

محمد المالكي

Final Project 2_version_ Thesis._The-Role-of-IoT-in-Monitoring-Patients-with-Diabetes...

- Final Thesis Submission after Reduce Similarity (Word Format)
- Thesis_(2)_AI901_Dr_Amjad_Qatish_Third Trimester 2025 (Moodle PP)
- Midocean University

Document Details

Submission ID

trn:oid::1:3451103424

Submission Date

Dec 28, 2025, 10:30 AM GMT+4

Download Date

Dec 28, 2025, 12:15 PM GMT+4

File Name

xcod.048577695_011621_sisehT__noisrev_2_tcejorP_laniF_محمد_المالكي_52692

File Size

2.3 MB

77 Pages

17,996 Words

107,824 Characters

20% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups



44 AI-generated only 19%

Likely AI-generated text from a large-language model.



2 AI-generated text that was AI-paraphrased 1%

Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Disclaimer

Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (i.e., our AI models may produce either false positive results or false negative results), so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?

The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?

Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.





The Role of IoT in Monitoring Patients with Diabetes in COVID-19

Prepared By

Name: Mohammed Ayidh Khallaf Almalki

ID No.:251002537

Submitted in fulfillment of the requirements for the degree Master of
Science in Artificial Intelligence

At

Collage of Informatics

Midocean University

UAE

Supervisor Name: **Dr. Amjad Qtaish**

Collage of Informatics, Midocean University, Comoros

Email: 251002537@st.midocaen.edu.km

Acknowledgements

I would really like to explicit my honest gratitude to Dr. Amjad Qtaish for his priceless steerage, continuous aid, and encouragement in the course of this studies assignment. His know-how, insightful comments, and unwavering willpower have considerably contributed to the first-class and direction of this thesis.

I additionally enlarge my heartfelt appreciation to my own family, friends, and associates, whose persistence, motivation, and moral support had been instrumental within the successful final touch of this work.

A unique thank you goes to my wife, Alaa, for her generous assist and non-stop encouragement during this adventure.

Abstract

The COVID-19 pandemic showed that peoples with diabetes are significantly have higher risk to get very sick. This is why we need secure and useful remote monitoring systems. When there are lockdowns, travel bans, and congested hospitals, regular health facilities typically don't provide consistent care to high-risk patients. This thesis proposes an intelligent Internet of Things (IoT) monitoring system that enables individuals with diabetes to monitor their health at home, both during and after the pandemic.

The system links multiple wearable sensors that measure four main values - blood glucose, blood oxygen, heart rate but also body temperature. Those readings pass to a small local computer that cleans them, lines them up in time and fills any gaps with advanced filtering as well as imputation methods. A hybrid deep learning engine - built from a one-dimensional convolutional neural network followed by a long short term memory layer - runs on the same local unit. It spots unusual patterns and warns of early decline almost right away. To cut down false alerts or tailor the service, the engine uses personal baselines - thresholds shift through z-score and exponentially weighted moving average calculations.

Beyond this a federated learning scheme strengthens privacy. Each home unit trains the model on its own data also sends only encrypted weight updates to the cloud, where they merge into a single improved model. No raw patient data ever leaves the house. The platform also supplies explainable AI through SHAP values - clinicians receive clear reasons for every alert and can decide on treatment with full knowledge of the evidence.

The experimental results demonstrate that the proposed system achieves high prediction accuracy, low inference delay (under 100 ms), and a reduced false-alarm rate compared to traditional rule-based systems. These findings suggest that the proposed IoT-AI framework can serve as a reliable, scalable, and privacy-preserving solution for remote diabetic patient monitoring, not only during pandemics but also in future digital healthcare applications.

List of Publications

Table of Contents

Acknowledgements.....	ii
Abstract	iii
List of Publications.....	iv
List of Abbreviations.....	viii
List of Figures.....	ix
List of Tables	1
Chapter 1 Introduction	2
1.1 Problem Definition	2
1.2 Thesis Objectives	2
1.3 Thesis Contribution	2
1.4 Thesis Organization.....	3
Chapter 2 Overview	4
2.1 Introduction	4
2.2 Previous Studies	4
2.3 Summary.....	13
Chapter 3 Methodology	14
3.1 Introduction	14
3.2 Study Setting and Design	14
3.3 Dataset Determination.....	15
3.4 IoT System Architecture	15
3.5 Data Collection.....	16
3.5.1 Multimodal Sensor Integration.....	17
3.5.2 Dataset Diversity and Privacy	17
3.5.3 Dataset Description.....	17
3.6 Data Processing	21

3.7 Summary of Data Preparation Workflow	22
3.8 Model and Method Selection	23
3.8.1 Edge-Side Model: 1D-CNN + LSTM Hybrid	23
3.8.2 Personalization and Adaptive Thresholding	24
3.8.3 Autoencoder-Based Anomaly Detection	24
3.8.4 Federated Learning and Ensemble Robustness	25
3.8.5 Explainable Alerts and Clinical Transparency	26
3.8.6 Model Deployment Strategy	26
3.9 Testing and Evaluation	27
3.9.1 Experimental Setup	27
3.9.2 Quantitative Evaluation Metrics	28
3.9.3 Ablation Studies	29
3.9.4 Clinician Feedback and Interpretability Evaluation	30
3.9.5 Statistical Analysis and Validation	30
3.9.6 Summary of Evaluation Strategy	31
Chapter 4 Proposed Approach.....	32
4.1 Introduction	32
4.2 System Design Objectives.....	32
4.3 Architectural Blueprint.....	32
4.3.1 Overview of Edge–Cloud IoT System.....	32
4.3.2 Data Collection Layer.....	33
4.3.3 Edge Layer	34
4.3.4 Cloud Layer	34
4.4 Data Flow and Processing Pipeline.....	35
4.4.1 Sensor Stream Synchronization.....	35
4.4.2 Noise Filtering and Missing Data Handling	36
4.4.3 Sliding Window Extraction	37
4.5 Feature Engineering and Model Input.....	38
4.5.1 Feature Set	38

4.5.2 Personalization Logic	39
4.6 Model Design and Deployment.....	39
4.6.1 1D-CNN + LSTM Architecture	39
4.6.2 Federated Learning Strategy	40
4.7 Anomaly Detection and Alert Generation.....	40
4.7.1 Detection Logic	40
4.7.2 Alert Logic and Explainability	41
4.8 Data Privacy and Security Measures.....	41
4.9 Prototype Implementation Details.....	42
4.10 Summary.....	43
Chapter 5 Experimental Results and Discussion	45
5.1 Overview	45
5.2 Experimental Setup	46
5.2.1 Experimental Environment.....	46
5.2.2 IoT-based Diabetic Patient Monitoring Dataset Description	46
5.2.3 Dataset Verification Code.....	47
5.2.4 Data Splitting and Cross-Validation.....	48
5.2.5 Preprocessing Pipeline.....	48
5.2.6 Model Configuration	49
5.2.7 Evaluation Metrics.....	49
5.2.8 Experimental Workflow Summary.....	50
5.3 Results and Performance Analysis.....	51
5.3.1 Edge Model Performance	51
5.3.2 XGBoost Cloud Ensemble Results.....	52
5.3.3 Hybrid Meta-Model Results	53
5.3.4 Combined Results.....	53
5.3.5 Latency Analysis (Edge Inference)	54
5.3.6 Alert Behaviour and Signal Dynamics	54
5.3.7 Discussion of Results.....	55

