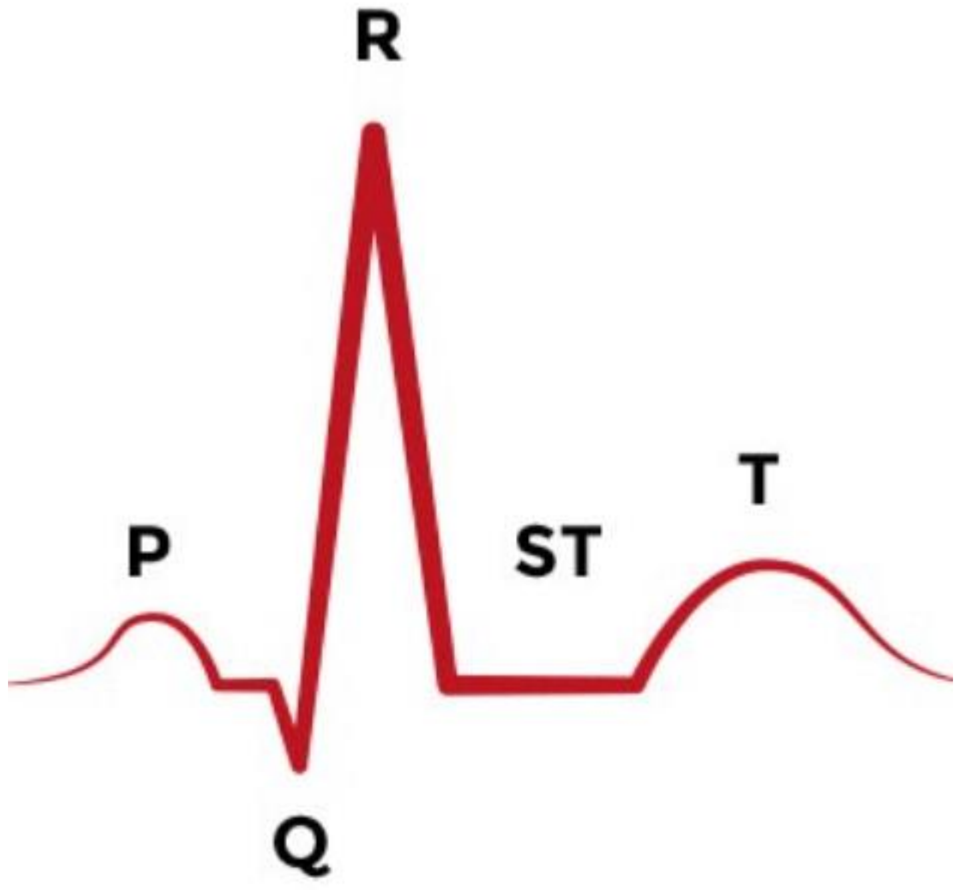
머신러닝 개인별 과제

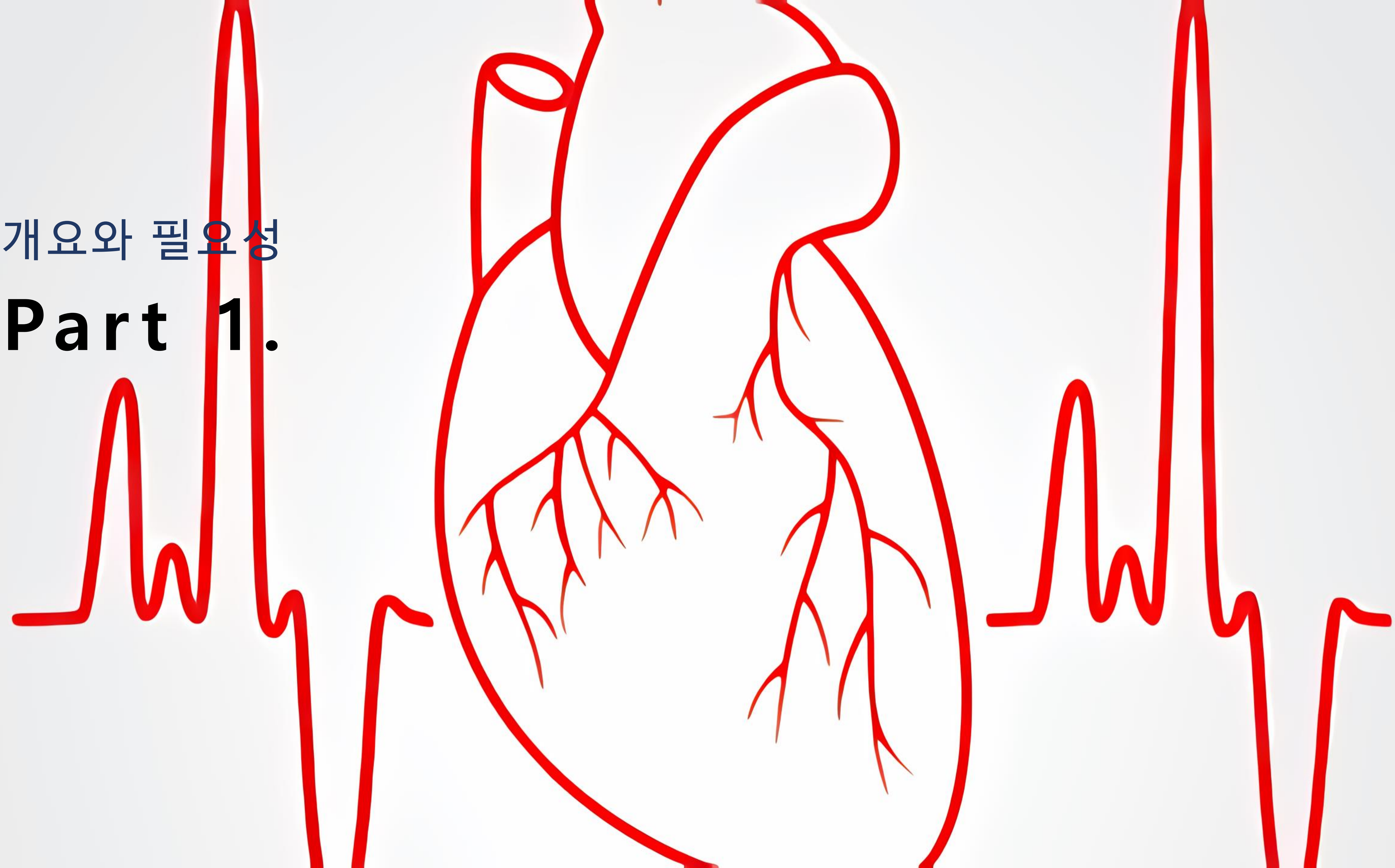
컴퓨터공학과 3학년
2018108264 학번
선종호

목적을 이루지 못한

CNN을 이용한
ECG 판독과 부정맥

목차





개요와 필요성

Part 1.

1.1 개요

주제 개요

부정맥 환자와 비 환자의 ECG 데이터 SET 학습을 통한 부정맥 예측을 하는 것이 제가 선정한 Kaggle Note의 목적입니다.

Kaggle 주소

https://www.kaggle.com/code/gregoiredc/arrhythmia-on-ecg-classification-using-cnn

