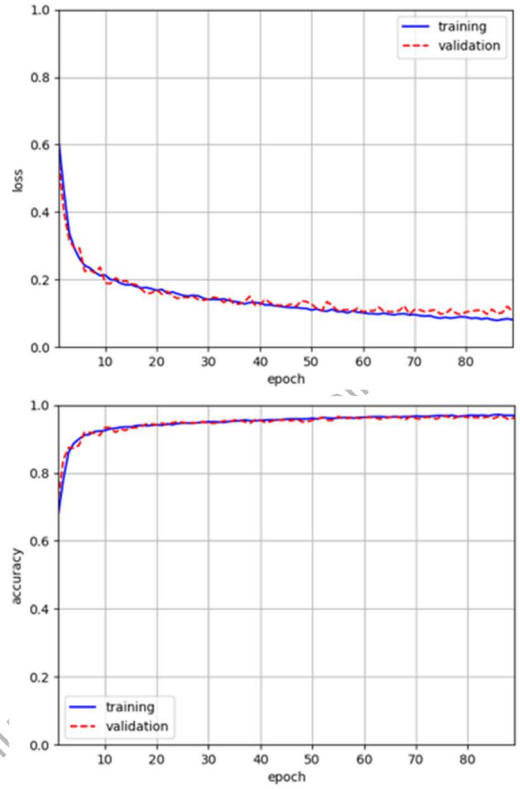


그림 10. 기본(base) 모델에 dropout 기법을 적용한 CNN-LSTM 모델의 loss vs. epoch 그래프(상), accuracy vs. epoch 그래프(하)

	Train Result				Test Result			
Accuracy	0.969838				0.963274			
Precision	0.973930				0.849576			
Recall	0.965540				0.956280			
F1 score	0.965540				0.899776			
Cohens kappa	0.939675				0.877390			
ROC AUC	0.969839				0.960506			
Matrix			Predictive Values				Predictive Values	
			Positive	Negative			Positive	Negative
	Actual Values	Positive	1469	39	Actual Values	Positive	17479	639
		Negative	52	1457		Negative	165	3609

표 2. 기본(base) 모델로 사용한 CNN-LSTM 모델의 Train Result와 Test Result.

3.3.3 CNN-LSTM, Dropout, Ensemble

이번 실험에 사용한 데이터셋[1]에는 MITBIH 와 PTBDB 데이터셋 두가지가 있었다. 부정맥 판단 기준이 명확한 MITBIH 데이터셋을 이용해 normal 인지 abnormal 인지 분류하는 모델을 만들었지만, test data 에 PTBDB 데이터셋의 abnormal 데이터셋을 투입 시 올바른 예측이 전혀 되지 않는 문제가 있었다. MITBIH 데이터셋에 포함된 ECG category 와 PTBDB 데이터셋에 포함된 ECG category 는 서로 다른 점도 있는 것이 원인으로 추정한다. 이 문제를 해결하기 위해 MITBIH 데이터셋과 PTBDB 데이터셋을 합쳐서 학습시키기 보단, 각각 학습시킨 모델을 하나로 합치는 ensemble 기법을 사용해 극복하고 싶었다.

데이터셋이 2 개뿐이었기에, 투표 방식(voting)을 사용하기 어려움이 있었다. 그래서 각 모델 별 예측 score 을 합친 뒤, 전체 모델 개수로 나누는 방식으로 ensemble 기법을 적용해보았다. 식으로 나타내면 $최종\ 예측\ score = \frac{(각\ 모델별\ 예측\ score)}{(전체\ 모델\ 개수)}$ 이다. 여기서 전체 모델 개수는 2 개이다.

그림 11 처럼 각기 다른 두 데이터셋을 학습한 모델 2 개를 결합해서 만들어진 모델이라, Train Result 는 생략하고 Test Result 만 표 3 에 보여주었다.