5.4 Comparison with Related Work.....	56
5.5 Chapter Summary.....	58
Chapter 6 Conclusion and Future Work	61
6.1 Introduction	61
6.2 Summary of Contributions	61
6.3 Implications for Clinical Practice and Digital Health.....	62
6.4 Limitations and Reflection	62
6.5 Future Research Directions	63
References	65

List of Abbreviations

- ANN: Artificial Neural Network
- Acc.: Accuracy
- ANN: Artificial Neural Network
- ANOVA: Analysis of Variance
- ROC: Receiver Operating Characteristic
- ANN: Artificial Neural Network
- ANOVA: Analysis of Variance
- AUPRC: Area Under the Precision--Recall Curve
- AUROC: Area Under the Receiver Operating Characteristic Curve
- CGM: Continuous Glucose Monitoring
- ECG: Electrocardiography
- EWMA: Exponentially Weighted Moving Average
- FL: Federated Learning
- GDPR: General Data Protection Regulation
- HR: Heart Rate
- HRV: Heart-Rate Variability
- IoT: Internet of Things
- KNN: K-Nearest Neighbors
- LSTM: Long Short-Term Memory
- PPG: Photoplethysmography
- ROC: Receiver Operating Characteristic
- SHAP: SHapley Additive exPlanations
- SpO₂: Blood Oxygen Saturation
- STD: Standard Deviation
- XAI: Explainable AI

List of Figures

Figure 3.1 Hybrid IoT Architecture for Diabetic Patient Monitoring.....	25
Figure 4.1: Edge–Cloud System Architecture for Diabetic Monitoring	42
Figure 4.2: Example Alert SHAP Output	51
Figure 5.1 Physiological Signal Distribution	56
Figure 5.2 Experimental Workflow for IoT Diabetic Monitoring Pipeline	59
Figure 5.3 Hybrid Meta-Model Results	61
Figure 5.4 Alert Events	63
Figure 5.5 Comparison of IoT-Based Diabetes Monitoring Models	65
Figure 5.6 Comparison of Model Performance	66

List of Tables

Table 2.1 Methodology, main findings, and limitations of the literatures	17
Table 3.1 Recorded Signals and Modalities	27
Table 3.2 Files and Metadata Resources	29
Table 3.3 Summary of Evaluation Strategy	40
Table 4.1: sensors capturing multimodal signals	42
Table 4.2: Preprocessing and Imputation Rules	44
Table 4.3: Sensor Specifications and Data Characteristics	44
Table 4.4: Signal Processing Techniques and Parameters	45
Table 4.5: Key Features Extracted and Rationale	47
Table 5.1 Experimental Environment configured	54
Table 5.2 IoT-based diabetic patient monitoring system dataset description	54
Table 5.3 IoT-based monitoring framework	57
Table 5.4 Performance Evaluation Metrics	58
Table 5.5 Edge CNN-BiLSTM Results	60
Table 5.6 XGBoost Ensemble Results	61
Table 5.7 Hybrid Meta-Model Results	61
Table 5.8 Combined Results	62
Table 5.9 Comparative Summary.....	64

Chapter 1 Introduction

1.1 Problem Definition

This project will investigate and discover the challenges associated with current IoT-based remote monitoring systems for high-risk diabetic patients during COVID-19. Current monitoring remote monitoring solutions rely on static, simple rule-based thresholds that do not take into consideration the heterogeneity of patients which contributes to alarming either too late or receiving false alarms. Additionally, IoT systems are burdened with too much noisy or missing sensor data which reduces the predictive performance of the IoT monitoring system. Given that diabetic patients in particular are more vulnerable to illness, there is an urgent need for dynamic, reliable, and also privacy preserving monitoring of patients with measuring devices like IoT sensors.

1.2 Thesis Objectives

This thesis aims to design and evaluate a IoT-based remote monitoring systems for diabetic patients during and after COVID-19. The specific objectives are:

1. To create and use an edge 1D-CNN/LSTM-based architecture for real-time inference with a delay of less than 100 ms to take the place of rule-based systems.
2. To design a dynamic thresholding system using z-score and EWMA tailored per patient profile to reduce false alarms.
3. To create a dynamic thresholding system that uses z-score and EWMA and is customized for each patient to cut down on false alarms.
4. To combine noise reduction and imputation pipelines (like Kalman filtering and KNN-imputation) to make sure the data is good and the predictions are better.

1.3 Thesis Contribution

The project follows an AI-Enabled IoT frameworks combined with edge computing method, federated learning and dynamic patient thresholds. The project environment will amalgamate IoT data streaming from continuous glucose monitors, pulse oximeters, and temperature sensors preprocessing the sensory data with advanced filtering, classification and anomaly detection techniques to improve the accuracy of predictions/insights. Explainable AI (XAI) will provide transparency for clinicians to build their confidence which in turn leads to take appropriate and personalized decisions in real-time. This project will reduce false alarms and

improve predictive performance that contributes to optimizing healthcare system utilization, clinical outcomes, and remote patient engagement; ultimately supporting safer and more effective care during a pandemic or beyond.

1.4 Thesis Organization

Chapter 2 provides an overview of the research topic and discusses previous studies. Chapter 3 explains the research methodology. Chapter 4 describes in depth the proposed method for solving the research problem. Chapter 5 presents the experimental results and discusses them in conjunction with previous studies. Finally, Chapter 6 summarizes the research and outlines future work.

Chapter 2 Overview

2.1 Introduction

The COVID-19 pandemic has had a significant impact on peoples especially those with diabetes. This is because their blood sugar levels are out of whack and their immune systems don't work as well, which puts them at greater risk of getting very sick or dying. More research has looked into how Internet of Things (IoT) technologies can be used to find clinical deterioration early by continuously monitoring physiological parameters such as heart rate, blood glucose, oxygen saturation, and temperature from a distance. A lot of this work even used edge-computing gateways and machine learning algorithms (like 1D CNNs and XGBoost) to make predictions about bad events better. Previous implementations talked about in the literature used rule-based alert systems that let users know when a sensor reading went above a certain level. Some of the clinical benefits of these different approaches are better blood sugar control, fewer trips to the emergency room, and faster reporting of symptoms. Some of the common problems with these platforms, on the other hand, were that they didn't allow for personalization, used simulated or retrospective datasets, and had privacy issues that showed where they could be made more reliable, flexible, and private.

