심장질환 환자 ECG 데이터 분석을 위한 딥러닝 기법 설계 및 경량화 모델 구축

풀악셀(A)

중간보고서

신다윗(201824510)
안주현(201824520)
이재현(201824554)

지도교수
송길태 교수님

목 차

1. 요구조건 및 제약 사항 분석에 대한 수정사항

1.1 요구조건

본 졸업과제는 딥러닝을 기반으로 하는 심장 질병 분류 모델에 관한 것으로, 보다 상세하게는 심장질환에 따라 각각의 다른 파형으로 출력되는 심전도를 이용하여 다양한 조합의 데이터셋을 생성하고, 생성한 데이터셋에 의해 학습된 딥러닝 모델을 이용하여 분류대상이 되는 심전도가 어떤 종류의 심장질환에 속하는지 여부를 결정하는 것이 목표이다.

부정맥과 관상동맥질환(심장동맥질환)의 진단에 많은 검사들이 이용되고 있으나, 그 중에서도 심전도(Electrocardiogram : ECG)를 이용한 검사는 많은 장점을 가지며 임상에서 가장 많이 사용되는 검사이다. 심전도는 심장에서 발생하는 전기적인 신호(심전도 신호)를 측정하여 심박동과 관련된 전위를 신체 표면에서 도형으로 기록한 것으로, 이 기록지를 판독하여 심장에서부터 전극까지의 전도계통의 이상 유무를 확인하여 질환 유무를 판별한다. 일반적으로 부정맥은 질환이 있다, 없다로 진단하는 것이 아니라, 정상이 아닌 것으로 판단된 경우, 즉, 부정맥으로 판단된 경우 부정맥의 종류까지 구분해야하므로, 최근에는 딥러닝을 활용한 심전도 분석과 인공지능 모델이 개발되어 부정맥 진단에 이용되고 있다. 우리는 이러한 딥러닝 모델을 개발하여 경량화까지 진행하여야 한다.

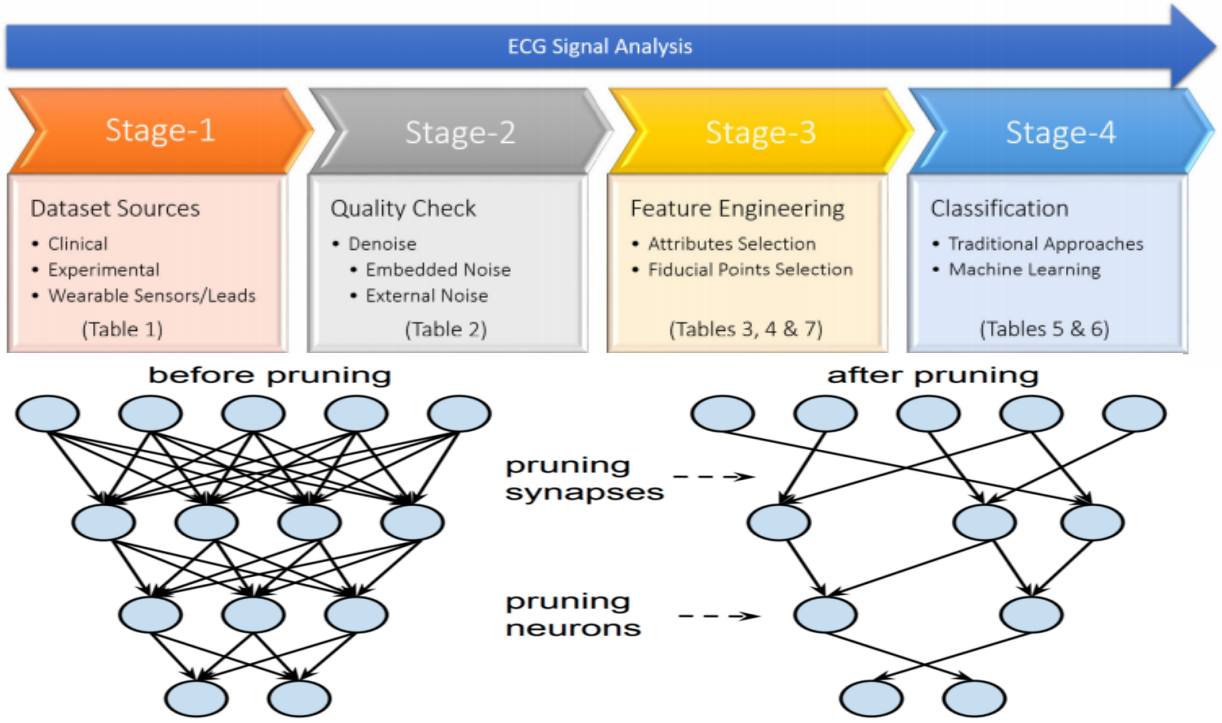

그림 1. ECG 분석 모델 개발 과정 및 경량화

1.2 제약 사항 분석에 대한 수정사항

CNN을 이용한 Base 모델을 설계하고 학습을 진행하였지만 정확도가 낮게 나와서 Inception을 이용하여 모델을 구축하였다. 정확도를 더 올리기 위해 Inception 모델 외에도 더 복잡하고 세련된 모델을 사용하여 문제를 해결할 계획이다. 다양한 최신 네트워크 구조들을 검토하고, 하이퍼파라미터 튜닝도 계속해서 시도해볼 계획이다.