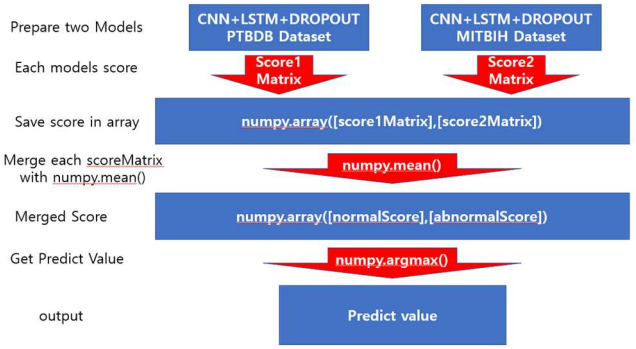


그림 11. CNN-LSTM base모델에 dropout을 적용한 모델을 각기 다른 데이터셋으로 학습한 모델 2개를 사용한 Ensemble 구조

이 모델은 실험적인 모델로 만든 것이므로 성능이 별로 좋진 않다. ensemble 모델을 만들기 전보다 각기 다른 데이터셋 예측을 최소 accuracy 59% 이상 예측하기 시작했다는 의의가 있다.

	MITBIH Test Result				PTBDB normal Result				PTBDB abnormal Result			
Accuracy	0.590124				0.989867				0.710070			
Precision	0.293182				0.000000				1.000000			
Recall	0.976418				0.000000				0.710070			
F1 score	0.450958				0.000000				0.830458			
Cohens kappa	0.252836				0.000000				0.000000			
ROC AUC	0.743038				Not defined				Not defined			
Matrix			Predictive Values				Predictive Values				Predictive Values	
			Pos.	Neg.			Pos.	Neg.			Pos.	Neg.
	Actual Values	Pos.	9234	8884	Actual Values	Pos.	4005	41	Actual Values	Pos.	0	0
		Neg.	89	3685		Neg.	0	0		Neg.	3046	7460

표 3. ensemble 모델의 각 데이터 셋 별 Test Result.

3.3.4 Transformer

Transformer 모델은 self-attention 기법을 이용해 순차 데이터(sequential data) 내의 관계를 추적해 맥락과 의미를 학습하는 신경망이다. encoder 에서 특징을 뽑고, decoder 에서 주로 자연어처리(NLP, Natural Language Processing) 에 사용되지만 신호, 심지어 영상에도 적용하는 등 다양한 분야에서 사용하려고 연구중인 모델이다. 생성을 하려면 decoder 가 필요하지만, 분류(classification) 모델을 만들 때는 encoder 만 있어도 된다. 그렇기에 부정맥인지 아닌지 분류하는 이번 실험의 모델로 transformer 를 만드는 것은 적합하다. Transformer 모델은 Keras 에서 제공하는, Attention is All You Need 기반으로 만든 Timeseries classification 용 transformer 모델[7]을 사용했다.

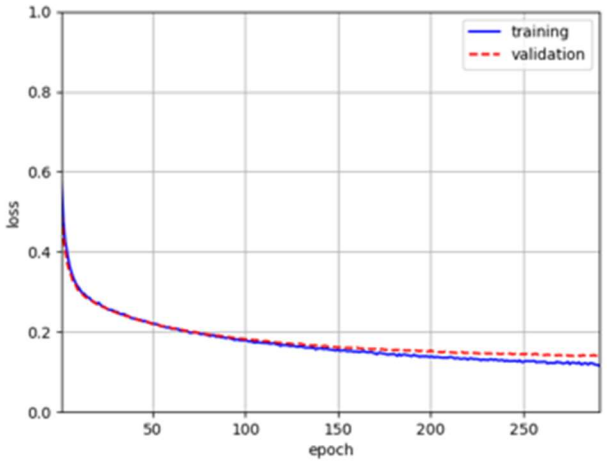
Keras 의 예제[7] 모델을 잘 보면, Conv1d layer 가 있다. 이는 transformer 의 encoder 블록 구성 요소 때문인데, encoder 블록은 크게 Multihead attention part 와 Feed Forward part 로 나뉜다. 이 예제 모델은 Feed Forward part 의 projection layer 는 conv1D layer 를 통해 구현된다.

Transformer 모델은 CNN 에 비해 inductive bias 가 낮다. Inductive bias 란, 모델에 사전적으로 주어지는 가정으로 이전에 만나지 않은 입력에 대한 출력을 예측하는데 사용된다. Inductive bias 가 낮다는 말은, inductive bias 까지 학습하기

위해 많은 양의 데이터가 필요함을 의미한다. 그 예로, Vision Transformer 논문[7]에서 imagenet-1k 데이터셋 정도의 크기는 CNN 기반 모델과 비슷한 성능을 보였고 imagenet-21k 처럼 학습데이터가 많을수록 CNN 보다 나은 성능을 보였다.