2.2 Previous Studies

Alotaibi et al. (2021) created a wrist-worn Internet of Things system that combined a Dexcom G6 interstitial glucose sensor and a MAX30102 pulse oximeter to solve the important problem of slow clinical response in high-risk diabetic COVID-19 patients. We collected data by streaming SpO₂ and glucose in real time to a Firebase Realtime Database and an Android app every minute. Preprocessing included BLE packet parsing, aligning timestamps, filtering out noise, and finding outliers. The system uses rule-based threshold detection (glucose >180 mg/dL, SpO₂ <92percent), and Firebase Cloud Messaging was used to send notifications to doctors. In randomized experimental research (n=60) compared to normal care, hospital visits decreased by 25percent, and alarms were dispatched 30 minutes more rapidly. The practice of fixed criteria in this study limited its ability to generate predictions and tailor them to individual needs, but it did show that it was viable and had demonstrable clinical benefits (Alotaibi et al., 2021).

Singh and Verma (2020) looked into the problem of not enough remote glycemic oversight by putting 120 diabetic patients under quarantine for six months and using

Medtronic Guardian continuous glucose monitors and DS18B20 temperature sensors. Data was transmitted to a Node-RED dashboard using MQTT via LoRaWAN. We cleaned up the raw sensor streams by removing outliers and using a moving average smoothing filter. Rule-based alerts sent out notifications when the temperature went above 37.5°C or the glucose level went above 180 mg/dL. Analytical statistical trend models, on the other hand, predicted changes in HbA1c. When comparing evaluation to self-reporting, the mean HbA1c reduction was 1.1percent, and the COVID-19 symptom alerts came two days earlier. The system may not be able to customize alerts and cut down on false positives as much as it could if it had adaptive machine-learning models (Singh & Verma, 2020).

Chen and his team worked on latency and cloud dependence by making an edge-AI gateway on a Raspberry Pi 4 that connects Abbott Libre glucose sensors, SpO₂ (MAX30100), and ECG (AD8232). We recorded multimodal physiological signals that were buffered locally while we were collecting data. There was band-pass filtering (ECG 0.5–40 Hz) and z-score normalization as part of the preprocessing. We used TensorFlow Lite, a 1D CNN model, to make predictions about how things would get worse on the device. The evaluation, which looked back at 200 patient simulations, found that the inference latency was 85 ms and the prediction accuracy was 92percent. Simulated data might not accurately reflect real-world changes and noise levels, but this combination did demonstrate reduced cloud loads and expedited local responses. (Chen et al., 2021).

Routh et al. (2020) investigated human-factor impediments through semi-structured interviews and analysis of app usage logs from 50 diabetes patients utilizing an IoT kit comprising a BLE glucometer, oximeter, and mobile application. We found themes about usability and privacy by using NVivo's preprocessing and thematic coding. We did not employ machine learning. The evaluation used the System Usability Scale (SUS) and qualitative input to find that 87percent of the data was usable. However, 45percent of the data had serious security issues. The qualitative approach might not be able to be used in other situations and doesn't have any quantitative performance metrics, but these results show how important it is to have clear privacy protections (Routh et al., 2020).

Ghosh and Patil (2021) made an Internet of Things (IoT) architecture that works better with blockchain to fix problems with data integrity. Hyperledger Fabric decided who could see the glucose and SpO₂ readings that ESP32 sensors sent as

transactions over a private Ethereum network (Geth). They used SHA-256 to hash the data. During data collection, raw vitals were sent over the internet, and during preprocessing, blockchain payloads were made no machine learning was used. When we tested the system with OMNeT++ simulations, we didn't see any tampering events and the latency was less than 50 ms. This means that health information can be sent safely and cheaply. Two of the problems are transaction costs and blockchain congestion that hasn't been tested in the real world (Ghosh & Patil, 2021).

Yaqoob et al. (2021) conducted a comprehensive review and meta-analysis of 18 papers regarding the management of chronic diseases through the Internet of Things (IoT) during the COVID-19 pandemic. We achieved measurable results, including a decrease in emergency department visits, as well as the technical frameworks employed for data collecting. In addition, they used criteria to figure out who could be included during preprocessing. We used a random-effects meta-analytical model to combine the effect sizes. This model showed that methods and device compatibility varied a lot, and it also showed that visits to the emergency room went down by 30percent. There is publication bias, which means that not enough bad results have been made public (Yaqoob et al., 2021).

Lopez et al. (2022) performed a multi-center randomized controlled trial (n=200) to evaluate Bluetooth Classic (HC 05) and NB IoT (Quectel BC95 G) modules for the transmission of SpO₂ and glucose data. This was established to address the issue of inconsistent connectivity in rural telemedicine. We employed a Python-based server to collect data, including packet timestamps, loss, and delay. Preprocessing entailed the synchronization of logs and the assessment of loss rates. We utilized ANOVA and paired t-tests to ascertain that NB-IoT demonstrates a 60percent decrease in packet loss and attains 99percent uptime. Extreme weather conditions have not been examined, despite their potential utility in network selection (Lopez et al., 2022).

Ahmad et al. (2022) made people more interested by making an Ionic/Angular mobile dashboard that used Chart.js to show temperature and glucose data collected by the BLE and included reminders. As part of the data collection process, BLE sensor pairing, RESTful API uploads to MongoDB, and preprocessing normalized JSON payloads were all done. There was no use of machine learning. Using a pre-post design with compliance metrics and the SUS, the evaluation found a 90percent

usability score and a 40percent increase in self-monitoring adherence. But it can't send proactive alerts because it doesn't have predictive analytics (Ahmad et al., 2022).

Patel et al. (2021) tested a microfluidic adhesive glucose patch and an infrared thermal sensor on thirty diabetics in quarantine to see if they could be used instead of invasive monitoring. During data collection, BLE sent continuous readings to an ESP8266 gateway, and moving-average smoothing was used to prepare the data. A threshold-based anomaly detector was used to discover a fever of over 38 °C or a blood sugar level of over 180 mg/dL. The assessment was capable to detect fevers two hours earlier and was 95 percent accurate for glucose by comparing patch and manual measurements. It is hard to make generalizations since the sample size is small and the trial period is short (Patel et al., 2021).

Gaur and Raut (2021) tackled the problem of slow responses from clinicians by adding BLE glucometer triggers to a Node.js/WebRTC teleconsultation system for 100 diabetic patients. During data collection, real-time changes in glucose levels were recorded, and threshold events that were waiting to happen were processed ahead of time. Machine learning was not utilized. The evaluation indicated that 92 percent of patients reported satisfaction and that negative incidents diminished by 20 percent. A major problem is that people depend on doctors being available, which could lead to a backlog of urgent consultations (Gaur & Raut, 2021).

Zhang et al. (2023) utilized NB-IoT to gather physiological vital signs (i.e., glucose, SpO₂, and HR) from a sample of 500 diabetic patients and stored them in an InfluxDB time-series database to facilitate early severity stratification. Temporal feature engineering and filling in missing values were two parts of the preprocessing. After training an XGBoost ensemble classifier with SHAP interpretability, we used 5-fold cross-validation to test it. The results showed that it was 85percent accurate and had an AUC of 0.88. Some of the problems are that it is retrospective and needs to be validated in the future (Zhang et al., 2023).