2. 설계 상세화 및 변경 내역

2.1 데이터 설명

데이터의 저작권은 “크리에이티브 커먼즈 저작자 표시 4.0 국제 공중 라이선스[CC BY 4.0]”을 따른다.

심전도(ECG)는 신뢰할 수 있고 비침습적이며 심장 기능을 모니터링하는 데 널리 사용되는 접근 방식으로 가슴, 발목 및 손목에 있는 12개의 전극에서 파동 정보가 있다. 아래의 그림을 통해 학습시킬 ECG 데이터를 볼 수 있다.

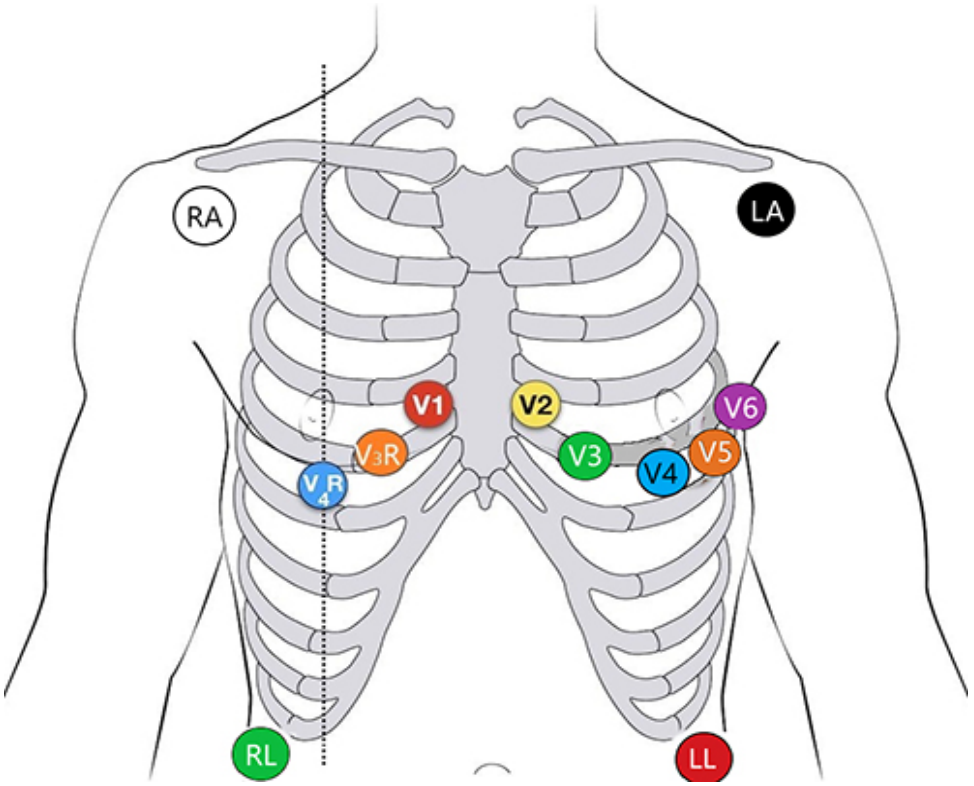

그림 2-1. ECG 데이터의 12개 전극

출처 : www.urmc.rochester.edu

1. I 리드 (Lead I)
    a. 왼쪽 팔에서 오른쪽 팔로의 전기 차이를 측정한다.
2. II 리드 (Lead II)
    a. 오른쪽 다리에서 왼쪽 팔로의 전기 차이를 측정한다.
3. III 리드 (Lead III)
    a. 오른쪽 다리에서 왼쪽 다리로의 전기 차이를 측정한다.
4. aVR 리드 (Augmented Vector Right)
    a. 전기적 중심을 오른쪽 상부에서 볼 때의 전압을 측정한다.
5. aVL 리드 (Augmented Vector Left)
    a. 전기적 중심을 왼쪽 상부에서 볼 때의 전압을 측정한다.
6. aVF 리드 (Augmented Vector Foot)
    a. 전기적 중심을 다리에서 볼 때의 전압을 측정한다.
7. V1 리드 (Chest Lead 1)
    a. 가슴의 오른쪽 가장자리 위치에서의 전기 활동을 측정한다.
8. V2 리드 (Chest Lead 2)
    a. 가슴의 왼쪽 가장자리 위치에서의 전기 활동을 측정한다.
9. V3 리드 (Chest Lead 3)
    a. 가슴의 중앙 위치에서의 전기 활동을 측정한다.
10. V4 리드 (Chest Lead 4)
    a. 가슴의 왼쪽 중앙 위치에서의 전기 활동을 측정한다.
11. V5 리드 (Chest Lead 5)
    a. 가슴의 왼쪽 안쪽 중앙 위치에서의 전기 활동을 측정한다.
12. V6 리드 (Chest Lead 6)
    a. 가슴의 왼쪽 최하단 위치에서의 전기 활동을 측정한다.

학습시킨 모델로 심전도에 대해 5가지의 클래스로 분류하고, 단일 분류가 아닌 Multi-Label 분류를 한다.

1. 모델이 분류하는 class labels 5가지 = ['NORM', 'MI', 'STTC', 'CD', 'HYP']
    a. NORM (Normal):
        i. 정상적인 심전도 신호를 나타낸다. 심박수와 심장의 전기 활동이 정상 범위 내에서 발생하는 것을 나타낸다.
    b. MI (Myocardial Infarction):
        i. 심근경색을 나타내는 심전도 신호를 나타낸다. 심근 경색은 심장 근육 조직이 혈류 부족으로 손상을 입은 상태로, 흔히 심실 경색 또는 심근 경색으로 불린다.

c. STTC (ST-T Change):
    i. ST-T 변화를 보이는 심전도 신호를 나타낸다. ST-T 변화는 심전도의 ST 세그먼트와 T파의 모양이 정상 범위에서 벗어나는 현상으로, 심장의 특정 문제나 부상을 나타낼 수 있다.
d. CD (Conduction Disorder):
    i. 전도 장애를 가진 심전도 신호를 나타낸다. 심장의 전기 신호가 정상적으로 전달되지 않는 상황으로, 이로 인해 심방 세동이나 심실 세동 등이 발생할 수 있다.
e. HYP (Hypertrophy):
    i. 심근비대를 나타내는 심전도 신호를 나타낸다. 심근비대는 심장 근육이 비정상적으로 커지는 상태로, 심장 질환의 한 형태다

