2021년도

학사학위논문

**딥러닝 과 MQTT를 활용한**

**당뇨병 입원환자 혈당 관리 어플리케이션 서비스**

**Blood Glucose Management Application Service For Diabetic Inpatients Using Deep Learning And MQTT**

2021년 11월 26일

순천향대학교 공과대학

컴퓨터 공학과

이인재

2021년도

학사학위논문

**딥러닝 과 MQTT를 활용한**

**당뇨병 입원환자 혈당 관리 어플리케이션 서비스**

**Blood Glucose Management Application Service For Diabetic Inpatients Using Deep Learning And MQTT**

2021년 11월 26일

순천향대학교 공과대학

컴퓨터 공학과

이인재

초 록

**목 차**

**제 1 장 서 론**

**제 2 장 이론적 배경**

2.1 당뇨병 시뮬레이터

2.3 MQTT

2.4 LSTM

2.1 Flutter

**제 3 장 환자 데이터 및 예측 모델**

3.1 환자 데이터 생성 시뮬레이터

3.2 모델 훈련 시나리오

3.2.1 계층 별 구성

3.2.2 특성 변수

3.3 모델 평가 결과

**제 4 장 IOT기반 데이터 전송**

4.2 MQTT

4.2.1 연결 정보 구성

4.3 테스트

4.3.1 테스트 결과

**제 5 장 스마트폰 애플리케이션**

3.1 애플리케이션 설계

3.2 애플리케이션 구현

**제 6 장 실험 및 검증**

6.1 평가 기준

**참고문헌**

**1 장 서론**

현대에 살아가는 사람들에게 있어서 더 이상 자연사는 거의 보기 드문 경우가 되었다. 통계청에서 제공하는2020년도 사망 원인 통계에 따르면 전년도 총 사망자 수 30만명 대상으로 대부분 사망원인 추이는 각각 악성신생물(암)이 27%로 가장 높으며, 심장 질환 및 폐렴, 당뇨 등의 질환이 그 뒤를 따른다. 이러한 통계를 기반으로 볼 때, 질병과 현대인들과의 관계에는 필연적인 연관성이 존재함을 시사한다.

본 논문에서는 위에서 언급된 당뇨라 불리는, 합병증의 시발점이 되는 만성질환을 다룬다. 1970년대까지만 해도 국내에서 해당 질환을 겪는 인구는 전체의 1.5%밖에 되지 않았다. 하지만 2018년도 기준 국내 당뇨병 유병률은 13.8%라는 30여년의 기간동안 9배의 증가폭을 보였으며 더 나아가 2019년 기준 전 세계 성인이 9.3%의 당뇨 유병률을 관측했다. 결과적으로 해를 거듭할수록 당뇨의 유병률은 점점 더 높아지고 있는 추세이다.  
소변으로 포도당이 빠져나가면서 체내 수분을 같이 끌고 나가는 해당 질환은 급성대사성 합병증과 동맥경화, 만성 신부전증과 같은 만성 합병증을 일으키는 등 신체에 치명적인 문제를 일으킨다. 당뇨는 크게 1형 당뇨병과 2형 당뇨병으로 분류된다. 1형 당뇨병은 유전적, 환경적 요인에 의해 췌장에서 인슐린이 분비되지 않아 발생하며 2형 당뇨병의 경우에는 인슐린의 분비량이 상대적으로 적거나 체내에서 인슐린의 활용이 온전히 이루어지지 못할 경우에 발생한다.

본 논문에서 설명하는 어플리케이션 서비스는 이러한 당뇨 환자, 특히 병원 내 입원해 있는 환자를 대상으로 MQTT프로토 콜을 사용한 혈당값을 실시간으로 받아들이고, 사전에 시뮬레이터로 추출한 하루 치의 혈당데이터와 LSTM(Long Short-Term Memory)를 기반으로 구성한 딥러닝 모델을 거쳐 30분 후의 혈당데이터의 값을 스마트폰 어플리케이션에 전달 후 시각화한다. 그 과정에서 일정 수준의 혈당값이 감지가 될 경우, 스마트폰에 푸쉬 알림이 전해지는 서비스를 제공하는 과정을 서술한다.