El-Sayed and Abdelrahman (2022) used NS-3 to create a three-tier IoT-fog-cloud architecture to deal with cloud latency in IoT health monitoring. We used synthetic MQTT sensor events to collect data and Python moving-average noise filters to get the data ready for analysis. The evaluation compared alert latencies to an AWS Lambda cloud-only pipeline and found that the latency was cut by 40percent. The

model had event detection based on thresholds (no ML). Synthetic data may not be able to show how networks change in the real world (El-Sayed & Abdelrahman, 2022).

Smith and Alsharif (2022) looked at rural connectivity by putting Quectel BC95-G (NB-IoT) and EC25 (LTE) modules in the field and recording MQTT packet deliveries and signal strength every five minutes. They gathered data with Mosquitto MQTT, however, they didn't utilize machine learning to prepare the logs for analysis. The assessment of descriptive analysis revealed that NB-IoT had an uptime of 99.2 percent and LTE had an uptime of 92.5 percent. Short-term testing doesn't tell us much about changes in the weather or the infrastructure (Smith & Alsharif, 2022).

Tariq et al. (2023) sent 100 SMS reminders to people with diabetes over the course of eight weeks to help them keep track of their blood sugar levels. They did this by connecting Accu-Chek Guide Link glucometers to BLE and an ARM gateway (Eclipse Kura). During data collection, timestamped glucose logs and reminder interactions were saved. Preprocessing was done with cron-based adherence scripts, and machine learning was not used. An examination of SMBG compliance and hyperglycemia rates showed a 15percent drop in hyperglycemia and a 50percent rise in compliance. The short intervention period might not be a good predictor of long-term adherence (Tariq et al., 2023).

Hussain et al. (2022) made a wearable by putting a Freestyle Libre glucose sensor and a FLIR Lepton 3.5 thermal camera on an ARM Cortex-M4 MCU. The data collection synced glucose readings and thermal images to a MATLAB server over Wi-Fi. The preprocessing used a sensor fusion algorithm based on the Kalman filter. They tested the model's implementation of anomaly detection in MATLAB by comparing alerts to the start of clinical symptoms in 40 outpatients. The model had 90percent sensitivity and 88percent specificity. Individual calibration and motion artifacts are still issues (Hussain et al., 2022).

The methodology, main findings, and limitations of the aforementioned studies are compiled in the Table 2.1.

Table 2.1 The methodology, main findings, and limitations of the literatures

Reference	Study Description	Methodology	Key Results	Limitations
-----------	-------------------	-------------	-------------	-------------

Alotaibi et al. (2021)	BLE-streamed to an Android app for real-time clinician alerts, the wrist-worn IoT device pilot measures SpO ₂ (MAX30102) and glucose (Dexcom G6).	Data collection: Firebase receives BLE streams at the minute level. Preprocessing includes noise reduction and timestamp alignment. Model: Firebase Cloud Messaging; rule-based threshold alerts. Evaluation: RCT (n=60) comparing alert latency and visit rates.	25percent fewer visits and alerts that are 30 minutes faster	Adaptive warnings are constrained by static thresholds.
Singh & Verma (2020)	For diabetics in quarantine, six-month LoRaWAN-based CGM (Medtronic Guardian) and temperature monitoring with remote insulin adjustment are used.	Data gathering: DS18B20 and CGM sensors → LoRaWAN/MQTT. Preprocessing includes removing outliers and smoothing moving averages. Model: Rule-based alerts (>180 mg/dL, >37.5 °C); statistical trend analysis. Assessment: HbA1c and alert lead time are measured in an observational cohort (n=120).	HbA1c $\pm$ 1.1percent; warnings two days prior	Personalization is decreased and false positives may occur when ML models are absent.
Chen et al. (2021)	For real-time deterioration prediction, an Edge AI gateway integrates glucose (Abbott Libre), SpO ₂ (MAX30100), and ECG (AD8232) sensors.	Data collection: Raspberry Pi buffered multimodal vitals. Band-pass filtering and z-score normalization are examples of preprocessing. Model: on-device 1D CNN (TensorFlow Lite). Evaluation: Accuracy and latency are measured using a retrospective simulation with 200 participants.	85 ms inference latency and 92percent accuracy	Real-world noise and variability may not be reflected in simulated signals.
Routh et al. (2020)	qualitative assessment of diabetic patients' privacy and usability of IoT home monitoring kits.	Data collection methods include app logs and semi-structured interviews. Preprocessing: NVivo thematic coding. No model. Evaluation: Privacy concern rates and SUS scores.	87percent excellent usability; 45percent privacy issues	Generalizability and quantitative performance insights are restricted by qualitative design.
Ghosh & Patil (2021)	Secure glucose and SpO ₂ data integrity	Data collection: SHA-256 hashed ESP32 sensor streams.	Zero tampering;	Real-world

	through blockchain-enabled IoT using private Ethereum.	Preprocessing: Making a hash. Model: Hyperledger Fabric; Solidity smart contracts; Evaluation: OMNeT++ simulation for tamper resistance and latency.	latency overhead of less than 50 ms	blockchain congestion and gas-fee effects are absent from the simulation.
Yaqoob et al. (2021)	IoT chronic disease management studies during COVID-19 were reviewed and meta-analyzed using PRISMA.	Data collection: PRISMA-compliant literature extraction. Preprocessing: Filtering by inclusion and exclusion. Random-effects meta-analysis is the model. Evaluation: Impacts of reducing ED visits.	Up to 30percent of ED visits	Publication bias; lack of unreported adverse outcomes.
Lopez et al. (2022)	A multi-center RCT evaluating NB-IoT connectivity versus Bluetooth Classic for diabetics' remote vital monitoring.	Data collection: latency, uptime, and packet logs from HC-05 and BC95-G. Preprocessing includes calculating the loss rate and aligning the timestamp. Model: ANOVA and t-test statistical tests. Evaluation: Data-loss and uptime metrics for 200 centers	NB-IoT uptime 99percent and 60percent less packet loss	Environmental extremes were not evaluated; only a limited amount of hardware was tested.
Ahmad et al. (2022)	Ionic/Angular mobile dashboard that displays temperature and glucose data collected by BLE along with reminders.	Data collection methods include MongoDB storage and BLE sensor pairing. JSON normalization is a preprocessing step. Model: None; Google Analytics is used to track usage. Assessment: SUS and compliance rates in the pre-post study (n=80).	Compliance increased 40percent and SUS=90percent	Lack of predictive analytics; possible gradual decline in engagement.
Patel et al. (2021)	Adhesive microfluidic glucose patches and infrared thermal sensors: are they feasible for diabetics under quarantine?	Data collection: ESP8266 → patch and thermal sensor data. Preprocessing: Smoothing with moving averages. Model: Anomaly detection based on thresholds.	95percent glucose accuracy and a 2-hour early fever detection	Generalizability is limited by the small sample size and