1.1 필요성

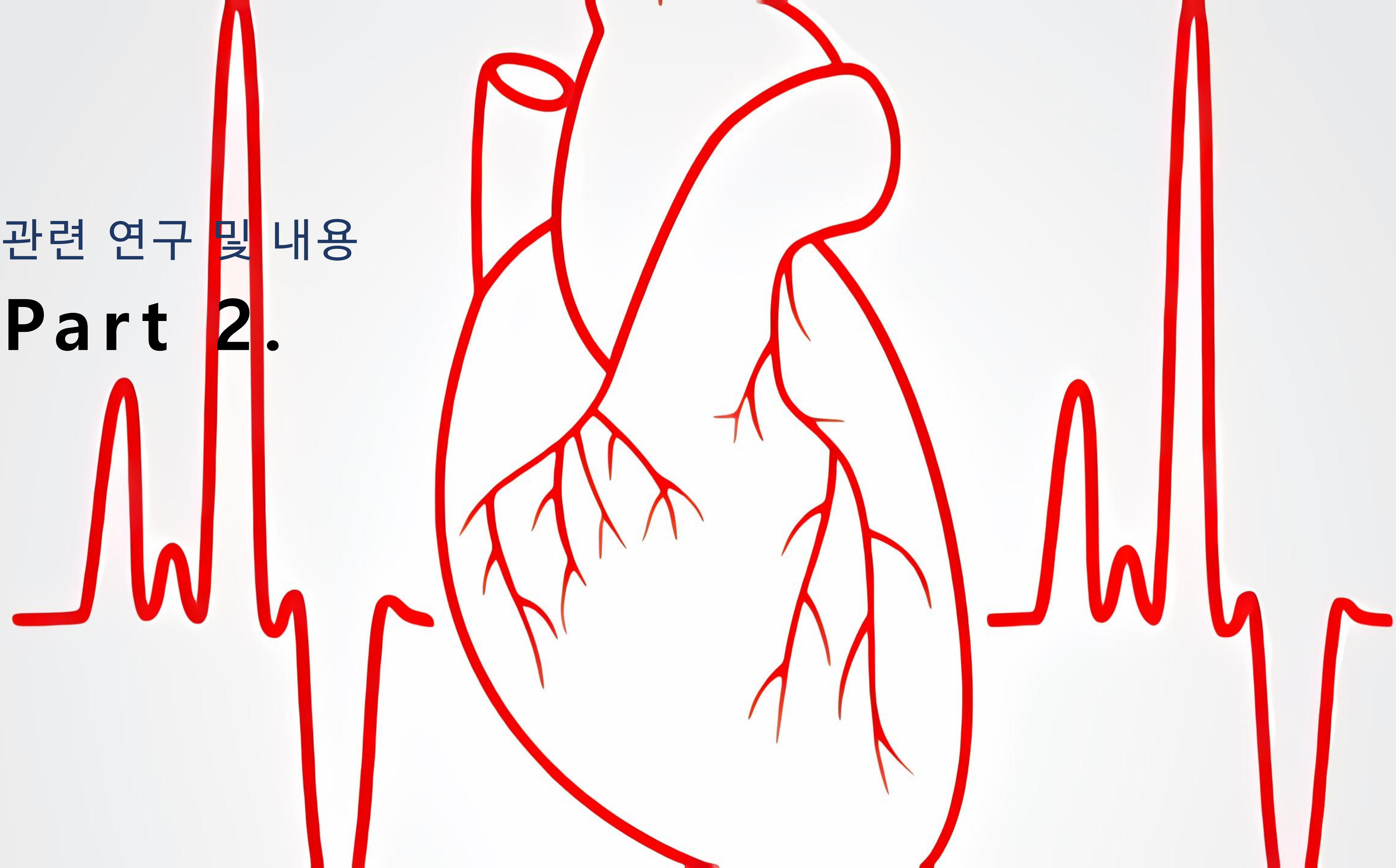
주제 선정 이유와 필요성

최근에 본인의 심전도 데이터를 얻을 일이 생겨서, 머신러닝을 활용한 데이터 분석이 가능할 것으로 생각되어 해당 주제를 선정하게 되었습니다.

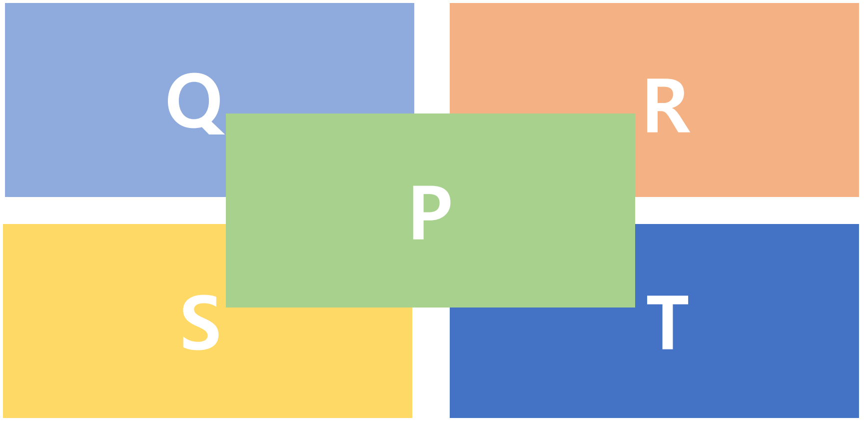
부정맥 증세를 호소하는 사람은 점차 늘어가는 중이고, 본 주제에서 활용한 ECG 측정 장비는 근래에 스마트 웨어러블 디바이스의 보편화로 인하여 접근성이 좋아졌습니다. 의료적 판단을 내리기에 유효한 근거 자료는 되지 못하겠으나, 개인의 병원 내방에 대한 이유가 될 수 있겠습니다.

관련 연구 및 내용

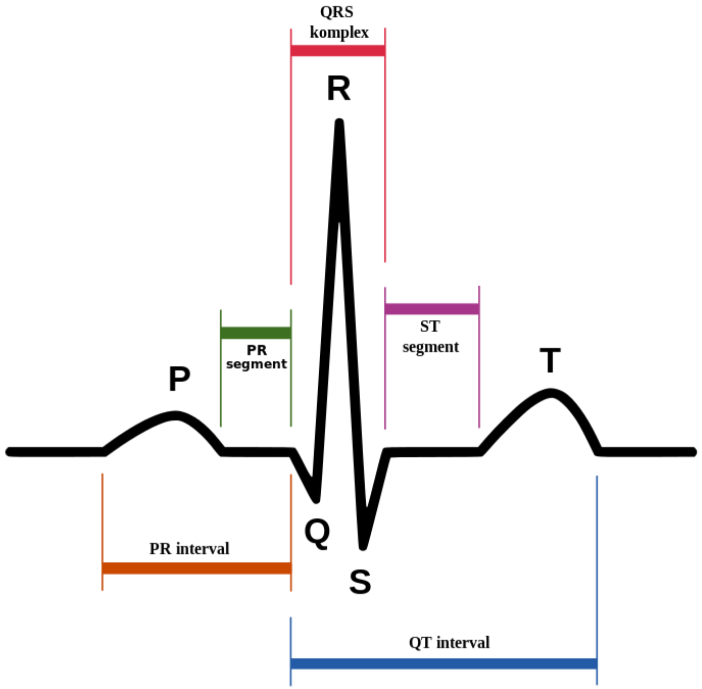
Part 2.



2.1 사전 정보

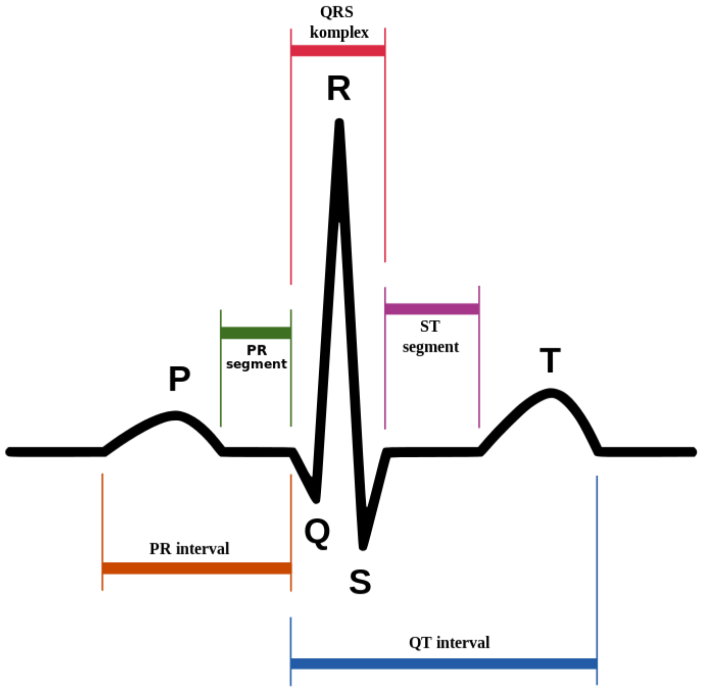


ECG (심전도)의 구성 요소



심박이 일정한 경우,
RR Interval =가장 높은 파형인 R과 R 사이의 간격이 1분에 몇 번 나타나느냐가 바로 심박수가 된다.

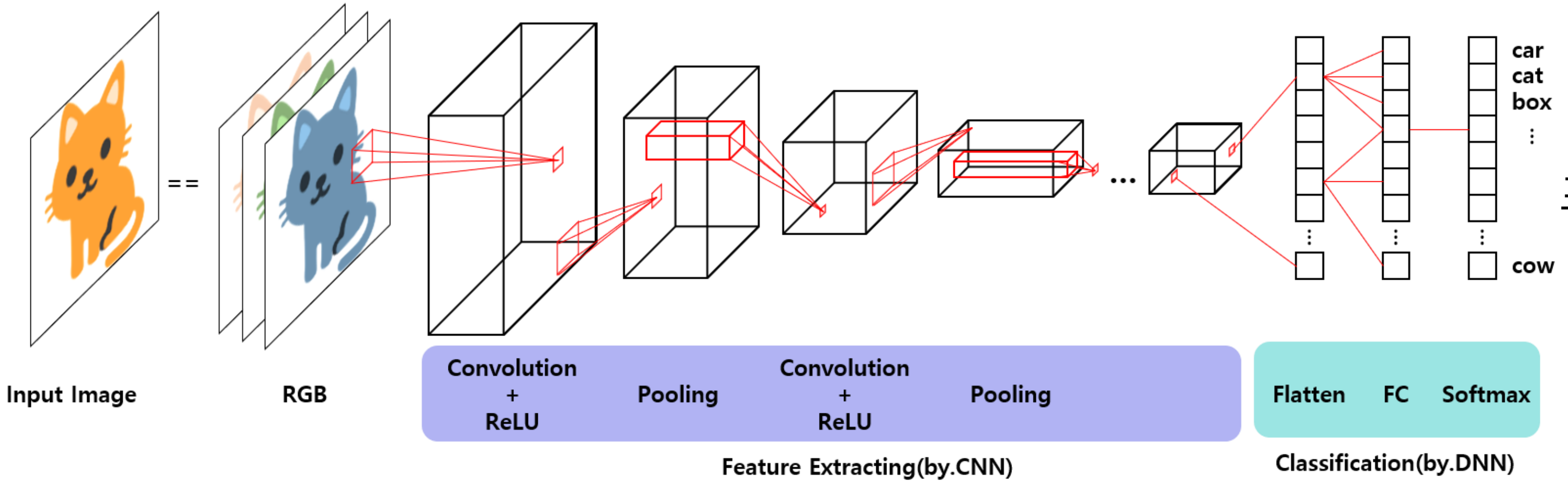
2.1 사전 정보



부정맥

不整脈. 맥박이 가지런하지 않은 질병이다.
앞서 말한 심박 측정 방법을 사용할 수 없다.
대신 3초나 6초 동안의 박동 수(파형 반복 횟수)를 기록하여 해당 수에 20 혹은 10을 곱하여 분당 심박수를 임의로 계산한다.