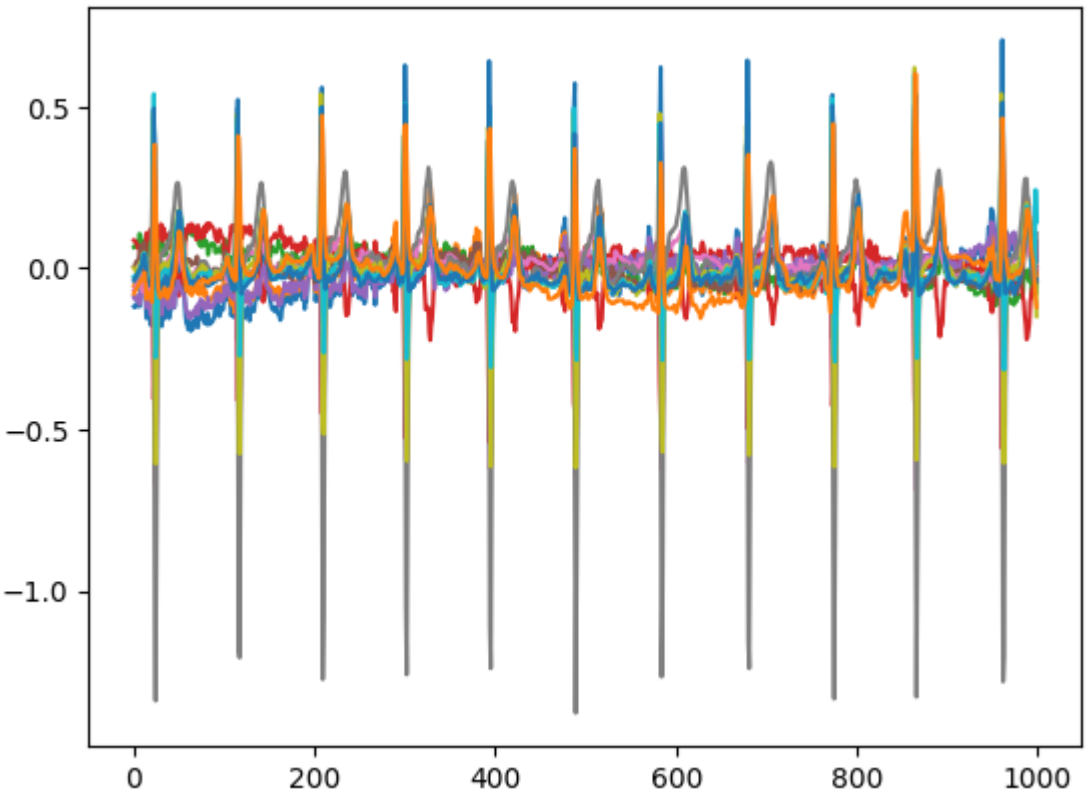

그림 2-2. ECG 데이터 샘플 1에 대한 모든 신호가 중첩된 그래프

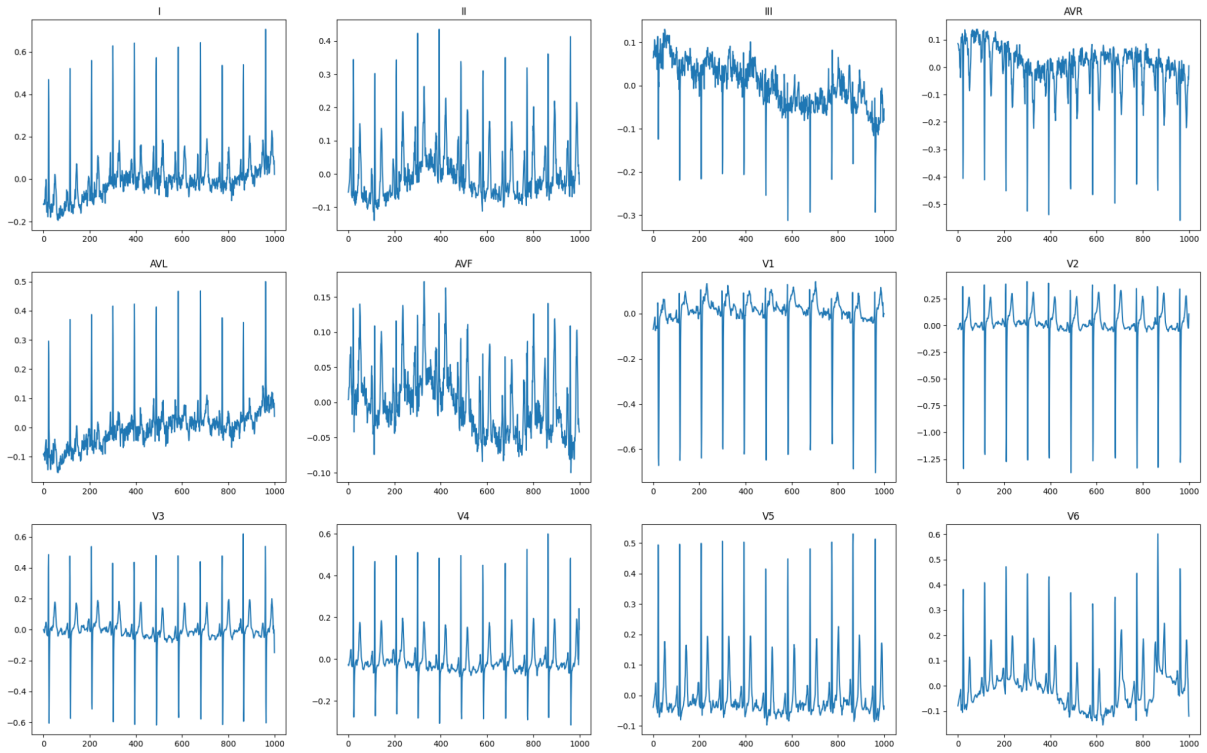

그림 2-3. ECG 데이터 샘플 1에 대한 각 신호를 출력한 그래프

2.2 모델 설계

Inception 기반의 CNN모델을 사용하여 심장질환 분류를 수행하였다. Inception 모델은 다양한 필터 크기를 사용하여 특징을 추출하고, 병렬로 구성된 레이어들을 통해 보다 복잡한 패턴을 학습하는 데 효과적이다. 이러한 특징은 ECG 데이터와 같이 시계열적인 특성을 가진 데이터의 분류에 유리하다.

2.3 모델 구조

Inception 모델의 기본 아키텍처는 입력 레이어, 다양한 컨볼루션 레이어, 풀링 레이어, fully-connected 레이어로 구성된다. 입력 레이어는 ECG 데이터의 형태에 맞게 설정되며, 컨볼루션 레이어에서는 다양한 필터 크기를 사용하여 특징을 추출한다. 풀링 레이어는 공간 차원을 축소하여 계산 비용을 줄이고, fully-connected 레이어는 분류를 수행하는 데 사용된다.