실시간 데이터 전달 및 사전에 모델을 구성하는데 사용된 시뮬레이터는 Python기반 라이브러리 Simglucose를 사용했다. 해당 라이브러리에서는 연령대, 환자의 신체 특성을 정의할 수 있으며 식사와 관련된 부분을 시나리오화 하여 커스텀 할 수 있다는 장점이 있다.   
딥러닝 모델은 앞서 언급한 LSTM 신경망을 기반으로 구성되어 있으며, 학습을 위해 사용된 데이터는 10명의 환자들의 하루치 혈당값이다. 이에 더하여 특성 변수로써 유의미한 값을 갖는 체중을 추가하여 단순히 혈당으로만 구성되어 있는 모델보다 더 높은 정확도를 기대할 수 있도록 하였다. 어플리케이션과의 통신은 MQTT프로토콜 브로커의 역할을 할 수 있는 경량 프로그램 Mosquitto를 사용하였으며 이는Ubuntu18.04기반 가상머신에서 상시 작동할 수 있도록 하였다.  
결과적으로 앞서 서술한 데이터와 데이터 통신의 결과는 Flutter 프레임워크로 만들어진 모바일 어플리케이션 내에서 동작하여 사용자가 입원 환자의 혈당을 모니터링 및 알림을 받을 수 있다.

**2 장 이론적 배경**

**2.1. 시뮬레이터**

논문 및 서비스를 구현하는 과정에서 실제 환자의 데이터를 구하기에는 어려움이 따른다.   
따라서 실제 환자의 데이터를 대체하기 위한 시뮬레이터로 Python의 Simglucose라이브러리를 채택했다. Simglucose는 제1형 당뇨환자의 혈당 통제 방식을 모방하는 알고리즘을 채택하여 사용하고 있으며 파이썬 상에서 혈당 측정 센서, 인슐린 투여량, 식사량, 신체 조건 등 환자의 상황과 유사한 상황을 시나리오 상에서 튜닝 할 수 있다. 본 논문에서는 혈당의 계산 방식을 기본 계산 방식인 BB(Basal-Bolus)controller를 통해 계산하였으며, 특정 인물의 경우엔 python의 tuple 형식의 식사 데이터를 추가하여 혈당 데이터를 추출하였다.

**2.2. MQTT**

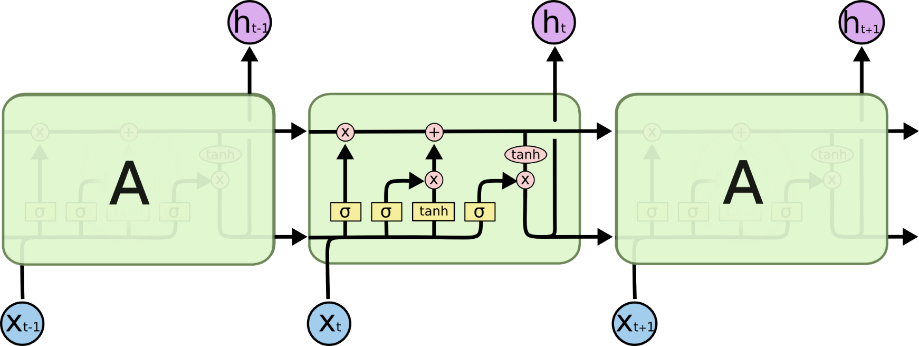
MQTT(Message Queueing Telemetry Transport)는 ISO 표준 발행 – 구독 기반의 메시징 프로토콜이다. 해당 프로토콜은 TCP/IP 상에서 동작하며, 독특하게도 발행자(Publisher) 및 구독자(Subscriber) 사이에 브로커(Broker)라 불리는 별도의 메시지 큐가 존재한다. 해당 프로토콜을 이용하여 계층을 구성할 경우, 1:N 통신구축에 있어서 타 프로토콜에 비하여 상대적으로 유용하기 때문에 IOT 센서를 통한 데이터 송수신에 큰 메리트가 있다.