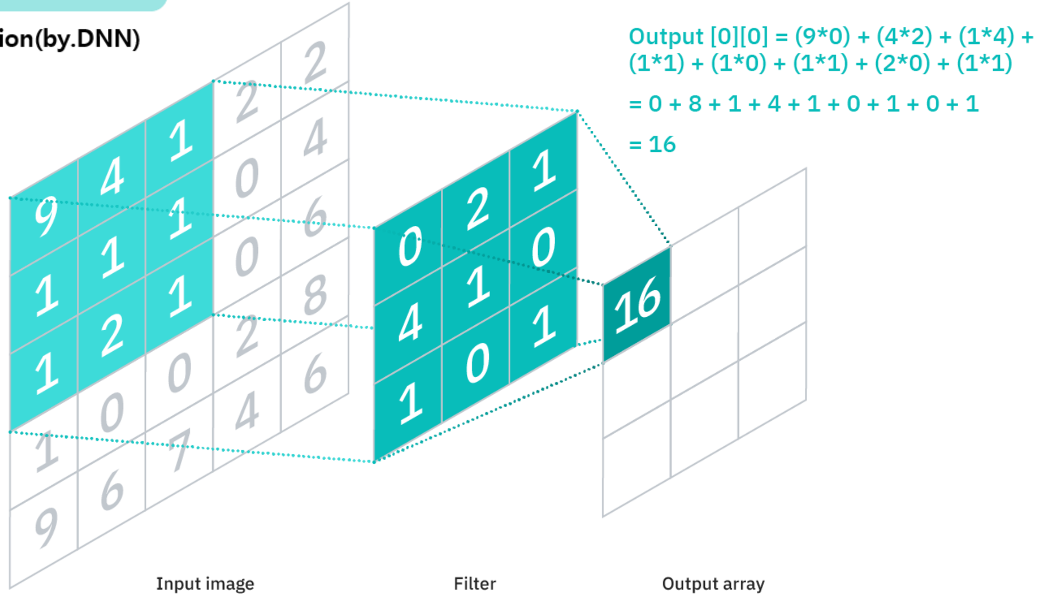
2.2 CNN



Convolution이란?

이미지 전체가 아닌 부분의 특성을
포착하는 필터를 적용해나가는 과정.

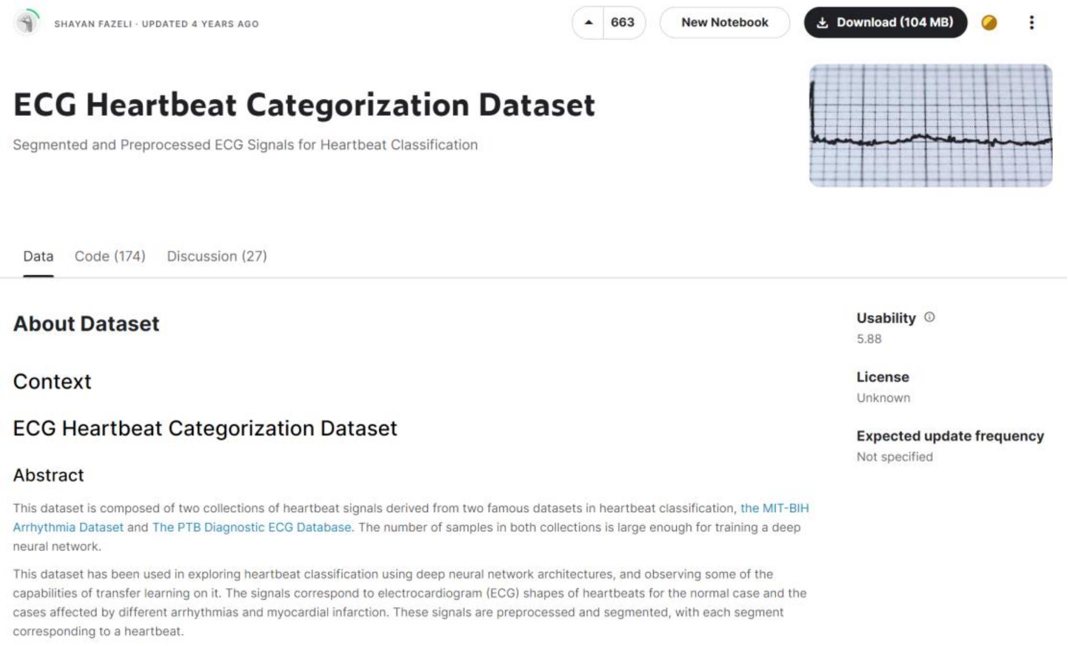
↓ ↓ ↓



↑ CNN이란? ↑

기존 Deep Neural Networks에서
Convolutional 전처리 작업을 추가한 방식.

2.3 데이터셋 소개



분류화된 ECG 심박 데이터셋

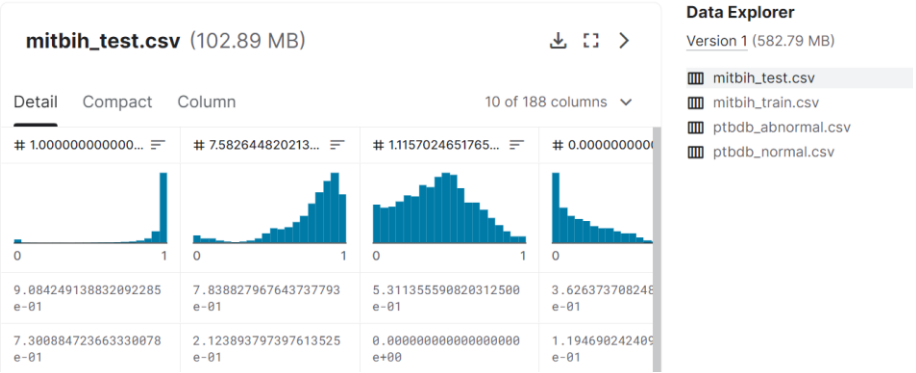
https://www.kaggle.com/datasets/shayanfazeli/heartbeat

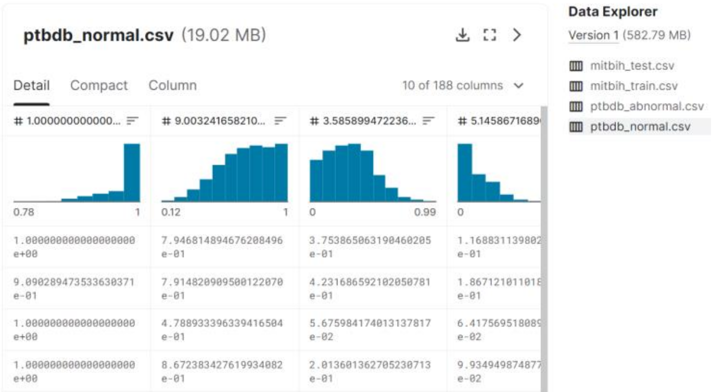
데이터셋 구성

MIT-BIH의 부정맥 데이터셋 (109466샘플)

\+

PTB 진단 ECG DB (14552샘플)





2. 관련 연구 및 내용

2.4 데이터셋 구조

카테고리 분류

0 N: 정상 박동
1 S: 상심실 조기 박동
2 V: 조기 심실 수축
3 F: 심실과 정상 박동의 융합
4 Q: 분류할 수 없는 박동

0 : 정상

1 : 비정상

데이터셋 구조

콘텐츠

부정맥 데이터세트

- 샘플 수: 109446
- 카테고리 수: 5
- 샘플링 주파수: 125Hz
- 데이터 출처: Physionet의 MIT-BIH 부정맥 데이터세트
- 등급: ['N': 0, 'S': 1, 'V': 2, 'F': 3, 'Q': 4]

PTB 진단 ECG 데이터베이스

- 샘플 수: 14552
- 카테고리 수: 2
- 샘플링 주파수: 125Hz
- 데이터 소스: Physionet의 PTB 진단 데이터베이스

2.3 데이터셋 구조

MIT-BIH Arrhythmia Database

George Moody, **Roger Mark**

Published: Feb. 24, 2005. Version: 1.0.0

When using this resource, please cite the original publication:
Moody GB, Mark RG. The impact of the MIT-BIH Arrhythmia Database. IEEE Eng in Med and Biol 20(3):45-50 (May-June 2001). (PMID: 11446209)

Please include the standard citation for PhysioNet: (show more options)
Goldberger, A., Amaral, L., Glass, L., Hausdorff, J., Ivanov, P. C., Mark, R., ... & Stanley, H. E. (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation [Online]. 101 (23), pp. e215–e220.