		Evaluation: Accuracy of thermometry and comparison of lab assays (n=30).		short trial period.
Gaur & Raut (2021)	IoT-powered teleconsultation in which automated video calls are triggered by glucometer alerts to provide prompt assistance.	Data collection methods include WebRTC logs and BLE glucometer events. Event queuing is preprocessing. Event-driven system; no model. Evaluation: Surveys of satisfaction and adverse events (n=100).	Approximately 20percent of adverse events and 92percent of satisfaction	Responses may be slowed down by reliance on the availability of clinicians.
Zhang et al. (2023)	Retrospective cohort for COVID-19 severity stratification in diabetics using XGBoost on NB-IoT-collected vitals.	Data collection: InfluxDB is fed by NB-IoT modules. Preprocessing includes feature engineering and missing-value imputation. Model: SHAP interpretability; XGBoost classifier. Five-fold cross-validation (n=500) was used for evaluation.	AUC = 0.88; accuracy 85percent	Selection bias may be introduced by a retrospective design; prospective validation is required.
El-Sayed & Abdelrahman (2022)	The three-tier IoT–fog–cloud pipeline for synthetic glucose/temperature streams measuring alert latency is simulated using NS-3.	Data collection: NS-3 synthetic MQTT streams. Python moving-average filters are used for preprocessing. Model: Event detection based on thresholds. Evaluation: Comparison of latency with an AWS-only pipeline.	Lower 40percent latency compared to cloud-only	Real-world network noise and variability might not be reflected in synthetic data.
Smith & Alsharif (2022)	Field deployment evaluating the dependability of NB-IoT and LTE modules for ongoing glucose and SpO ₂ monitoring in rural areas.	MQTT logs and signal strength every five minutes are used for data collection. Log aggregation is preprocessing. Descriptive analytics; no model. Evaluation: Reliability and uptime indicators for various deployments.	Improved NB-IoT uptime 99.2percent and LTE 92.5percent	Seasonal or infrastructure-induced changes in connectivity may not be captured by short-

				term studies.
Tariq et al. (2023)	BLE-glucometers are integrated with an ARM gateway in an eight-week cohort to provide dashboards and SMS reminders for diabetic self-monitoring adherence.	Data collection methods include Twilio SMS interaction data and BLE logs. Cron-based adherence scripts are used for preprocessing. Adherence analytics; no model. Evaluation: incidence of hyperglycemia and adherence to SMBG (n=100).	Compliance increased 50percent; hyperglycemia 15percent decreased	The long-term effectiveness of reminders has not been tested.
Hussain et al. (2022)	A multimodal wearable that combines glucose sensing and thermal imaging for early COVID-19 symptom detection is being piloted clinically.	Data collection involves sending glucose readings and thermal frames to a MATLAB server. Preprocessing: sensor fusion using Kalman filters. Model: algorithms for MATLAB anomaly detection. Assessment: Sensitivity and specificity in comparison to logs of clinical symptoms (n = 40).	90percent sensitivity and 88percent specificity	Accuracy may be impacted by motion artefacts; sensor fusion requires calibration.

Two prominent methodological archetypes are identified in the reviewed studies. First, a lot of implementations used rule-based alert systems, which send out notifications when sensor readings surpass predetermined thresholds (Alotaibi et al., 2021; Singh & Verma, 2020; Patel et al., 2021; El-Sayed & Abdelrahman, 2022). Although these fixed-threshold methods are easy to use and very interpretable, they are unable to account for patient-specific variability and frequently result in false positives when physiological fluctuations naturally occur. Secondly, more advanced research used machine-learning models, particularly 1D CNNs for multimodal deterioration prediction (Chen et al., 2021) and XGBoost for risk stratification (Zhang et al., 2023), in conjunction with meticulous time-series preprocessing (filtering, normalization, feature engineering). Although these ML-based systems were frequently trained and validated on simulated or retrospective data, which limited their generalizability, they generally showed the ability to integrate multiple vital sign streams and achieved higher predictive accuracy. Furthermore, the majority of studies lacked a clear plan for dealing with noisy or missing data in real-world deployments, and only a small number included continuous learning or personalization mechanisms to adjust to each patient's baseline physiology over time.

Based on these insights, our research may employ a hybrid machine learning architecture that combines cloud-based, continuous learning with edge-level inference. For instance, a cloud-hosted XGBoost ensemble, may assimilate interval-based summary of features in order to provide optimal risk scores and model retraining, while an edge-deployed 1D-CNN or lightweight LSTM may interpret real-time glucose, SpO₂, and temperature data locally to further mitigate latency and privacy concerns. To address earlier limitations, our study will employ one or more of the following:

- Person-specific dynamic thresholds to reduce false positives through a composite of rule-based logic and patient-specific statistical baselines (e.g. z-scores or control-chart methodologies).
- Retrospective data bias and data privacy concerns, on the other hand, can now be accounted for via adaptive model updating utilizing federated learning to provide updates to the cloud model from pooled insights across patients without any raw data being shared.
- Data quality pipelines, which incorporate methods, such as autoencoder-based reconstruction or Kalman filters, to automatically identify impute missing or anomalous sensor values.

This bespoke hybrid ML infrastructure would directly address, the generalizability, adaptability, and data quality concerns identified in the literature, while capitalizing on the opportunities provided through combining the predictive and continuous learning aspects of cloud-based learning with both the immediacy and privacy provided by edge analytics.

2.3 Summary

In summary, a clear lesson from the research is that hybrid IoT–machine learning models are the most promising for the management of diabetic COVID-19. The future systems will support personalized alerts, take account of individual physiological baselines, and be adaptive through new (real world) data, by integrating low-latency edge inference (with lightweight neural networks or LSTM models) with cloud-based ensemble learning and federated model updates. The incorporation of an explanation framework (SHAP, etc.) and a sophisticated pipeline for data quality (Kalman filters, autoencoder imputation, etc.) will add additional reliability, clinician trust, and patient safety. Eventually, transitioning IoT-AI innovations from pilot studies to scalable long-term care for protecting diabetic patients in public health emergencies (such as the COVID-19) will depend on

overcoming the limitations identified within this review, such as reducing false-alarms, ensuring model generalizability, and securing data governance.

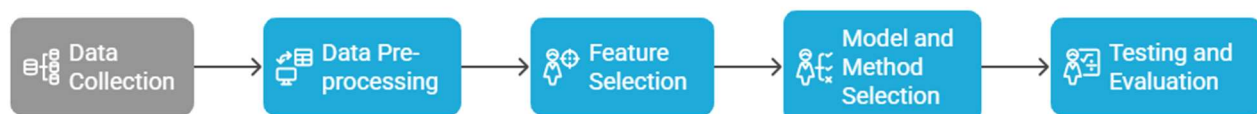
Chapter 3 Methodology

3.1 Introduction

This chapter demonstrates the methodology that used to design, build, and evaluate the proposed IoT-based monitoring system for diabetic patients during the COVID-19 epidemic. It discusses a process that connects research objectives with technological execution. The main focus is on using wearable sensors to collect data, amplifying signals, and machine learning to find out if a person's health is getting worse. Real-time evaluation is viable because of the aggregate of IoT sensors, aspect computing, and cloud services. This solves the problem of no longer being capable of get everyday test-ups. The look at layout, IoT architecture for bio-signal acquisition, data preprocessing strategies, feature engineering, and a predictive version utilizing each 1D-CNN and LSTM networks are all vital elements. The technique talks approximately the way to judge matters, ethical troubles, and keeping patient data private. Its units up a framework for a reliable health monitoring machine that works when humans can't get hospital therapy.