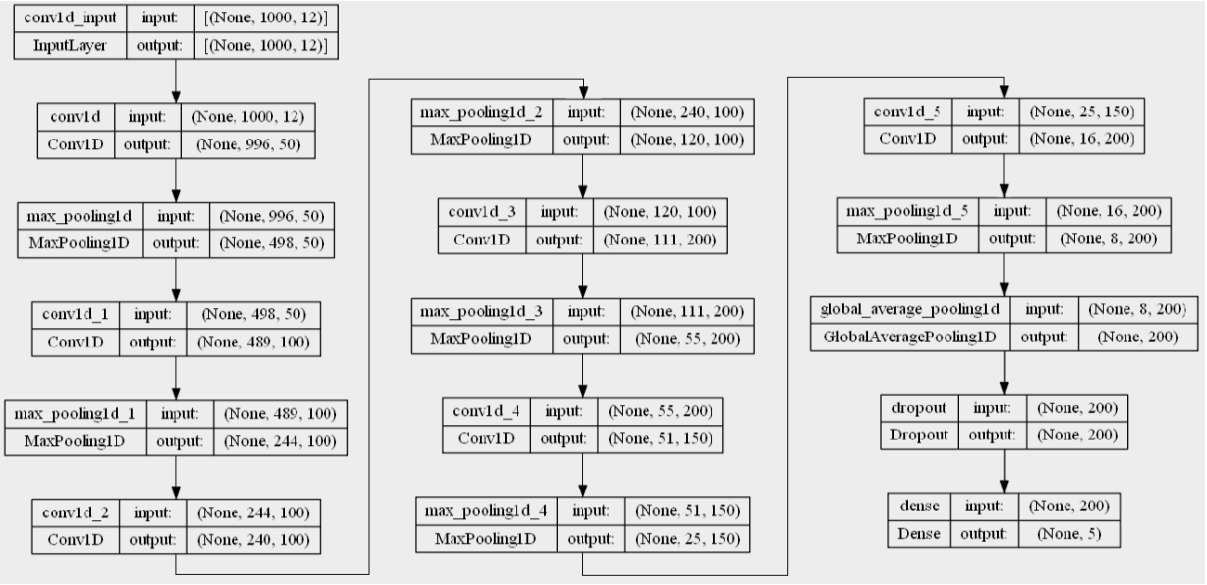

그림 3. CNN Base 모델 구조

2.4 모델 평가

CNN Base 모델의 정확도는 Training 데이터에 88% 정도로, 좀 더 복잡한 모델을 설계할 필요가 있다.

```
97/97 [==============================] - 168s 2s/step - loss: 0.0650 - accuracy: 0.8828
Epoch 31/100
97/97 [==============================] - 167s 2s/step - loss: 0.0552 - accuracy: 0.8877
Epoch 32/100
97/97 [==============================] - 168s 2s/step - loss: 0.0507 - accuracy: 0.8933
Epoch 33/100
97/97 [==============================] - 167s 2s/step - loss: 0.0469 - accuracy: 0.8949
```

그림 4. CNN Base 모델의 학습 과정

2.5 변경 내역

정확도를 더 올리기 위해 Inception 모델 외에도 더 복잡하고 세련된 모델을 사용하여 문제를 해결할 계획이다. 정확도(accuracy)를 구하기 위해 sklearn accuracy를 사용하였는데 ecg 데이터에서 multi-label 문제가 상당히 존재하여 다른 metric를 사용해 볼 계획이다. federated transfer learning 기법을 시도해 볼 예정이다.

3. 갱신된 과제 추진 계획

개발 일정	상태	마감일
프로젝트 착수보고서 작성	완료	2023년 5월 22일
ECG 데이터 분석 및 특징 파악, 이상치 처리	완료	2023년 7월 21일
중간 보고서 작성	완료	2023년 8월 2일
딥러닝 모델 구축 및 성능 테스트 및 최적화	진행 중	2023년 8월 15일
딥러닝 모델 경량화	시작되지 ...	2023년 9월 28일
최종 보고서 작성	시작되지 ...	2023년 9월 30일
졸업과제 발표	시작되지 ...	2023년 10월 6일

4. 구성원별 진척도

이름	진척도
신다윗	ECG 데이터 분석 및 특징 파악, 이상치 처리 Inception을 이용한 CNN 모델 설계 및 학습
안주현	ECG 데이터 전처리 CNN Base 모델 설계 및 학습
이재현	ECG 데이터 분석 및 전처리, 이상치 처리 CNN Base 모델 설계 및 학습