Files

Total uncompressed size: 104.3 MB.

https://www.physionet.org/content/mitdb/1.0.0/

Access the files

- Download the ZIP file (73.5 MB)
- Access the files using the Google Cloud Storage Browser here. Login with a Google account is required.
- Access the data using the Google Cloud command line tools (please refer to the gsutil documentation for guidance):
  `gsutil -m -u YOUR_PROJECT_ID cp -r gs://mitdb-1.0.0.physionet.org DESTINATION`
- Download the files using your terminal: `wget -r -N -c -np https://physionet.org/files/mitdb/1.0.0/`

Visualize waveforms

Folder Navigation: <base>

Name	Size	Modified
mitdbdir		
x_mitdb		
100.atr	4.5 KB	1992-07-29
100.dat	1.9 MB	1992-07-30
100.hea	143 B	1992-07-30
100.xws	88 B	1999-12-12
101.atr	3.7 KB	1992-07-30
101.dat	1.9 MB	1992-07-30

데이터셋 전처리

https://arxiv.org/pdf/1805.00794.pdf

ECG Heartbeat Classification: A Deep Transferable Representation

Mohammad Kachuee, Shayan Fazeli, Majid Sarrafzadeh
University of California, Los Angeles (UCLA)
Los Angeles, USA

Abstract—**Electrocardiogram (ECG) can be reliably used as a measure to monitor the functionality of the cardiovascular system. Recently, there has been a great attention towards accurate categorization of heartbeats. While there are many commonalities between different ECG conditions, the focus of most studies has been classifying a set of conditions on a dataset annotated for that task rather than learning and employing a transferable knowledge between different tasks. In this paper, we propose a method based on deep convolutional neural networks for the classification of heartbeats which is able to accurately classify five different arrhythmias in accordance with the AAMI EC57 standard. Furthermore, we suggest a method for transferring the knowledge acquired on this task to the myocardial infarction (MI) classification task. We evaluated the proposed method on PhysioNet's MIT-BIH and PTB Diagnostics datasets. According to the results, the suggested method is able to make predictions with the average accuracies of 93.4% and 95.9% on arrhythmia classification and MI classification, respectively.**

Index Terms—**ECG, deep learning, transfer learning, heartbeat, myocardial infraction**

I. INTRODUCTION

ECG is widely used by cardiologists and medical practitioners for monitoring the cardiac health. The main problem with manual analysis of ECG signals, similar to many other time-series data, lies in difficulty of detecting and categorizing different waveforms and morphologies in the signal. For a human, this task is both extensively time-consuming and prone to errors. Note that the proper diagnosis of cardiovascular diseases is of paramount importance since these are the cause of death for about one-third of all deaths around the globe [1]. For instance, millions of people experience irregular heartbeats which can be lethal in some cases. Therefore, accurate and low-cost diagnosis of arrhythmic heartbeats is highly desirable [2].

To address the problems raised with the manual analysis of ECG signals, many studies in the literature explored using machine learning techniques to accurately detect the anomalies in the signal [3], [4]. Most of these approaches involve a preprocessing phase for preparing the signal (e.g., passing it through band-pass filters, etc). Afterwards, the handcrafted features which are mostly statistical summarizations of signal windows are extracted from these signals and used in further analysis for the final classification task. As for the inference engine, conventional machine learning approaches for ECG analysis include Support Vector Machines, multi-layer perceptrons, decision trees, etc. [5], [6], [7]

These handcrafted features provide us with an acceptable representation of the signal, based on recent machine learning studies, automated feature extraction and representation methods are proven to be more scalable and are capable of making more accurate predictions. An end-to-end deep learning framework allows the machine to learn the features that are best suited to the specific task that it is dedicated to carry out [8], [9], [10]. This approach provides us with a more accurate representation of ECG signal using which the machine can compete with a human cardiologist in analyzing the signal [11]. Deep learning approaches, however, contain a tremendously large amount of variables which require massive amounts of data to be trained.

One way of dealing with the need to a massive amount of data is the concept of knowledge transfer between different tasks. In computer vision, as an example, ImageNet dataset along with the state of the art deep learning models have been used to transfer knowledge between different image understanding tasks [12]. As another example, it has been shown that different sentence categorization tasks can share a considerable amount of sentence understanding [13]. On the other hand, there has been limited uses of transfer learning in health informatics. For example, Alaa *et al.* [14] have used the parameters of a Gaussian expert process trained on patients with stable conditions for patients with deteriorating conditions.

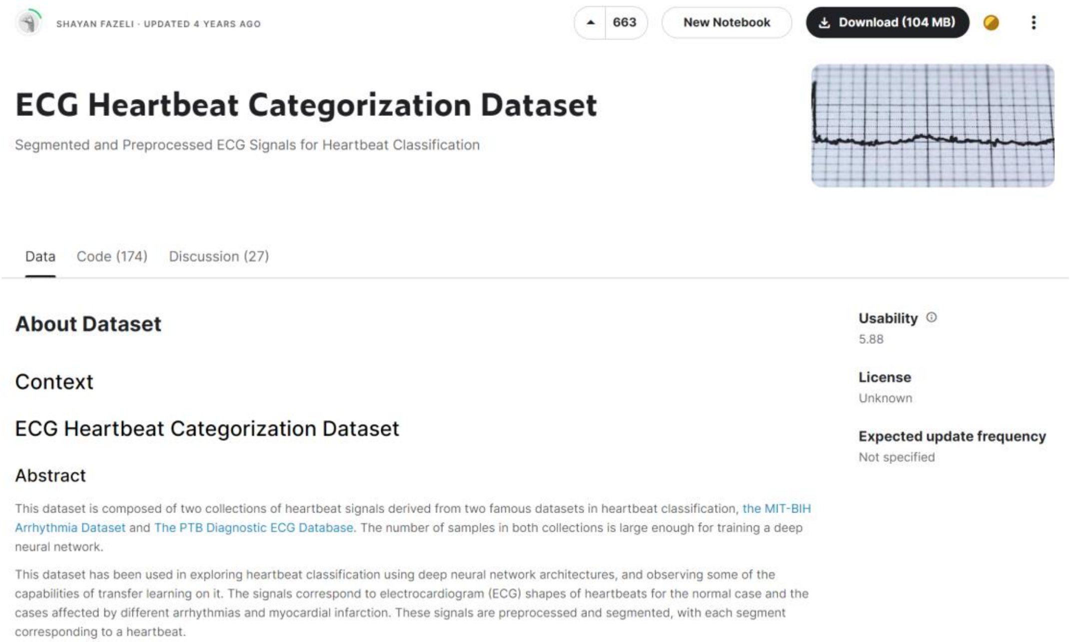
In this paper, we propose a novel framework for ECG analysis that is able to represent the signal in a way that is transferable between different tasks. For this to happen, we describe a deep neural network architecture which offers a considerable capacity for learning such representations. This network has been trained on the task of arrhythmia detection for learning which it is plausible to assume that the model needs to learn most of the shape-related features of the ECG signal. Also, we have a large amount of labeled data for this task, which makes it easy to train a network with a large amount of parameters. Furthermore, we show that the signal representation learned from this task is successfully transferable to the task MI prediction using ECG signals. This method allows us to use these deep representations to share knowledge between ECG recognition tasks for which enough information may not be available for training a deep architecture.

The rest of this paper is organized as follows. Section II explains the datasets used in this study. Section III presents the

좌측 자료를 위 논문의 방식대로 가공(전처리)하였다.

https://www.kaggle.com/datasets/nelsonsharma/ecg-lead-2-dataset-physionet-open-access
해당 링크 참고

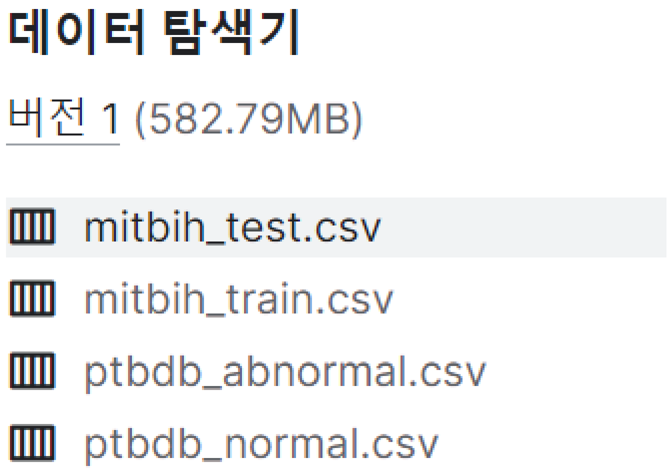
2.3 데이터셋 구조



분류화된 ECG 심박 데이터셋

https://www.kaggle.com/datasets/shayanfazeli/heartbeat

데이터셋 분할



mit-bih의 부정맥 데이터베이스를 전처리한 뒤 8:2로 분할하여 사용하였다.