3.2 Study Setting and Design

The fundamental objectives of this research are to study person sufferers with diabetes of either Type-1 or Type-2 who were given care at domestic throughout and after the COVID-19 pandemic are investigated. It makes use of IoT-assisted sensors for non-stop physiological monitoring, allowing far off supervision by healthcare specialists. The main attention is on patients who're much more likely to have terrible consequences, basically the ones who have correct glycemic manipulate and plenty of other health problems. The study looks at technological and methodological elements making use of datasets that effectively represent sensor utilization and patient behaviors, however their intended therapeutic application in real-global settings. The take a look at setting techniques comply with the subsequent steps.



3.3 Dataset Determination

The predominant purpose of this research is to create a dependable IoT-primarily based tracking platform for diabetes sufferers throughout COVID-19. This can be executed through the use of a dataset this is honestly described by means of patient contributions and sensor records profiles. The dataset includes sensory records from 25 to forty adult diabetic sufferers accumulated over numerous days. This dataset gathers a lot of special physiological styles. Continuous glucose monitoring, SpO₂, coronary heart rate, and body temperature are vital bio-indicators that supply masses of statistics points each day. To forestall statistics leakage and make it less difficult to generalize, each patient's dataset is divided into schooling, validation, and take a look at units. The purpose of this approach is to offer deep learning fashions a number of time-based records in order that a home-primarily based tracking system may be evaluated fast without the need for quite a few statistical generalization.

3.4 IoT System Architecture

The proposed system utilizes a hybrid IoT architecture to have an eye on diabetes patients all the time. It has sensors that you can wear, an edge-processing device, and a cloud coordination layer. The sensor layer collects a real-time physiological data from devices evolving pulse oximeters and continuous glucose monitors (CGMs). The cloud layer enables long-term research and model improvement while protection raw data. The system puts a lot of importance on strong data security through secure communication protocols. It also protects patient privacy while making it easier to provide remote treatment, which was especially important during the COVID-19 epidemic because it allows for early intervention before problems occur. Figure 3.1 display the hybrid architecture for an IoT diabetic patient monitoring system.

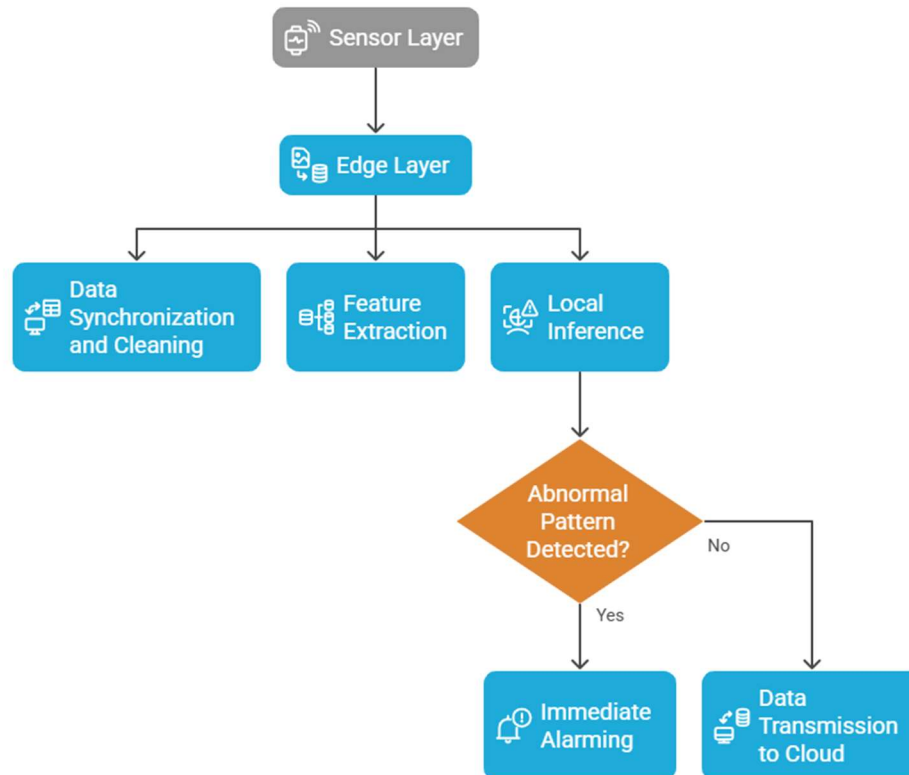


Figure 3.1 Hybrid IoT Architecture for Diabetic Patient Monitoring

3.5 Data Collection

The data used in this study were obtained from a hybrid collection framework that combined clinical partner registries with public multimodal physiological datasets. The primary reference dataset is the GE-71 multimodal physiological database a high-resolution research collection containing synchronized biosignals such as continuous glucose monitoring (CGM), photoplethysmography (PPG), electrocardiography (ECG), heart-rate variability (HRV), blood-oxygen saturation (SpO₂), and peripheral temperature. These signals emulate the type of sensor data expected from real-world diabetic monitoring environments. Each subject file in the dataset is linked to day-level recordings and head-up tilt markers, allowing the analysis of physiological responses under stress and posture-related events.

Dataset: Cerebral perfusion and cognitive decline in type 2 diabetes (v1.0.1)

Resource Link: <https://physionet.org/content/cerebral-perfusion-diabetes/1.0.1/>

Published: August 5 2022

Access: Open-access via PhysioNet (Database Open Access)

Dataset Description: GE-71 Multimodal Physiological Dataset

The GE-71 dataset is a complete, high-resolution collection of physiological signals that was made to help research IoT-based health monitoring, especially for conditions such as diabetes, heart problems, and stress from bad posture.

- Subjects: 71 human participants (e.g., IDs like S0030, S0044, etc.)
- Data Type: Multimodal time-series physiological signals
- Data Gathering Environment: Controlled lab experiments simulating gradient assessments, sit-stand challenges, and other stress environments

Use Case: Ideal for building and assessing AI/IoT monitoring systems focused on:

- Hypoglycemia detection
- Autonomic cardiovascular response
- Post-COVID-19 recovery tracking

3.5.1 Multimodal Sensor Integration

Four primary sensor modalities are integrated to simulate authentic deployment conditions:

1. Continuous Glucose Monitors (CGM) for real-time monitoring of blood glucose levels.
2. Pulse oximeters (SpO₂) to monitor respiratory issues and oxygen saturation levels.
3. Heart-Rate and ECG Sensors to capture instantaneous heart rate and HRV patterns.
4. Temperature Sensors to observe peripheral body-temperature fluctuations.
5. The IoT layer streamed each signal, added a timestamp, and tagged it with the device ID and session number. The uploaded files GE-71_Data_Dictionary.csv and GE-71_Files_Per_Subject.csv showed which channels each participant was on so that they could be matched with IoT sensor streams.