MQTT는 0, 1, 2 총 3단계의 QoS(Quality of Service)를 지원하며 본 논문에서는 QoS 2를   
사용하였다. QoS 2은 Exactly One, 가장 정확하지만 가장 느린 전송속도를 갖고 있다는 특징이 있다. 메시지는 처리 과정에서 최소 두 쌍의 전송이 발생하는데, 첫 번째 전송 쌍에서 송신자는 메시지를 전송하고 수신자에게서 메시지를 저장했다는 수신확인을 받는다. 송신자가 수신확인을 수신하지 않는 경우, 메시지는 수신확인이 수신될 때까지 DUP 플래그가 설정되어 다시 송신된다. 두 번째 전송 쌍에서 송신자는 수신자에게 메시지 “PUBREL”의 처리를 완료할 수 있다고 전달하며 송신자가 “PUBREL” 메시지의 수신확인을 수신하지 않은 경우 수신확인이 수신될 때까지 “PUBREL” 메시지가 다시 송신된다. 송신자는 “PUBREL” 메시지에 대한 수신확인을 수신하면 저장했던 메시지를 삭제한다. 메시지를 다시 처리하지 않는다는 가정 하에 수신자가 첫 번째 또는 두 번째 단계에서 메시지를 처리하는 경우도 발생할 수 있으며 수신자가 브로커인 경우 이는 메시지를 구독자에게 발행합니다. 수신자가 클라이언트인 경우 메시지는 구독자 애플리케이션으로 전달된다. 해당 경우에는 수신자는 송신자에게 메시지 처리가 완료되었다는 완료 메시지를 송신한다.

본 논문에서는 시나리오 상 환자의 5분단위 혈당 데이터를 받아오는 상황을 가정하였다. 따라서 QoS 2의 수준에 맞추어 데이터의 전달을 수행하였으며, MQTT의 브로커 역할을 하는 미들웨어로 오픈 프로그램인 Mosquitto를 사용하였다. 브로커에 연결을 수행하기 위한 토픽은 후술할 LSTM모델과 어플리케이션에 전달하기 위해 각각 별도로 지정하고 전달할 데이터의 형식은 Json 타입의 메시지를 전송한다.

**2.3. LSTM**

LSTM (Long Short-Term Memory) 신경망은 RNN(Recurrent Neural Network)의 한 종류로, 기존 RNN의 단점으로 지적 받는 층이 많은 레이어에서 레이어 간의 거리가 멀어질수록 해당정보의 기억의존도가 떨어진다는 점을 체인형태의 신경망 구조로 해결한 신경망이다.



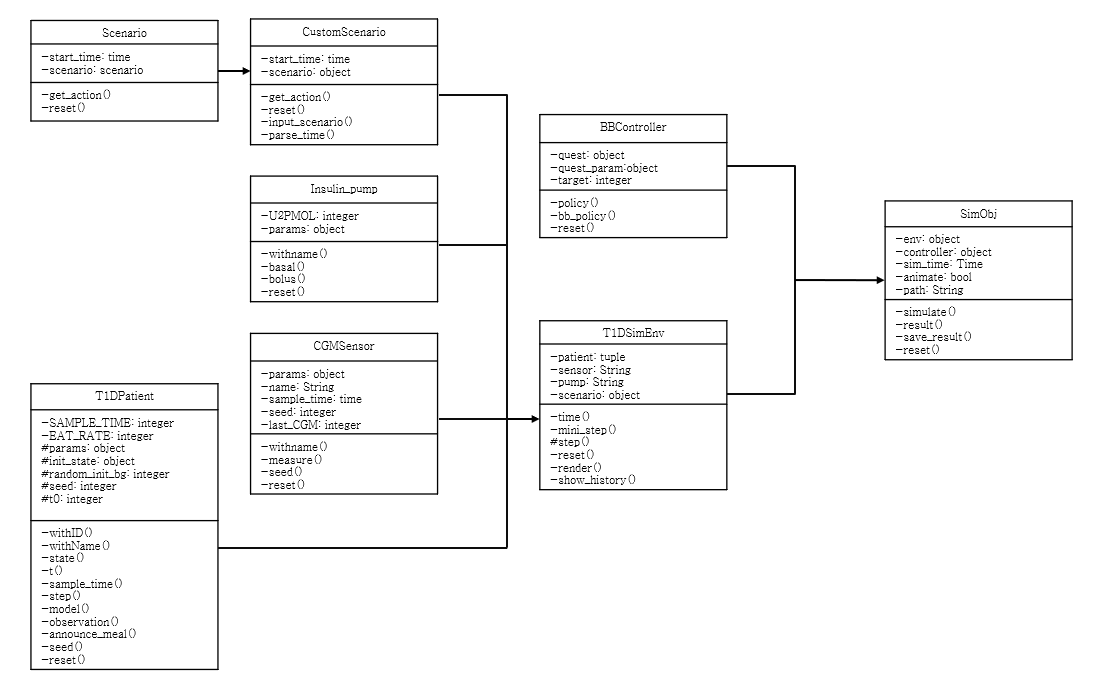
내부 구성에는 메모리 셀로 이루어진 Cell State와 이를 바탕으로 동작하는 3개의 게이트, 그리고정보 전달과정에서 연산을 수행하는 3개의 Sigmoid 함수와 tanh함수를 기반으로 동작하는 레이어들이 있다. Cell State는 그림()과 같이 선형 연산을 바탕으로 신경망 전체를 가로지르는 형태로 존재하며 정보를 전달한다. Cell State에는 2개의 gate가 들어오며 각각 forgetting gate, input gate의 명칭을 갖고 있다. Forget gate Layer는 입력계층 xt과 이전 값 ht-1을 받아 Sigmoid   
활성화 함수를 통해 0과 1사이의 값을 과거의 상태 값인 Ct-1 전달하며 Ct-1의 값이 0일 경우   
모든 정보를 말소시킨다. 반대로 Ct-1의 값이 1일 경우 정보를 보존한다는 원리로 동작한다.  
Input gate Layer의 역할은 어떠한 정보를 Cell State에 저장할 것인지 분류한다. 입력계층 xt과 이전 값 ht-1 전의 값을 Sigmoid함수를 통해 0과 1사이의 값을 갖는 it를 출력한다. it는 앞서   
서술한 Forget gate와 같이 0에 값이 나타나면 값을 말소시키며 1의 값이 나타날 경우, 그림()에 명시된 tanh레이어를 거쳐 만들어진 벡터에 정보를 추가하여 값을 업데이트 한다. 이후 Cell State에서 정보가 합쳐지며 업데이트가 완료된다. Output gate는 Sigmoid Layer에 입력 정보를 보내 어느 부분을 출력할 것인지 결정하는 역할을 한다. 출력이 결정된 정보는 Cell State의 정보와 함께 tanh 함수를 거쳐 출력한다.