2.5 코드 설명

```python
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.
```

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix
from keras.utils.np_utils import to_categorical
from sklearn.utils import class_weight
import warnings
warnings.filterwarnings('ignore')
```

사용된 라이브러리

1. os
   : 경로 찾는 등 os 기능 수행
2. numpy
   : 백터, 행렬 계산, 대수학
3. pandas
   : 데이터 분석용. 객체를 만들어 관리함.
4. matplotlib
   : 그래프를 그려준다.
5. seaborn
   : matplotlib 기반으로 미려한 시각화 제공
6. sklearn
   : 머신러닝 라이브러리. 다양한 모듈 존재
7. keras
   : 사용자 친화적 신경망 라이브러리
8. warnings
   : 경고 메시지 무시하기 위해 사용

https://www.kaggle.com/code/be11sky/arrhythmia-on-ecg-classification-using-cnn/edit

2.5 코드 설명

```
train_df=pd.read_csv('/kaggle/input/heartbeat/mitbih_train.csv',header=None)
test_df=pd.read_csv('/kaggle/input/heartbeat/mitbih_test.csv',header=None)
```

```
train_df[187]=train_df[187].astype(int)
equilibre=train_df[187].value_counts()
print(equilibre)
```

```
0    72471
4     6431
2     5788
1     2223
3      641
Name: 187, dtype: int64
```

train_df = 학습모델

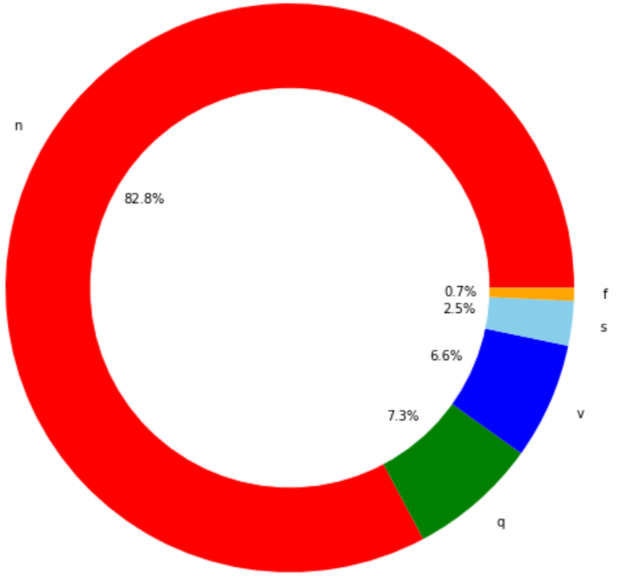
test_df = 테스트 모델

0-186번째까지 데이터,
187(188번째)는 NSVFQ(0 1 2 3 4)

https://www.kaggle.com/code/be11sky/arrhythmia-on-ecg-classification-using-cnn/edit

2.5 코드 설명

```
plt.figure(figsize=(20,10))
my_circle=plt.Circle( (0,0), 0.7, color='white')
plt.pie(equilibre, labels=['n','q','v','s','f'], colors=['red','green','blue','skyblue','orange'],autopct='%1.1f%%')
p=plt.gcf()
p.gca().add_artist(my_circle)
plt.show()
```



색상을 활용한 원형 그래프 분류

빨간색 = 정상

초록색 = 분류 불가

파란색 = 조기 심실 수축

하늘색 = 상심실 조기 박동

주황색 = 심실과 정상 박동의 융합 Fusion

-〉 비중 차이를 극복하기 위해 클래스 가중치 알고리즘 대신 리샘플링 기법 선택

2.5 코드 설명

```
from sklearn.utils import resample
df_1=train_df[train_df[187]==1]
df_2=train_df[train_df[187]==2]
df_3=train_df[train_df[187]==3]
df_4=train_df[train_df[187]==4]
df_0=(train_df[train_df[187]==0]).sample(n=20000,random_state=42)

df_1_upsample=resample(df_1,replace=True,n_samples=20000,random_state=123)
df_2_upsample=resample(df_2,replace=True,n_samples=20000,random_state=124)
df_3_upsample=resample(df_3,replace=True,n_samples=20000,random_state=125)
df_4_upsample=resample(df_4,replace=True,n_samples=20000,random_state=126)

train_df=pd.concat([df_0,df_1_upsample,df_2_upsample,df_3_upsample,df_4_upsample])
```

df_0,1,2,3,4에 5개의 카테고리를 나누어 담는다.

sklearn 라이브러리의 resample 모듈

-> 데이터를 일관된 방식으로 업/다운샘플링 하기 위해서. 여기서는 업샘플링을 위해 사용되었다.

concat은 pandas의 함수로 데이터를 결합해준다. 여기서는 방금 각각 random state를 매개로 주어 업샘플링한 데이터들을 결합하는 데에 사용하였다.

2.5 코드 설명

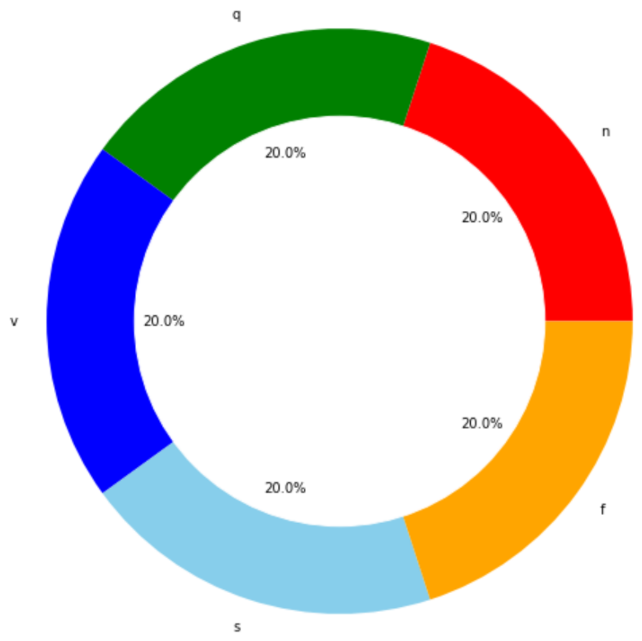
```
equilibre=train_df[187].value_counts()
print(equilibre)
```

```
4    20000
3    20000
2    20000
1    20000
0    20000
Name: 187, dtype: int64
```

방금 업샘플링한 자료들이 설정한 수 20000개씩이 맞는 지 확인하였다.

2.5 코드 설명

```
plt.figure(figsize=(20,10))
my_circle=plt.Circle( (0,0), 0.7, color='white')
plt.pie(equilibre, labels=['n','q','v','s','f'], colors=['red','green','blue','skyblue','orange'],autopct='%1.1f%%')
p=plt.gcf()
p.gca().add_artist(my_circle)
plt.show()
```



이제 각 카테고리의 샘플 수가 모두 2000 개로 동일하게 갖춰졌다.

2.5 코드 설명

```
c=train_df.groupby(187,group_keys=False).apply(lambda train_df : train_df.sample(1))
```

+ Code + Markdown

I take one sample per class and i store it in a datafrmae in order to have an exmeple.

5개의 행(카테고리)와 188개의 열로 구성되었다.

```
c
```

[13]:

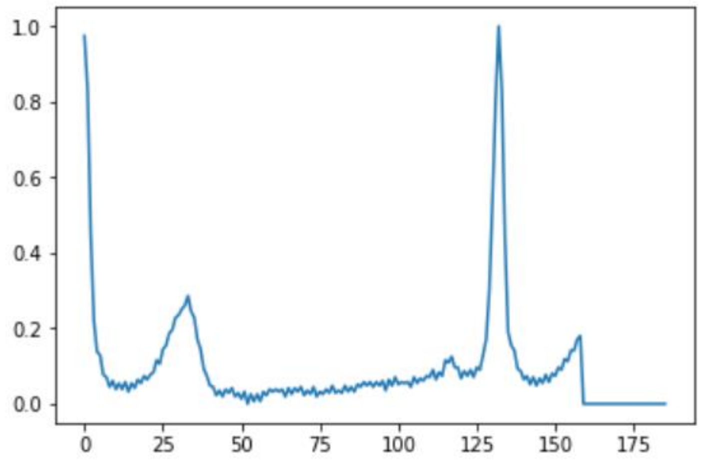
	0	1	2	3	4	5	6	7	8	9	...	178	179	180	181	182	183	184	185	186	187
17005	0.974277	0.832797	0.450161	0.221865	0.138264	0.128617	0.077170	0.070740	0.045016	0.061093	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
74239	0.979381	1.000000	0.350515	0.061856	0.092784	0.134021	0.149485	0.216495	0.257732	0.201031	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
78497	0.167708	0.223958	0.372917	0.495833	0.538542	0.543750	0.567708	0.579167	0.591667	0.594792	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2
80995	0.841365	0.851406	0.915663	1.000000	0.927711	0.522088	0.248996	0.158635	0.124498	0.108434	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3
82767	0.726496	0.638177	0.552707	0.424501	0.290598	0.170940	0.099715	0.045584	0.055556	0.088319	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4

5 rows × 188 columns

2.5 코드 설명