이번 실험의 데이터셋의 크기는 매우 작은 편에 속한다. 그렇기에 transformer 모델을 학습시키기엔 충분하지 않은 데이터 셋 크기일 수도 있다. 실제로 그림 12 의 Transformer 모델을 학습시켰을 때, CNN 보다 다소 부족한 성능을 보였다. 학습 과정은 그림 13 에, 성능 지표는 표 4 에 정리하였다.

그림 12. 이번 실험에서 사용한 Transformer 모델의 구조. 이미지가 너무 길어 좌측부터 우측으로 분할 배치함. model.summary() 정보도 위와 같이 너무 길어 생략함.



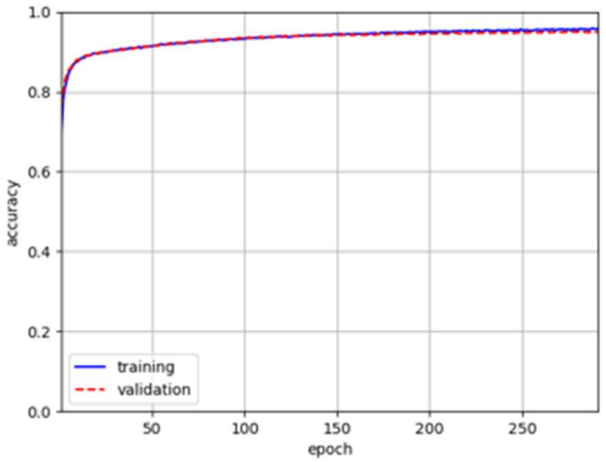


그림 13. Transformer 모델의 loss vs. epoch 그래프(좌), accuracy vs. epoch 그래프(우).

	Train Result				Test Result			
Accuracy	0.954922				0.963457			
Precision	0.977067				0.869166			
Recall	0.931743				0.927663			
F1 score	0.953867				0.897462			
Cohens kappa	0.909846				0.875258			
ROC AUC	0.954930				0.949288			
Matrix			Predictive Values				Predictive Values	
			Positive	Negative			Positive	Negative
	Actual Values	Positive	1475	33	Actual Values	Positive	17591	527
		Negative	103	1406		Negative	273	3501

표 4. Transformer 모델의 Train Result 와 Test Result.

3.4 Flask 의 활용

3.4.1 REST API 구축 및 Jinja2 템플릿

학습된 모델을 서빙하기 위해서는 모델에 대한 REST API 를 구축해야 한다. FLASK 를 통해 API 를 구축하여 사용자에게 머신러닝 모델 기반 웹 서비스를 제공한다. 사용자는 데이터 파일을 전송하여 API 를 호출해 데이터 파일을 서버에 전송한다. 그리고 서버에서 데이터 파일의 행을 개수를 확인하며, 모델에 적합하지 않은 데이터 파일은 데이터 전처리 과정을 수행한다. 모델은 데이터에 대한 예측을 수행하고 결과를 반환한다. 그리고 Jinja2 템플릿 엔진을 통해 반환된 결과값을 출력하는 동적 웹 페이지가 생성된다.

3.4.2 병렬성

Flask 는 단일 쓰레드에서 실행되며, 순차적으로 들어온 요청을 처리한다. 단일 쓰레드에서 실행되기 때문에 동시에 많은 요청을 처리하는데 한계가 있다는 단점이 존재한다. 이러한 단점을 보완하기 위해 WSGI 서버인 Gunicorn 을 사용했다. Gunicorn 은 master process 와 worker process 로 구성되어 있는데, gunicorn 이 실행될 때 프로세스가 master process 이다. master process 는 fork 를 통해 정의된 개수만큼 worker process 를 생성하고, worker processs 는 HTTP 요청을 독립적으로 처리한다. worker process 를 통해 병렬성을 확보할 수 있어 동시에 많은 사용자들의 요청을 처리할 수 있어 응답 속도를 향상시킬 수 있다.