**2.4. 플러터(Flutter)**

어플리케이션 개발에 쓰인 프레임워크 플러터(Flutter)는 Google사에서 개발한 프로그래밍 언어 Dart를 기반으로 설계된 어플리케이션 개발 프레임워크이다. 기반이 되는 언어인 Dart는 JavaScript와 Java의 문법 부분에서 유사함을 나타내고 있으며, 코드 작성 시 위젯이라 불리는 클래스 및 함수 기반 모듈을 가져와 기존의 어플리케이션 개발 언어인 Kotlin, Swift에 비해 상대적으로 손쉽게 어플리케이션을 개발할 수 있다.   
개발 과정에서의 플러터의 강점은 안드로이드, iOS앱을 동시에 개발할 수 있는 크로스 플랫폼이기 때문에 효율적인 개발이 가능하다는 점이다. 개발된 앱은 어떤 운영체제에서도 똑같은 사용자 인터페이스와 사용자 경험을 제공하며 핫 리로드(hot reload) 기능을 통해 별도의 빌드 과정 없이 결과를 출력해내기 때문에 개발 과정에서 시간 절감 효과를 볼 수 있다.  
플러터 기반 어플리케이션은 위젯(Widget)을 통해 구성되며 역동적이면서 유연한 사용자 인터페이스를 만들어 나갈 수 있다. 디자인 측면에서 보았을 때도 마찬가지로 안드로이드 머티리얼(marterial) 디자인, iOS의 쿠퍼티노(cupertino) 디자인을 기본적으로 지원하기 때문에 플랫폼에 종속된 디자인에서 벗어나 다양한 디자인으로 어플리케이션을 개발할 수 있다.  
플러터에서 전체 화면을 그리는 과정은 스키아(skia) 엔진을 통해 작성된다. C++언어로 개발된 오픈소스 2D 그래픽 엔진으로 플러터를 비롯한 많은 플랫폼 및 제품에 사용되는 엔진이다. 해당 엔진의 그림 동작 방식이 두번의 작업을 한 번으로 압축시키는 방식이다 보니 프레임 속도 또한 60프레임 이상의 속도로 화면을 갱신한다. 따라서 많은 비교 대상으로 여겨지는 네이티브 앱에 비하여 속도 차이가 크게 떨어지지 않는다.

**3 장 환자 데이터 및 예측 모델**

**3.1 환자 데이터 생성 시뮬레이터**

어플리케이션에 전달하기 위한 혈당 데이터는 Python에서 제공하는 라이브러리 Simglucose를 통해 생성하였다. 본 논문에서 수행하는 Simglucose 라이브러리의 혈당 생성 과정은 그림<>과 같다.  