```
plt.plot(c.iloc[0,:186])
```

[14]: [<matplotlib.lines.Line2D at 0x7fa734363048>]



+ Code + Markdown

Here is a normal beat. I don't have something particular to say on that class.

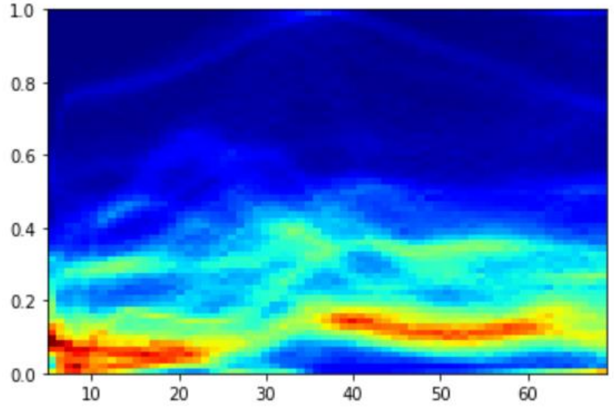
정상적인 심장 박동을 plot 그래프로 그려 내었다. 다음과 같은 분포를 가지는 것을 볼 수 있다.

2.5 코드 설명

```
def plot_hist(class_number,size,min_,bins):
    img=train_df.loc[train_df[187]==class_number].values
    img=img[:,min_:size]
    img_flatten=img.flatten()

    final1=np.arange(min_,size)
    for i in range (img.shape[0]-1):
        tempo1=np.arange(min_,size)
        final1=np.concatenate((final1, tempo1), axis=None)
    print(len(final1))
    print(len(img_flatten))
    plt.hist2d(final1,img_flatten, bins=(bins,bins),cmap=plt.cm.jet)
    plt.show()
```

```
plot_hist(0,70,5,65)
```

```
1300000
1300000
```

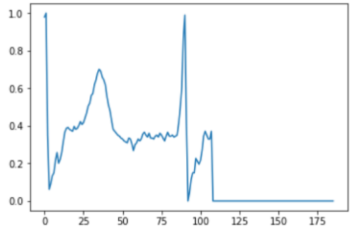


matplotlib의 hist2d 함수를 활용하여 2D 히스토그램 plot을 그리면 이런 모양이 나온다.

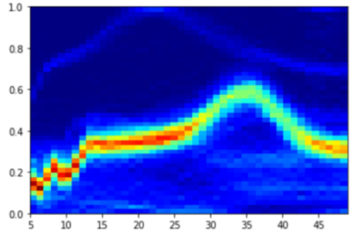
2. 관련 연구 및 내용

2.5 코드 설명

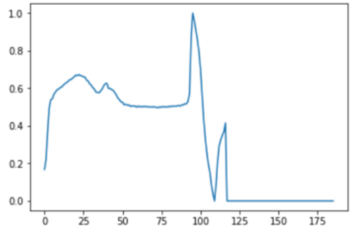
```
[18]: plt.plot(c.iloc[1,:186])
[18]: [<matplotlib.lines.Line2D at 0x7fa733ab0048>]
```



```
[19]: plot_hist(1,50,5,45)
900000
900000
```



```
[20]: plt.plot(c.iloc[2,:186])
[20]: [<matplotlib.lines.Line2D at 0x7fa7339ea710>]
```

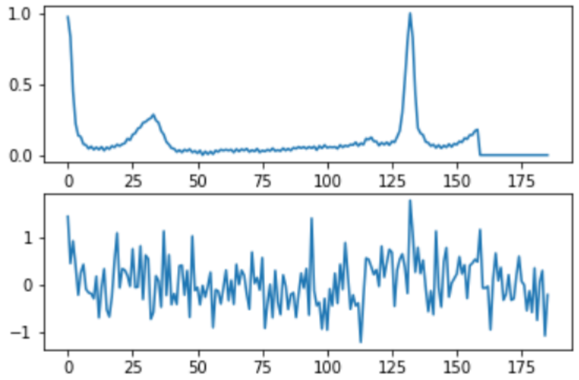


정상 심박인 카테고리 0 뿐만 아니라 1, 2,3 에 대해서도 시각화를 해준다.

2.5 코드 설명

```
def add_gaussian_noise(signal):
    noise=np.random.normal(0,0.5,186)
    return (signal+noise)
```

```
tempo=c.iloc[0,:186]
bruiter=add_gaussian_noise(tempo)

plt.subplot(2,1,1)
plt.plot(c.iloc[0,:186])

plt.subplot(2,1,2)
plt.plot(bruiter)

plt.show()
```



train 모델의 일반화를 위해 가우시안 잡음을 더해준다.

2.5 코드 설명

```python
def network(X_train,y_train,X_test,y_test):


    im_shape=(X_train.shape[1],1)
    inputs_cnn=Input(shape=(im_shape), name='inputs_cnn')
    conv1_1=Convolution1D(64, (6), activation='relu', input_shape=im_shape)(inputs_cnn)
    conv1_1=BatchNormalization()(conv1_1)
    pool1=MaxPool1D(pool_size=(3), strides=(2), padding="same")(conv1_1)
    conv2_1=Convolution1D(64, (3), activation='relu', input_shape=im_shape)(pool1)
    conv2_1=BatchNormalization()(conv2_1)
    pool2=MaxPool1D(pool_size=(2), strides=(2), padding="same")(conv2_1)
    conv3_1=Convolution1D(64, (3), activation='relu', input_shape=im_shape)(pool2)
    conv3_1=BatchNormalization()(conv3_1)
    pool3=MaxPool1D(pool_size=(2), strides=(2), padding="same")(conv3_1)
    flatten=Flatten()(pool3)
    dense_end1 = Dense(64, activation='relu')(flatten)
    dense_end2 = Dense(32, activation='relu')(dense_end1)
    main_output = Dense(5, activation='softmax', name='main_output')(dense_end2)


    model = Model(inputs= inputs_cnn, outputs=main_output)
    model.compile(optimizer='adam', loss='categorical_crossentropy',metrics = ['accuracy'])


    callbacks = [EarlyStopping(monitor='val_loss', patience=8),
             ModelCheckpoint(filepath='best_model.h5', monitor='val_loss', save_best_only=True)]

    history=model.fit(X_train, y_train,epochs=40,callbacks=callbacks, batch_size=32,validation_data=(X_test,y_test))
    model.load_weights('best_model.h5')
    return(model,history)
```

CNN을 train과 test에 적용하는 내용

2.5 코드 설명

```
def evaluate_model(history,X_test,y_test,model):
    scores = model.evaluate((X_test),y_test, verbose=0)
    print("Accuracy: %.2f%%" % (scores[1]*100))

    print(history)
    fig1, ax_acc = plt.subplots()
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.title('Model - Accuracy')
    plt.legend(['Training', 'Validation'], loc='lower right')
    plt.show()

    fig2, ax_loss = plt.subplots()
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.title('Model- Loss')
    plt.legend(['Training', 'Validation'], loc='upper right')
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.show()
    target_names=['0','1','2','3','4']

    y_true=[]
    for element in y_test:
        y_true.append(np.argmax(element))
    prediction_proba=model.predict(X_test)
    prediction=np.argmax(prediction_proba,axis=1)
    cnf_matrix = confusion_matrix(y_true, prediction)
```

CNN을 train과 test에 적용하는 내용

2.5 코드 설명

```
from keras.layers import Dense, Convolution1D, MaxPool1D, Flatten, Dropout
from keras.layers import Input
from keras.models import Model
from keras.layers.normalization import BatchNormalization
import keras
from keras.callbacks import EarlyStopping, ModelCheckpoint

model,history=network(X_train,y_train,X_test,y_test)
```

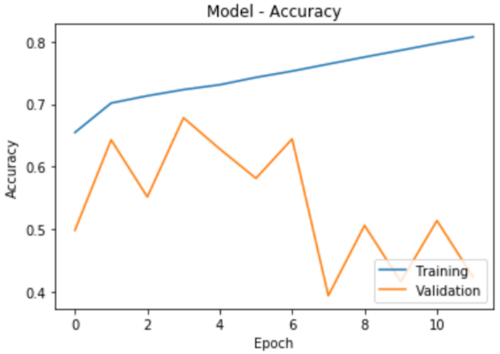
신경망 학습하는 코드.

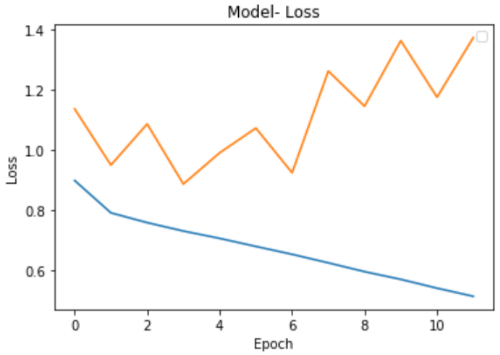
```
Train on 100000 samples, validate on 21892 samples
Epoch 1/40
100000/100000 [==============================] - 107s 1ms/step - loss: 0.8982 - accuracy: 0.6547 - val_loss: 1.1357 - val_accuracy: 0.4983
Epoch 2/40
100000/100000 [==============================] - 105s 1ms/step - loss: 0.7911 - accuracy: 0.7017 - val_loss: 0.9495 - val_accuracy: 0.6428
Epoch 3/40
100000/100000 [==============================] - 105s 1ms/step - loss: 0.7586 - accuracy: 0.7133 - val_loss: 1.0858 - val_accuracy: 0.5516
Epoch 4/40
100000/100000 [==============================] - 105s 1ms/step - loss: 0.7307 - accuracy: 0.7232 - val_loss: 0.8867 - val_accuracy: 0.6782
Epoch 5/40
100000/100000 [==============================] - 105s 1ms/step - loss: 0.7064 - accuracy: 0.7310 - val_loss: 0.9900 - val_accuracy: 0.6283
Epoch 6/40
100000/100000 [==============================] - 105s 1ms/step - loss: 0.6798 - accuracy: 0.7427 - val_loss: 1.0721 - val_accuracy: 0.5813
Epoch 7/40
100000/100000 [==============================] - 106s 1ms/step - loss: 0.6536 - accuracy: 0.7528 - val_loss: 0.9241 - val_accuracy: 0.6443
Epoch 8/40
100000/100000 [==============================] - 104s 1ms/step - loss: 0.6253 - accuracy: 0.7641 - val_loss: 1.2612 - val_accuracy: 0.3937
Epoch 9/40
100000/100000 [==============================] - 105s 1ms/step - loss: 0.5961 - accuracy: 0.7753 - val_loss: 1.1447 - val_accuracy: 0.5060
Epoch 10/40
100000/100000 [==============================] - 106s 1ms/step - loss: 0.5706 - accuracy: 0.7861 - val_loss: 1.3620 - val_accuracy: 0.4161
Epoch 11/40
100000/100000 [==============================] - 104s 1ms/step - loss: 0.5413 - accuracy: 0.7971 - val_loss: 1.1747 - val_accuracy: 0.5137
Epoch 12/40
100000/100000 [==============================] - 104s 1ms/step - loss: 0.5143 - accuracy: 0.8074 - val_loss: 1.3721 - val_accuracy: 0.4233
```

2.5 코드 설명