3.5 구현결과

그림 14 는 구현된 웹 페이지의 홈 화면이다. 홈 화면 또는 상단 바에서 성능지표, 파형, 예측의 3 가지 기능을 작동하도록 구현했으며 그림 15 의 성능 지표 기능은 3 가지 모델의 test result 와 train result 의 Accuracy, Precision, Recall, F1 Score, Cohens kappa, ROC AUC 6 가지 성능지표를 확인할 수 있다. 그림 16 의기능은 분석하려는 데이터를 업로드 할 때, 데이터 파일의 행을 선택해 확인하고 싶은 특정한 데이터의 행의 파형을 확인할 수 있다. 그림 17, 그림 18 의 예측 기능 모델을 선택해 전체, normal, abnormal 의 예측 결과를 반환하며 전체 데이터의 결과값을 확인할 수 있으며, 도출된 결과 값 중 확인하고자 하는 특정 행의 파형을 새로운 창에 출

력한다. 그림 19 는 모델의 성능을 확인하고자 할 때 사용하는 Performance evaluation 모드이다.

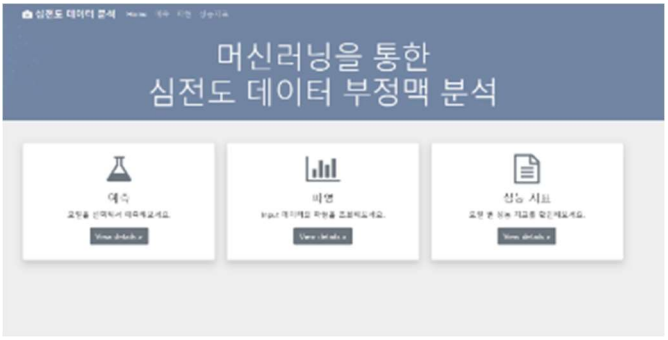


그림 14. 웹 페이지의 홈 UI 구현

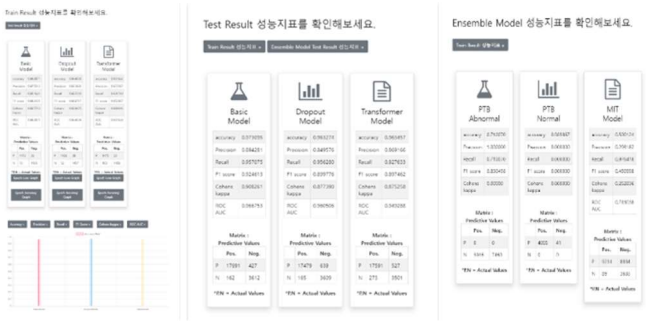


그림 15. 각 모델의 Train result의 성능 지표와 그래프, Test result의 성능 지표, Ensemble 모델의 Test result 성능 지표

그림 16. 파형을 확인할 데이터를 업로드 하는 화면(좌), 선택한 파형 출력 결과(우)