5. 보고 시점까지의 과제 수행 내용 및 중간 결과

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_12 (Conv1D)          (None, 996, 50)           3050

 max_pooling1d_12 (MaxPooli  (None, 498, 50)           0
 ng1D)

 conv1d_13 (Conv1D)          (None, 489, 100)          50100

 max_pooling1d_13 (MaxPooli  (None, 244, 100)          0
 ng1D)

 conv1d_14 (Conv1D)          (None, 240, 100)          50100

 max_pooling1d_14 (MaxPooli  (None, 120, 100)          0
 ng1D)

 conv1d_15 (Conv1D)          (None, 111, 200)          200200

 max_pooling1d_15 (MaxPooli  (None, 55, 200)           0
 ng1D)

 conv1d_16 (Conv1D)          (None, 51, 150)           150150

 max_pooling1d_16 (MaxPooli  (None, 25, 150)           0
 ng1D)

 conv1d_17 (Conv1D)          (None, 16, 200)           300200

 max_pooling1d_17 (MaxPooli  (None, 8, 200)            0
 ng1D)

 global_average_pooling1d_3  (None, 200)               0
  (GlobalAveragePooling1D)

 dropout_3 (Dropout)         (None, 200)               0

 dense_3 (Dense)             (None, 5)                 1005

=================================================================
Total params: 754805 (2.88 MB)
Trainable params: 754805 (2.88 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

그림 5. CNN Base 모델

Convolutional layer와 Max Pooling이 반복되는 단순한 구조로 설계한 Base 모델의 summary이다. 학습이 필요한 파라미터의 수는 약 75만개로 복잡하고 크기가 큰 모델은 아니지만 학습과정에서 Training 데이터에 대해 약 88%의 정확도를 보였다.

```
97/97 [==============================] - 169s 2s/step - loss: 0.0749 - accuracy: 0.8832
Epoch 29/100
97/97 [==============================] - 167s 2s/step - loss: 0.0675 - accuracy: 0.8826
Epoch 30/100
97/97 [==============================] - 168s 2s/step - loss: 0.0650 - accuracy: 0.8828
Epoch 31/100
97/97 [==============================] - 167s 2s/step - loss: 0.0552 - accuracy: 0.8877
Epoch 32/100
97/97 [==============================] - 168s 2s/step - loss: 0.0507 - accuracy: 0.8933
Epoch 33/100
97/97 [==============================] - 167s 2s/step - loss: 0.0469 - accuracy: 0.8949
Epoch 34/100
97/97 [==============================] - 167s 2s/step - loss: 0.0431 - accuracy: 0.8935
Epoch 35/100
97/97 [==============================] - 167s 2s/step - loss: 0.0355 - accuracy: 0.9003
Epoch 36/100
97/97 [==============================] - 168s 2s/step - loss: 0.0402 - accuracy: 0.8944
```

그림 6. CNN Base 모델의 학습 과정

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_1 (InputLayer)        [(None, 1000, 12)]        0         []

 conv1d (Conv1D)             (None, 994, 100)          8500      ['input_1[0][0]']

 max_pooling1d (MaxPooling1D)  (None, 497, 100)        0         ['conv1d[0][0]']

 max_pooling1d_1 (MaxPooling1D)  (None, 497, 100)      0         ['max_pooling1d[0][0]']

 conv1d_1 (Conv1D)           (None, 497, 100)          10100     ['max_pooling1d[0][0]']

 conv1d_2 (Conv1D)           (None, 497, 100)          30100     ['max_pooling1d[0][0]']

 conv1d_3 (Conv1D)           (None, 497, 200)          100200    ['max_pooling1d[0][0]']

 conv1d_4 (Conv1D)           (None, 497, 100)          10100     ['max_pooling1d_1[0][0]']

 concatenate (Concatenate)   (None, 497, 500)          0         ['conv1d_1[0][0]',
                                                                  'conv1d_2[0][0]',
                                                                  'conv1d_3[0][0]',
                                                                  'conv1d_4[0][0]']

 batch_normalization (BatchNorm  (None, 497, 500)      2000      ['concatenate[0][0]']
 alization)

 max_pooling1d_2 (MaxPooling1D)  (None, 248, 500)      0         ['batch_normalization[0][0]']

 max_pooling1d_3 (MaxPooling1D)  (None, 248, 500)      0         ['max_pooling1d_2[0][0]']

 conv1d_5 (Conv1D)           (None, 248, 200)          100200    ['max_pooling1d_2[0][0]']

 conv1d_6 (Conv1D)           (None, 248, 200)          300200    ['max_pooling1d_2[0][0]']
```