```
evaluate_model(history,X_test,y_test,model)
y_pred=model.predict(X_test)
```

```
Accuracy: 67.82%
<keras.callbacks.callbacks.History object at 0x7fa6fc418160>
```





+ Code + Markdown

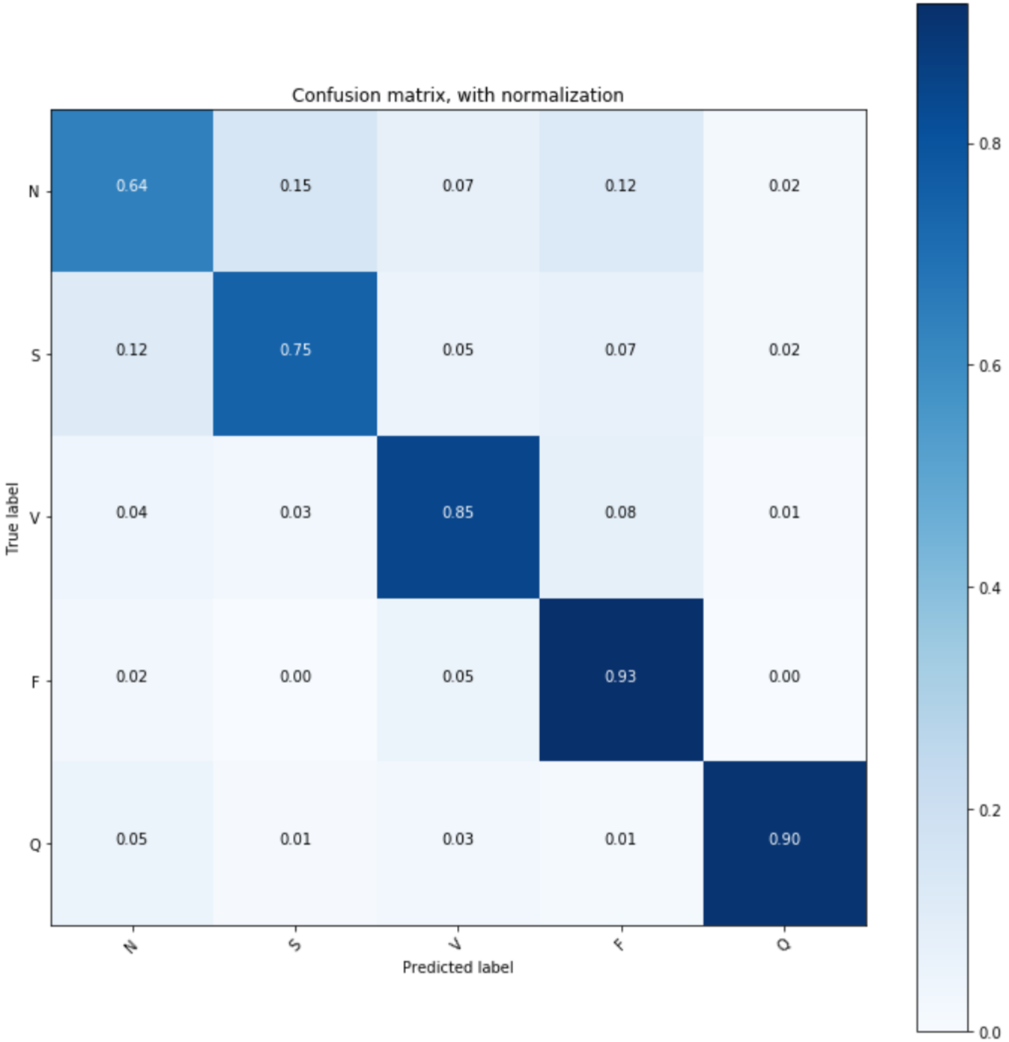
i take the next function from : https://www.kaggle.com/coni57/model-from-arxiv-1805-00794

Train을 학습하여 test를 predict한 결과. Accuracy가 67.82%로 그렇게 높지는 않은 모습을 보인다.

2.5 코드 설명

```python
import itertools
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

# Compute confusion matrix
cnf_matrix = confusion_matrix(y_test.argmax(axis=1), y_pred.argmax(axis=1))
np.set_printoptions(precision=2)