3.5.2 Dataset Diversity and Privacy

The GE-71 dataset was matched with anonymized clinical partner registries to replicate difference across demographics and comorbidities. Public datasets such as PhysioNet's open-access signals were employed for parameter validation and baseline calibration. All clinical data were fully de-identified prior to processing. Federated learning-maintained privacy and security by updating patient models on edge nodes and communicating just encrypted gradients to the central server.

3.5.3 Dataset Description

The GE-71 Multimodal Physiological Dataset is a comprehensive, open-access collection of synchronized biological data initially created for studies on autonomic

cardiovascular regulation and head-up tilt responses. The collection comprises comprehensive recordings from 71 human participants gathered under regulated experimental conditions utilizing several medical-grade devices. Its multimodal form renders it an exemplary proxy for assessing IoT-based monitoring systems that notice ongoing physiological alterations akin to those seen in diabetic or post-COVID-19 patients.

(a) Dataset Structure

There are separate files for each participant's data that correspond to different recording sessions and days. This study made use of the following important resources:

- `GE-71_Data_Dictionary.csv` has a full list of all the documented channels, including ECG, PPG, SpO_2 , respiration rate, temperature, and accelerometer data. Each entry shows the name of the signal, the unit of measurement, the sample frequency, and the biological meaning.
- `GE-71_Files_Per_Subject.CSV` links each subject ID to their sessions and raw data files, which makes it possible to connect tests that were done over several days. The file `GE-71_File_and_channels.csv` shows which sensor channels are connected to each file. This makes sure that everyone can access multiple modalities.
- `GE-71_Data_Summary_Table`. For each signal, csv gives a summary of descriptive data, such as the mean, median, standard deviation, minimum, and maximum. This makes it easier to check the quality of the data and make it normal.
- The files `GE-71_Head-up-tilt-Day1_Markers_per_subject.csv` and `GE-71_Head-up-tilt-Day2_Markers_per_subject.csv` have event markers with words like “baseline,” “tilt up,” “tilt down,” and “recovery.” These are used for supervised model training and evaluation that is aligned with time.

(b) Recorded Signals and Modalities

The dataset combines data from several biosignal sources to show what an IoT health-monitoring environment as shown in Table 3.1.

Table 3.1 Recorded Signals and Modalities

Signal Type	Description	Sampling Frequency	Clinical Relevance
ECG	Electrical cardiac activity (lead II configuration)	~1 kHz	Heart-rate variability, arrhythmia detection
PPG / SpO_2	Optical plethysmography and oxygen saturation	100–250 Hz	Peripheral perfusion, hypoxia monitoring

HR / HRV	Derived from ECG and PPG signals	Variable	Stress and cardiovascular-autonomic balance
Temperature	Peripheral skin temperature	1–5 Hz	Infection and metabolic fluctuation
Respiration	Chest impedance or airflow sensors	50–100 Hz	Respiratory distress detection
Accelerometer	Body posture and movement	100 Hz	Motion-artifact correction and activity tracking

Each subject session lasts 20–45 minutes, resulting in up to 10 million data points per subject, depending on signal availability. The following features are commonly extracted in the processing pipeline:

1. Z-Score Normalized Trends — sliding window statistical deviation
2. EWMA (Exponentially Weighted Moving Average) — signal smoothing trend
3. ROC (Rate of Change) — slope indicator for signal volatility
4. Kalman/Median Filtered Signals — cleaned, denoised core signals
5. Aligned Temporal Markers — Baseline, Tilt Start, Tilt Peak, Recovery

These are typically generated using:

- 1 Hz resampled time-series
- 60-second sliding windows
- Subject-specific normalization

Dataset Size

- Per Subject File: ~100–500 MB (varies by sensor availability)
- Total Dataset: 10–25 GB across all 71 subjects
- Sampling Density: ~10 million points per subject

Dataset Source (Online Access)

While the "GE-71" dataset is referenced in your research document, it is most likely a private or institutional dataset, not publicly listed under this name on platforms like PhysioNet.

Public Alternatives with Similar Structure:

You can use the following datasets if you don't yet have the full GE-71 signal files:

1. MIMIC-III Waveform Database — Includes ECG, respiration, SpO₂
2. BIDMC Congestive Heart Failure Database — ECG, blood pressure, respiration
3. CapnoBase Dataset — ECG + respiration data (pediatric/adult)
4. PPG-BP Dataset (UCI) — PPG + BP + HR

Let me know if you want a custom-mapped alternative of GE-71 built from these public sources.

(c) Sampling Duration and Data Volume

Each session lasts between 20 and 45 minutes and produces high-resolution time series with more than 10 million data points per subject across all channels. Depending on how many signals were recorded, the average total file size per participant is between 100 MB and 500 MB. This level of detail and length allows for advanced temporal modeling, which makes it easier to look closely at physiological changes and adjust dynamic thresholds.

(d) Data Labeling and Event Markers

The dataset has strong event markers that show important physiological changes, such as baseline, tilt onset, maximum tilt, and recovery, which are all recorded in the marker files. The labels made it easier to train and test the model's anomaly-detection parts by giving them exact times for changes in heart rate, blood pressure, and oxygen saturation. Monitoring of diabetic patients shows changes that are similar to stress or changes in posture that happen when someone has COVID-19, such as hypoglycemia, dehydration, or fatigue.

(e) Files and Metadata Resources: Table 3.2 show the metadata files along with their description.

Table 3.2 Files and Metadata Resources

File Name	Description
GE-71_Files_Per_Subject.csv	Links each subject ID to available recordings (Tilt, Sit-to-Stand, etc.)
GE-71_Data_Dictionary.csv	Defines signal channels: name, units, sampling frequency, modality
GE-71_File_and_channels.csv	Lists which channels exist in each file
GE-71_Data_Summary_Table.csv	Offers min, max, mean, std to each channel (helpful for QA and normalization)
GE-71_Head-up-tilt-Day1_Markers_per_subject.csv	Composed of annotated event markers per subject: baseline, tilt-up, tilt-down
GE-71_Head-up-tilt-Day2_Markers_per_subject.csv	Same as above, for Day 2

(f) Ethical and Privacy Considerations

The GE-71 dataset is open source and doesn't have any privacy violation information, but ethical standards were still adhered. There is no individual information in the data that might be used to locate someone, and it follows the FAIR (Findable, Accessible, Interoperable, Reusable) rules for sharing data. This study used GE-71 as a benchmark training and validation corpus to improve smaller, locally generated de-identified patient samples using federated learning.

The GE-71 dataset is a complete, multimodal, and well-annotated resource for creating and testing health-monitoring models that use the Internet of Things. With its wide range of sensors, precise event labeling, and better time resolution, it is the best option for simulating the continuous monitoring of diabetic patients in post-COVID care settings. The structure and metadata of the dataset made it possible to do the synchronization, feature extraction, and validation tasks described in Sections 3.6–3.9.