시뮬레이터를 사용한 혈당 생성은 그림<>의 가장 마지막 부분인 SimObj 클래스 객체로부터 생성이 되며, 그 이전의 모든 부분은 해당 클래스를 구성하는 요소이다.   
환자의 상태 구성을 결정하는 T1DPatient 클래스에서는 기본 변수로 환자의 상태를 설정하고 내부 함수로 환자의 상태를 업데이트 한다. 본 논문에서는 함수 부분의 withID를 통해 내장되어있는 환자의 상태를 가져와 사용하였다. 환자에게 투여되는 인슐린 펌프의 정보는 Insulin\_pump객체를 통해 가져왔으며, 혈당측정절차는 CGMSensor클래스 객체를 기반으로 CGM센서를 통해 혈당을 측정한 것을 가정할 수 있도록 했다. 최종적으로 환자의 식사량을 별도로 조정할 수 있도록 Scenario객체를 CustomScenario객체로 상속시켜 추가적으로 구성했으며 식사에 대한 시나리오는 다음과 같이 구성되었다.

* 아침 식사

- 시간: 07시

- 섭취 탄수화물양: 45g

* 점심 식사
  + 시간: 12시
  + 섭취 탄수화물양: 70g
* 간식
  + 시간: 16시
  + 섭취 탄수화물양: 15g
* 저녁 식사
  + 시간: 18시
  + 섭취 탄수화물양: 80g
* 야식
  + 시간: 23시
  + 섭취 탄수화물양: 10g

혈당을 계산하는 방식은 컨트롤러를 통한 계산방식을 사용한다. 본 논문에서는 기본값으로 쓰이는 BBController를 컨트롤러로서 사용했으며 BB(Basal-Bolus)Controller 콘트롤러는 제1형 당뇨병을 앓고 있는 사람이 어떻게 혈당 수치를 조절하는지 흉내낸다. UVa/Padova 시뮬레이터는 기본 인슐린산염 베이스, 보정 인자 CF 및 탄수화물 비율 CR과 같은 제1형 당뇨병 모델을 가진 각 가상 사람들에게 이 알고리즘에 필요한 매개 변수를 제공하며 시뮬레이터는 인슐린 주입의 양을 결정하도록 유도한다. BB컨트롤러의 혈당 계산식은 다음과 같다.

bas+(ct>0)·(ct/CR(bt>150)·(bt-btgt)/CF

ct는 t 단계에서 탄수화물, bt는 혈당 측정, btgt는 목표 혈당 수치이다. 마지막 항은 혈당 측정값이 150mg/dl을 초과하는 경우에만 적용된다. 이렇게 구성된 정보는 T1DSimEnv 클래스의 변수로 사용이 되며 이후 클래스 내부 함수인 step메서드에서 혈당 생성이 이루어진다.

**3.2. 모델 훈련 시나리오**

모델을 생성하기 위한 조건은 다음과 같다.

모델 훈련을 위해 준비한 데이터 셋은 청소년들의 하루 혈당치를 00시부터 익일 00시까지의 값을 5분 단위마다 측정한, 총 290개의 행으로 구성이 되어있는 csv파일이다. 해당 csv파일 jupyternotebook의 모델파일이 위치한 디렉토리의 하위 디렉토리에 위치한다. 모델 생성 과정에서 특성변수로 사용되는 체중은 simglucose 라이브러리 자체에 내장되어 있는 환자 특성이 정의된 csv파일에서 가져와 모델 파일 내부에 리스트 형식으로 정의하였다. 파일 내부에서는 30분 후의 혈당 값을 예측하기 위해 25분, 총 5개의 데이터를 하나의 데이터셋으로 묶어 tensorflow의 placeholder에 대입할 수 있도록 가공하였다.