# Plot non-normalized confusion matrix
plt.figure(figsize=(10, 10))
plot_confusion_matrix(cnf_matrix, classes=['N', 'S', 'V', 'F', 'Q'],normalize=True,
                      title='Confusion matrix, with normalization')
plt.show()
```

Normalized confusion matrix



Random Resampling을 통해 Upsampling을 하였다고는 하지만

샘플 수량이 다른 세 카테고리에 비해 부족했기 때문에 상대적으로 낮은 정확도를 보이는 S&F클래스를 볼 수 있다.

10만여개의 데이터 샘플을 통해 딥러닝 신경망 학습을 하기에 충분하다고 Note의 저자는 생각했으나 결국 샘플의 부족이 정확도의 발목을 잡았다.

내용 요약

Part 3.

3.1 내용 요약

1. 사용된 라이브러리

os, numpy, pandas, matplotlib,
seaborn, sklearn, keras, warnings

2. sklearn 업샘플링

카테고리별 비중을 동일하게 맞추기 위해 가중치 대신 리샘플링 함수를 사용.

3. Add Noise

일반화를 위해 가우시안 잡음을 더해줌

4. CNN 적용

5. 신경망 학습

6. Predict & Visualization

과제 소감

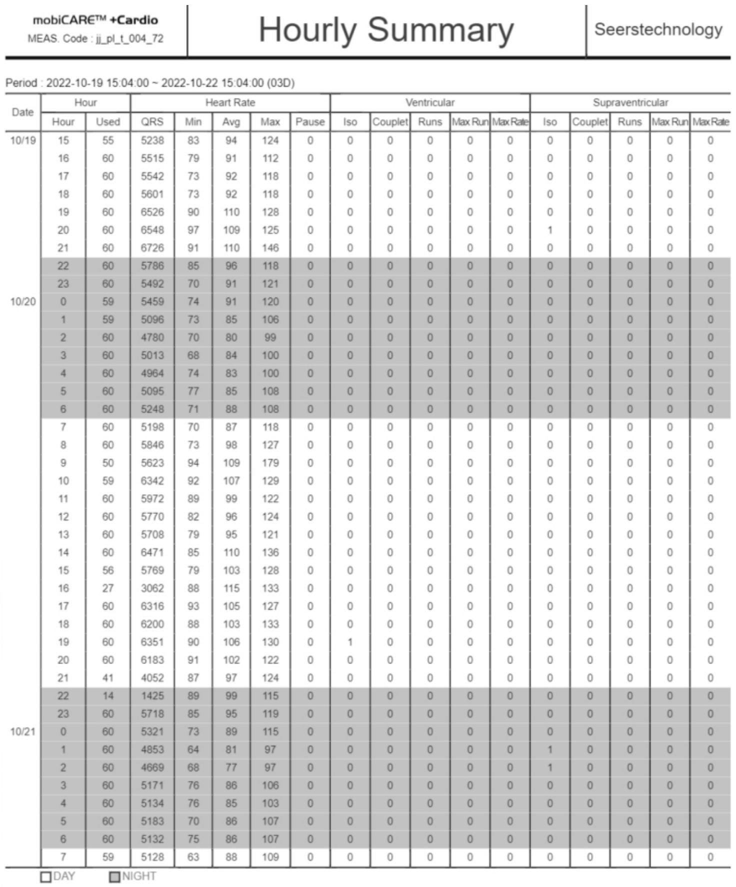
Part 4.

4.1 아쉬운 점

←같은 난관에 부딪힌 유저들

1. Kaggle Note의 정보 기재 부족으로 인한 데이터 가공(전처리)의 어려움

4.1 아쉬운 점



mobiCARE™ +Cardio
MEAS. Code : jj_pl_t_004_72

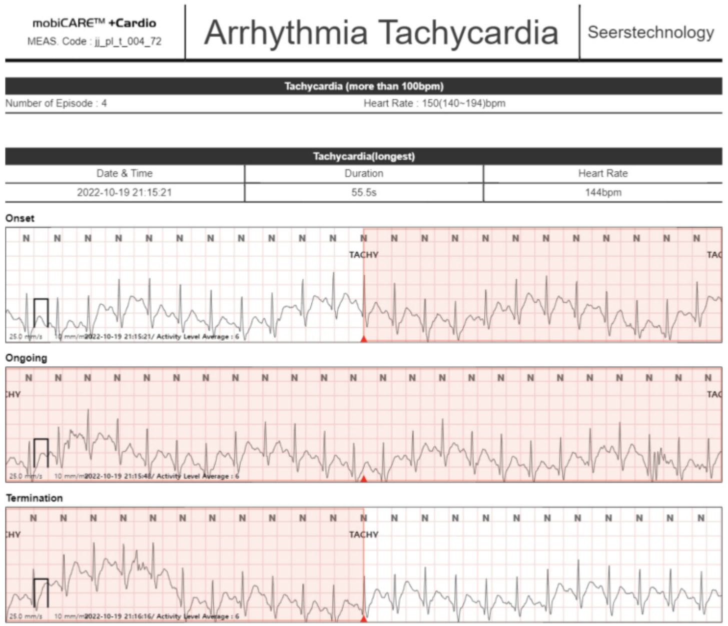
Hourly Summary

Seerstechnology

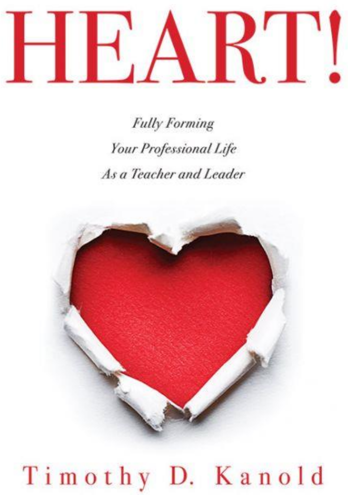
Period : 2022-10-19 15:04:00 ~ 2022-10-22 15:04:00 (03D)

Date	Hour		Heart Rate					Ventricular					Supraventricular				
	Hour	Used	QRS	Min	Avg	Max	Pause	Iso	Couplet	Runs	Max Run	Max Rate	Iso	Couplet	Runs	Max Run	Max Rate
10/19	15	55	5238	83	94	124	0	0	0	0	0	0	0	0	0	0	0
	16	60	5515	79	91	112	0	0	0	0	0	0	0	0	0	0	0
	17	60	5542	73	92	118	0	0	0	0	0	0	0	0	0	0	0
	18	60	5601	73	92	118	0	0	0	0	0	0	0	0	0	0	0
	19	60	6526	90	110	128	0	0	0	0	0	0	0	0	0	0	0
	20	60	6548	97	109	125	0	0	0	0	0	0	1	0	0	0	0
	21	60	6726	91	110	146	0	0	0	0	0	0	0	0	0	0	0
	22	60	5786	85	96	118	0	0	0	0	0	0	0	0	0	0	0
	23	60	5492	70	91	121	0	0	0	0	0	0	0	0	0	0	0
10/20	0	59	5459	74	91	120	0	0	0	0	0	0	0	0	0	0	0
	1	59	5096	73	85	106	0	0	0	0	0	0	0	0	0	0	0
	2	60	4780	70	80	99	0	0	0	0	0	0	0	0	0	0	0
	3	60	5013	68	84	100	0	0	0	0	0	0	0	0	0	0	0
	4	60	4964	74	83	100	0	0	0	0	0	0	0	0	0	0	0
	5	60	5095	77	85	108	0	0	0	0	0	0	0	0	0	0	0
	6	60	5248	71	88	108	0	0	0	0	0	0	0	0	0	0	0
	7	60	5198	70	87	118	0	0	0	0	0	0	0	0	0	0	0
	8	60	5846	73	98	127	0	0	0	0	0	0	0	0	0	0	0
	9	50	5623	94	109	179	0	0	0	0	0	0	0	0	0	0	0
	10	59	6342	92	107	129	0	0	0	0	0	0	0	0	0	0	0
	11	60	5972	89	99	122	0	0	0	0	0	0	0	0	0	0	0
	12	60	5770	82	96	124	0	0	0	0	0	0	0	0	0	0	0
	13	60	5708	79	95	121	0	0	0	0	0	0	0	0	0	0	0
	14	60	6471	85	110	136	0	0	0	0	0	0	0	0	0	0	0
	15	56	5769	79	103	128	0	0	0	0	0	0	0	0	0	0	0
	16	27	3062	88	115	133	0	0	0	0	0	0	0	0	0	0	0
	17	60	6316	93	105	127	0	0	0	0	0	0	0	0	0	0	0
	18	60	6200	88	103	133	0	0	0	0	0	0	0	0	0	0	0
	19	60	6351	90	106	130	0	1	0	0	0	0	0	0	0	0	0
	20	60	6183	91	102	122	0	0	0	0	0	0	0	0	0	0	0
	21	41	4052	87	97	124	0	0	0	0	0	0	0	0	0	0	0
	22	14	1425	89	99	115	0	0	0	0	0	0	0	0	0	0	0
	23	60	5718	85	95	119	0	0	0	0	0	0	0	0	0	0	0
10/21	0	60	5321	73	89	115	0	0	0	0	0	0	0	0	0	0	0
	1	60	4853	64	81	97	0	0	0	0	0	0	1	0	0	0	0
	2	60	4669	68	77	97	0	0	0	0	0	0	1	0	0	0	0
	3	60	5171	76	86	106	0	0	0	0	0	0	0	0	0	0	0
	4	60	5134	76	85	103	0	0	0	0	0	0	0	0	0	0	0
	5	60	5183	70	86	107	0	0	0	0	0	0	0	0	0	0	0
	6	60	5132	75	86	107	0	0	0	0	0	0	0	0	0	0	0
	7	59	5128	63	88	109	0	0	0	0	0	0	0	0	0	0	0

DAY NIGHT



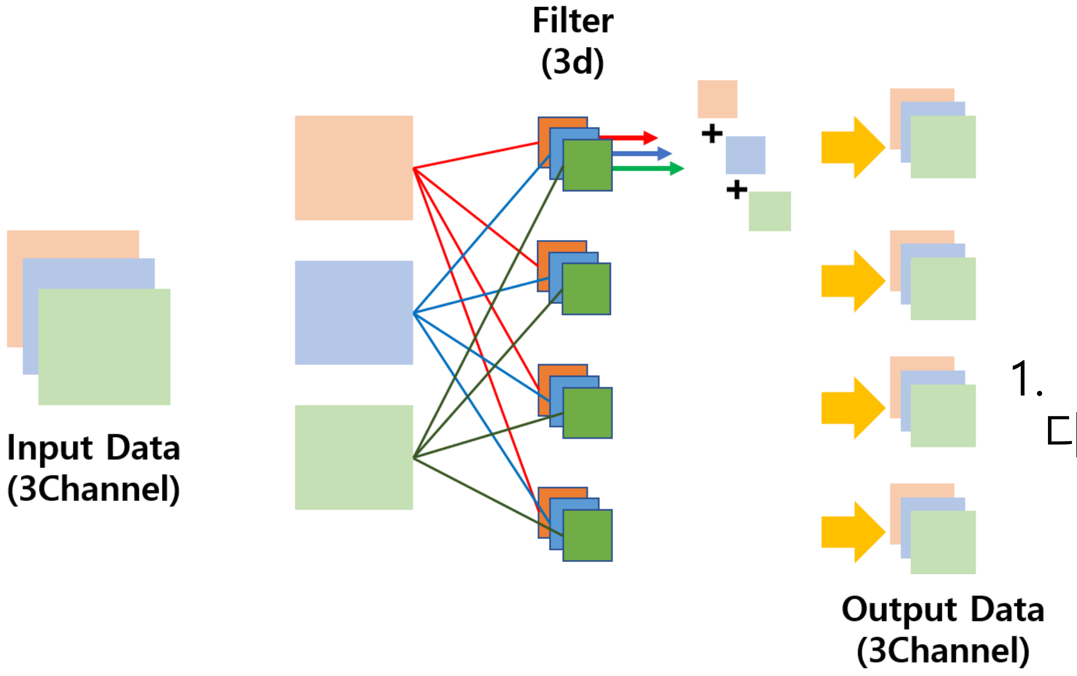
비교군으로 생각해둔
전문 인력에 의한 판독 자료



2. 사용하려고 한 개인 데이터의 형식 불일치로 인한 활용 불가

4. 과제 소감

4.2 배운 점



1. CNN이 영상 처리만이 아닌
   다양한 데이터 처리 분야에서 쓰인다는 점.

2. 데이터를 목적에 맞게 가공하는 경험
https://www.kaggle.com/datasets/nelsonsharma/ecg-lead-2-dataset-physionet-open-access

이 상 으 로
발 표 를
마 칩 니 다 .