```
conv1d_7 (Conv1D)              (None, 248, 200)     500200      ['max_pooling1d_2[0][0]']

conv1d_8 (Conv1D)              (None, 248, 100)     50100       ['max_pooling1d_3[0][0]']

concatenate_1 (Concatenate)    (None, 248, 700)     0           ['conv1d_5[0][0]',
                                                                 'conv1d_6[0][0]',
                                                                 'conv1d_7[0][0]',
                                                                 'conv1d_8[0][0]']

batch_normalization_1 (BatchNo  (None, 248, 700)    2800        ['concatenate_1[0][0]']
rmalization)

max_pooling1d_4 (MaxPooling1D)  (None, 124, 700)    0           ['batch_normalization_1[0][0]']

max_pooling1d_5 (MaxPooling1D)  (None, 124, 700)    0           ['max_pooling1d_4[0][0]']

conv1d_9 (Conv1D)              (None, 124, 200)     140200      ['max_pooling1d_4[0][0]']

conv1d_10 (Conv1D)             (None, 124, 200)     420200      ['max_pooling1d_4[0][0]']

conv1d_11 (Conv1D)             (None, 124, 200)     700200      ['max_pooling1d_4[0][0]']

conv1d_12 (Conv1D)             (None, 124, 100)     70100       ['max_pooling1d_5[0][0]']

concatenate_2 (Concatenate)    (None, 124, 700)     0           ['conv1d_9[0][0]',
                                                                 'conv1d_10[0][0]',
                                                                 'conv1d_11[0][0]',
                                                                 'conv1d_12[0][0]']

batch_normalization_2 (BatchNo  (None, 124, 700)    2800        ['concatenate_2[0][0]']
rmalization)

max_pooling1d_6 (MaxPooling1D)  (None, 62, 700)     0           ['batch_normalization_2[0][0]']

max_pooling1d_7 (MaxPooling1D)  (None, 62, 700)     0           ['max_pooling1d_6[0][0]']

conv1d_13 (Conv1D)             (None, 62, 200)      140200      ['max_pooling1d_6[0][0]']

conv1d_14 (Conv1D)             (None, 62, 200)      420200      ['max_pooling1d_6[0][0]']

conv1d_15 (Conv1D)             (None, 62, 200)      700200      ['max_pooling1d_6[0][0]']

conv1d_16 (Conv1D)             (None, 62, 100)      70100       ['max_pooling1d_7[0][0]']

concatenate_3 (Concatenate)    (None, 62, 700)      0           ['conv1d_13[0][0]',
                                                                 'conv1d_14[0][0]',
                                                                 'conv1d_15[0][0]',
                                                                 'conv1d_16[0][0]']


batch_normalization_3 (BatchNo  (None, 62, 700)     2800        ['concatenate_3[0][0]']
rmalization)

global_average_pooling1d (Glob  (None, 700)         0           ['batch_normalization_3[0][0]']
alAveragePooling1D)

dropout (Dropout)              (None, 700)          0           ['global_average_pooling1d[0][0]'
                                                                ]

dense (Dense)                  (None, 5)            3505        ['dropout[0][0]']

==================================================================================================
Total params: 3,785,005
Trainable params: 3,779,805
Non-trainable params: 5,200
__________________________________________________________________________________________________
```

표 1. Inception을 이용한 CNN 모델

Inception을 이용하여 설계한 CNN 모델의 summary이다. 학습이 필요한 파라미터의 수는 약 378만개로 CNN Base 모델보다 복잡한 모델이지만 학습과정에서 Training 데이터에 대해 loss값은 많이 줄어들었지만 정확도는 Base 모델과 비슷하게 약 88%를 나타냈다. 따라서 Inception을 이용한 모델을 더 고도화하거나 Inception 모델보다 더 복잡한 모델을 사용하는 방법으로 Training 데이터에 대한 정확도가 90%를 넘는 모델을 설계할 계획이다.