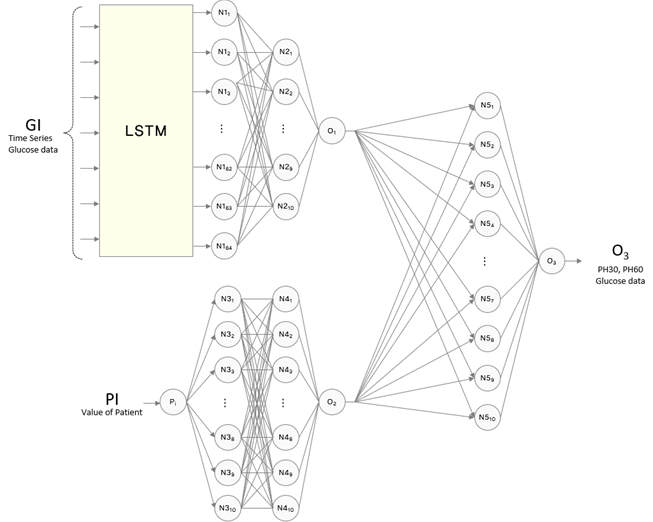
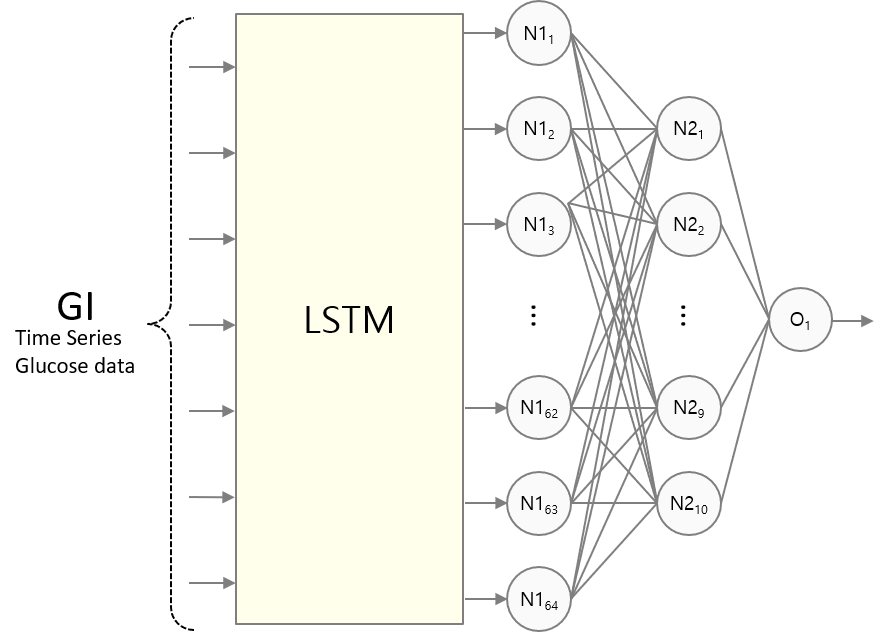
3.2.1. 특성 변수

특성 변수는 신경망 자체의 정확도를 증가시키기 위한 하나의 보정 값으로 여겨지며 본 논문에서는 환자의 체중을 이용해 정확도를 증가시킨다. 환자의 구성과 각각이 보유중인 체중에 대한 정보는 다음 [표]와 같다.

|  |  |
| --- | --- |
| 환자 명 | 체중 |
| adolescent#001 | 68.706 |
| adolescent#002 | 51.046 |
| adolescent#003 | 44.791 |
| adolescent#004 | 49.564 |
| adolescent#005 | 47.074 |
| adolescent#006 | 45.408 |
| adolescent#007 | 37.898 |
| adolescent#008 | 41.218 |
| adolescent#009 | 43.885 |
| adolescent#010 | 47.378 |

3.2.2. 계층 별 구성

전처리된 데이터를 바탕으로 학습하는 5개의 LSTM셀로 구성된 LSTM망과 환자의 특성변수 체중값을 입력 받는 완전 신경망을 구성한다. 완전신경망의 최종 형태는 다음 [그림]과 같으며 각각의 Ni는 은닉계층의 노드들을 의미하며 해당 노드들의 집합이 하나의 계층으로서 동작한다. 최종적인 출력 값은 PH = 6 \* 5, 30 분 뒤의 혈당 값을 반환하는 것으로 모델 훈련을 마친다. 신경망의 각각의 부분은 [그림]과 [그림]으로 나누어지며 각각 앞서 서술한 LSTM망과 환자의 특성 변수값을 입력으로 받는다.



[그림]에서 제시하는 GI는 연속적으로 측정된 5개의 5분단위 혈당 데이터를 입력으로 사용하며, 활성화 함수는 하이퍼볼릭 탄젠트함수를 거쳐 학습을 진행한다. 해당 셀을 거친 출력값은 완전 신경망의 연결 이후 벡터값으로 출력된다.