그림 17. 업로드 할 데이터 파일과 출력될 행 선택할 체크박스 화면

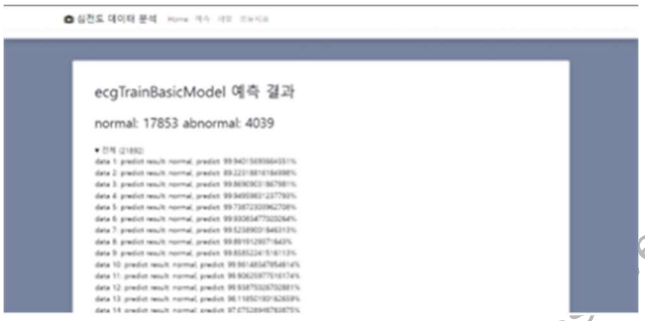


그림 18. 입력한 전체 데이터와 normal, abnormal 데이터

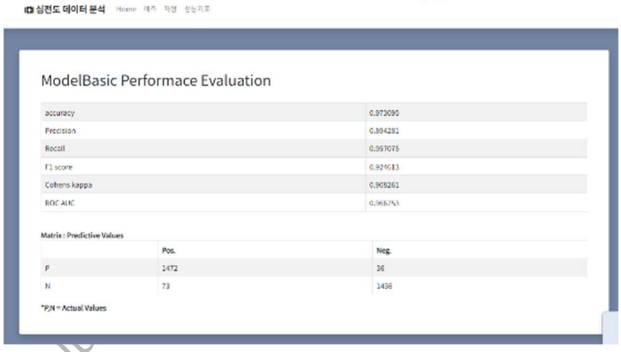


그림 19. Performance evaluation 모드를 체크하고 예측했을 때 화면, 레이블 데이터와 예측 데이터 비교 가능

5. 결론 및 향후 연구과제

제안된 4 가지 모델 중, 실험 성격이 강한 ensemble 모델을 제외한 나머지 모델들을 통해 심전도 분석이 약 95% 이상의 accuracy 로 가능하다는 것을 보여준다. 완전히 동일한 데이터셋으로 비교하지 못해 성능 비교가 어렵지만, CNN 을 활용해 PhysioNet/CinC challenge 2015 결과를 비교한 연구[2]의 성능인 Accuracy 88.67%, Sensitivity 89.47%, Specificity 88.03% 보다 더 좋은 성능을 보여준다. 하지만 우리가 만든 모델을 실전에 사용하기 위해 다음 사항을 해결해야 한다. 실험에 사용한 데이터셋 보다 노이즈가 적고, baseline 이 뚜렷한 양질의 데이터셋이 요구되며, 구체적이고 체계적인 부정맥 판별 기준이 필요하다. 학습 데이터도 매우 많아야 한다. MITBIH 데이터셋과 PTBDB 데이터셋에 포함된 데이터 양이 작은 편이라, 실험에 사용된 데이터셋은 원래의 연속된 심전도 파형을 P Q R S T 파로 구분 가능한 심박(heartbeat) 1 회별로 잘라서 데이터의 개수를 늘리는 방식으로 전처리를 하였다. 이것의 장점은 heartbeat 별로 잘라서 학습된 모델에 넣으면 어느 파형에서 이상이 발생했는지 쉽게 찾을 수 있다. 실제 심전도 데이터는 한 데이터 안에 여러 heartbeat 가 포함되어 있기에, 더 많은 사람의 심전도 데이터를 수집한 후 train 하면 더 정확한 모델을 만들 수 있을 것이다.

Test 과정에서, 0.9 의 값 들로만 이루어진 일직선 파형 데이터 등 심전도 파형에서 나올 수 없는 데이터들을 생성해서 넣으면 정상으로 분류하는 문제가 있었다. 향후 연구과제로

는, 완전히 이상한 데이터 값이 input 되었을 시 오작동하는 문제를 피하기 위한 OOD 검출 문제(Out Of Distribution Detection problem)를 개발하는 것과, 부족한 데이터 양을 해결하기 위해 데이터를 증강시켜 학습에 도움을 주는 data augmentation 기법 적용 방법을 알아보아야 한다. Data augmentation 기법을 적용할 수 있으면 정상인 심전도 데이터보다 비정상인 심전도 데이터가 매우 작을 수밖에 없어 발생하는 data imbalance 문제 해결도 가능하고, transformer 모델의 성능 향상에 큰 도움이 될 것이다.

만든 모델을 웹 기반으로 연동하여, 일반인들까지 접근성을 향상시킨 것은 장점이다. 모델의 input 에 맞게 변형시킨 심전도 데이터를 가지고 있으면 인터넷에 연결된 모든 환경에서 데이터를 올리고 검사 결과를 확인할 수 있을 것이다

참고 문헌

[1] Fazeli, S. (2018, May 31). ECG Heartbeat Categorization Dataset. Kaggle. https://www.kaggle.com/datasets/shayanfazeli/heartbeat

[2] 박희환. (2016). 딥 러닝 알고리즘을 이용한 중환자실 내 위급한 부정맥 증상 탐지.

[3] EKG worksheet - use powerpoints to fill out - needs to be done for Day 1 of class. HCCS Learning Web. (n.d.). https://learning.hccs.edu/programs/emergency-medical-services/rn-to-paramedic-program/ekg-items-use-powerpoints-to-fill-out-the-worksheet-before-class/ekg-powerpoints-part-1 page 12

[4] 배기수. (2005). 심전도 신호의 기저선 변동 잡음 (Baseline wander) 제거를 위한 형태연산 (Morphology operation) 필터 설계 (Doctoral dissertation, 연세대학교 대학원).

[5] School of Health Sciences. The University of Nottingham's website. (n.d.). https://www.nottingham.ac.uk/nursing/practice/resources/cardiology/acs/changes.php

[6] Kachuee, M., Fazeli, S., & Sarrafzadeh, M. (2018, June). Ecg heartbeat classification: A deep transferable representation. In 2018 IEEE international conference on healthcare informatics (ICHI) (pp. 443-444). IEEE.

[7] Team, K. (n.d.). Keras Documentation: Timeseries classification with a Transformer model. https://keras.io/examples/timeseries/timeseries_transformer_classification/

[8] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.