```
Epoch 1/50
246/246 [==============================] - 59s 226ms/step - loss: 0.2913 - accuracy: 0.7058
Epoch 2/50
246/246 [==============================] - 56s 227ms/step - loss: 0.2902 - accuracy: 0.7089
Epoch 3/50
246/246 [==============================] - 56s 228ms/step - loss: 0.2869 - accuracy: 0.7111
Epoch 4/50
246/246 [==============================] - 56s 228ms/step - loss: 0.2866 - accuracy: 0.7092
Epoch 5/50
246/246 [==============================] - 57s 230ms/step - loss: 0.2879 - accuracy: 0.7106
Epoch 6/50
246/246 [==============================] - 57s 230ms/step - loss: 0.2853 - accuracy: 0.7131
Epoch 7/50
246/246 [==============================] - 57s 231ms/step - loss: 0.2835 - accuracy: 0.7124
Epoch 8/50
246/246 [==============================] - 57s 230ms/step - loss: 0.2836 - accuracy: 0.7118
Epoch 9/50
246/246 [==============================] - 57s 230ms/step - loss: 0.2802 - accuracy: 0.7185
Epoch 10/50
246/246 [==============================] - 57s 230ms/step - loss: 0.2813 - accuracy: 0.7146
Epoch 11/50
246/246 [==============================] - 57s 230ms/step - loss: 0.2798 - accuracy: 0.7157
35/35 [==============================] - 2s 63ms/step - loss: 0.3062 - accuracy: 0.6922
Test Loss: 0.3062296509742737
Test Accuracy: 0.6922378540039062
```

그림 7. Inception을 이용한 CNN 모델의 학습 과정

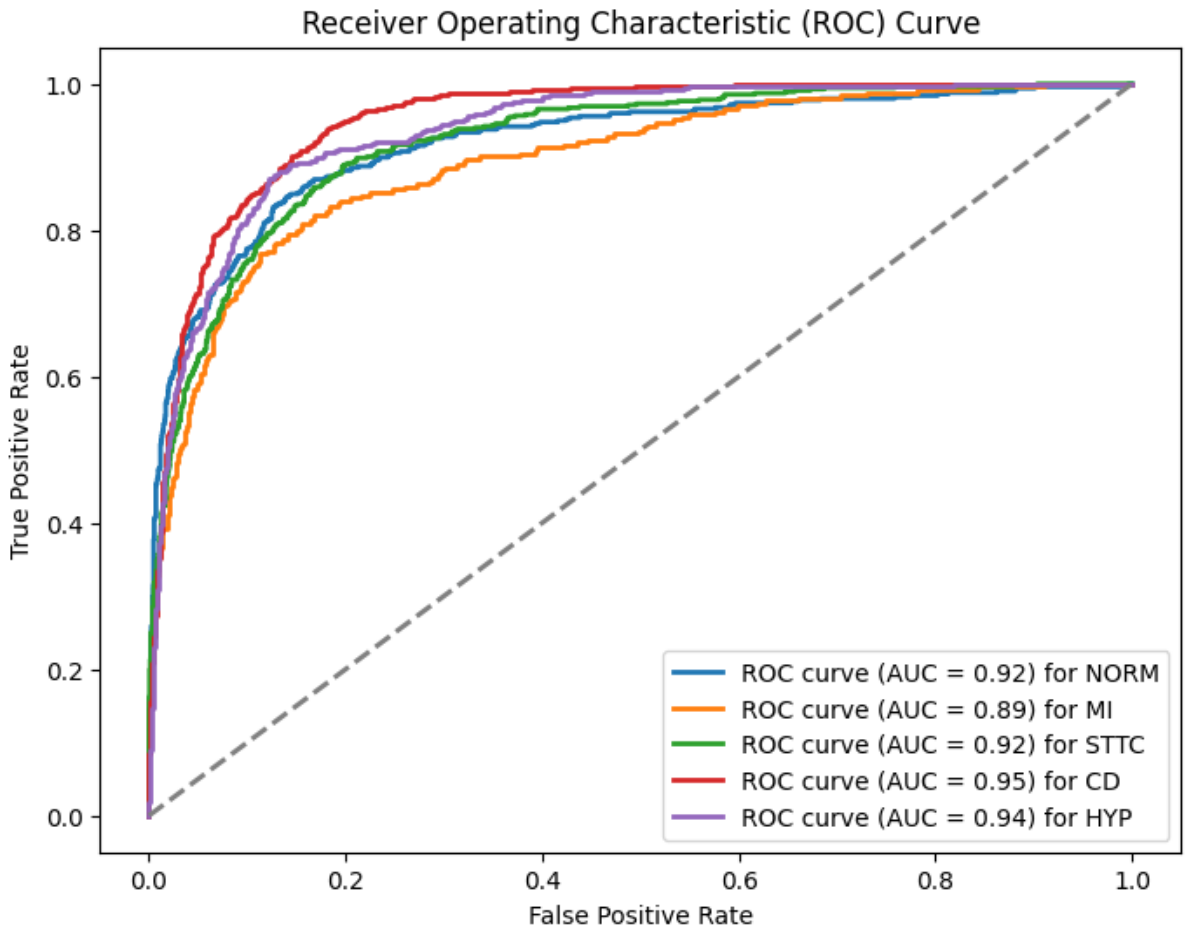

그림 8. Inception을 이용한 CNN 모델의 ROC Curve

ROC Curve의 AUC 값은 각 클래스(혹은 레이블)에 대한 분류 성능을 평가하는데 사용되는 값이다. 분류해야할 클래스 5개에 대한 AUC 값을 보면 전체 AUC 값 평균 0.924 이고 분류를 잘 수행하고 있다. 원하는 성능인 0.95 이상은 도달하지 못했고 조금 더 모델을 개선할 필요가 있음을 느낀다.