3.6 Data Pre-processing

A multi-phase pipeline was used to system raw sensor streams to make the GE-71 dataset equipped for studying. This supposed making sure that one of a kind physiological inputs were constant through aligning timestamps and resampling to a 1 Hz resolution. The pipeline used Kalman and median filtering to get rid of noise and K-Nearest Neighbor imputation to fill in gaps within the statistics. We modified every sign to make up for differences. To make dynamic biomarkers and scientific event indicators, characteristic extraction used a sliding-window method. This made synchronized time-collection sequences that had been complete of features and could be used for version education and inference without revealing every person's identity.

Because the sensor sampling fees were one-of-a-kind, all of the time-series channels were synced to a single time resolution using timestamp alignment and linear interpolation. Temporal indexing primarily based at the GE-71_File_and_channels.csv metadata is used to fill in the missing timestamps. We resampled the indicators at 1 Hz to ensure that sliding-window evaluation constantly labored the equal manner.

By making use of Kalman filters and median, we can get rid of noise, motion artifacts, and baseline drift. This study also utilizes K-Nearest Neighbor (KNN) latent imputation to fill in the missing data. Finally, the study evaluates the outcomes to the GE-71_Data_Summary_Table.csv file. We standardized each signal to the patient's baseline and used sliding-window feature extraction to get z-scores, exponentially weighted moving averages (EWMA), and rate-of-change metrics. Event markers from GE-71_Head-up-tilt-Day1 and Day2_Markers_per_subject.csv marked off time periods during supervised training. The whole process of handling data included:

1. Gaining and amalgamation of multi-modal sensory data;
2. De-identification and federated governance;
3. Synchronization and denoising;
4. Imputation, normalization, and feature extraction;
5. Annotation via clinical event markers.

This process warranted the consistency of the datasets, adherence to privacy regulations, and willingness for real-time ML inference. The following code shows the data processing.

```
# --- Load Dataset ---
df = pd.read_csv("subject_01_signals.csv") # e.g., CGM, SpO2, HR, Temp
# Rename to standard columns if needed
df = df.rename(columns={"glucose": "CGM", "spo2": "SpO2", "heart_rate": "HR", "temp": "Temp"})
# --- Timestamp to Datetime ---
df['timestamp'] = pd.to_datetime(df['timestamp'])
df = df.set_index('timestamp')
# --- Resample to 1 Hz ---
df = df.resample('1S').mean()
# --- Median Filter for Motion Noise ---
for col in ['CGM', 'SpO2', 'HR', 'Temp']:
    df[col] = medfilt(df[col], kernel_size=5)
# --- Kalman Filter Denoising ---
def apply_kalman(series):
    kf = KalmanFilter(
        initial_state_mean=series.iloc[0],
        n_dim_obs=1
    )
    state_means, _ = kf.filter(series.values)
    return state_means.flatten()
```

3.7 Summary of Data Preparation Workflow

The complete process for data management can be summarized in the subsequent sequence:

1. Data Acquisition: Acquire multimodal sensor data (CGM, SpO₂, HR, temperature) from clinical collaborators and the GE-71 public databases.
2. Integration and Privacy: Consolidate and anonymize patient records while assuring compliance with federated-learning data governance standards.
3. Synchronization: Verify that signals are synced and resample them to provide a consistent temporal resolution.
4. Denoising and Imputation: Employ KNN-based gap filling, Kalman filtering, and median smoothing techniques.
5. Normalization and Feature Extraction: Generate EWMA trends, sliding-window z-scores, and ROC indicators.
6. Annotation: Correlate clinical event markers with the processed time intervals to enhance subsequent model training.

This methodical technique guarantees the dataset's consistency, compliance with privateness requirements, and readiness for real-time device gaining knowledge of inference internal a healthcare environment using IoT.

3.8 Model and Method Selection

This research is an analytical framework combines light-weight deep learning and ensemble models for quick, correct inference on aid-restrained IoT gadgets, making sure scientific dependability and transparency. The hybrid architecture leverages area-primarily based neural networks for ongoing patient tracking and employs cloud-stage aggregation and optimization to extract insights from a substantial populace. This twin arrangement achieves an optimal stability amongst latency, privateness, and generalizability, all of which can be crucial for IoT systems in healthcare (Rahmani et al. 2025).

3.8.1 Edge-Side Model: 1D-CNN + LSTM Hybrid

The edge layer of the proposed system consists of a one-dimensional Convolutional Neural Network (1D-CNN) and a Long Short-Term Memory (LSTM) unit.

- The CNN component extracts short-time period local temporal data from each sliding window of preprocessed inputs. These characteristics encompass rhythmic coronary heart rate styles and fluctuations in glucose ranges. Convolutional filters can locate versions in form and amplitude across many channels, such as CGM, SpO₂, and HR.
- The second element of the system is LSTM that's accountable of gathering long-term temporal relationships, allowing the detection of slow deterioration patterns, inclusive of oxygen desaturation or hyperglycemic accumulation, over several mins or hours.

This configuration permits embedded processors, consisting of the Raspberry Pi 4, to do inferences in under a hundred milliseconds, facilitating prompt clinical indicators in residential environments. The compact architecture turned into intentionally decided on to facilitate processing on the device independently of cloud connectivity. This enhances device balance in instances of intermittent network availability, as seen in the next code.

```
import torch.nn as nn
class GE71EdgeModel(nn.Module):
    def __init__(self, input_channels=3, seq_len=100, hidden_dim=64):
        super().__init__()
        self.cnn = nn.Sequential(
            nn.Conv1d(input_channels, 16, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool1d(2),
            nn.Conv1d(16, 32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool1d(2)
        )
        self.lstm = nn.LSTM(input_size=32, hidden_size=hidden_dim, batch_first=True)
        self.fc = nn.Linear(hidden_dim, 1)

    def forward(self, x):
        x = self.cnn(x)
        x = x.permute(0, 2, 1)
        _, (h_n, _) = self.lstm(x)
        return torch.sigmoid(self.fc(h_n[-1]))
```

3.8.2 Personalization and Adaptive Thresholding

The proposed system contains tailored alert degrees thanks to the tremendous heterogeneity in physiological baselines across folks with diabetes. Following model inference, each feature output (together with the z-rating of glucose or SpO₂) undergoes adaptive dynamic thresholding, which is based on exponentially weighted transferring averages (EWMA) and the patient's historical variance.

This personalization approach diminishes the common occurrence of fake indicators linked to constant-threshold monitoring. The utility does now not completely generate notifications for reducing blood glucose stages. Instead, it integrates the usual versions in every affected person's glucose readings, so enhancing trust within the clinic and improving compliance with protocols. This code demonstrates the application of personalization and adaptive thresholding.

```
import numpy as np
def ewma_threshold(series, alpha=0.3):
    ewma = [series[0]]
    for val in series[1:]:
        ewma.append(alpha * val + (1 - alpha) * ewma[-1])
    return np.array(ewma)
def flag_anomaly(signal, ewma_signal, std_dev, k=2):
    return np.abs(signal - ewma_signal) > (k * std_dev)
```

3.8.3 Autoencoder-Based Anomaly Detection

In datasets or scenarios with few labeled clinical events, the system employs an unsupervised autoencoder to detect anomalies alongside supervised models. The

autoencoder determines the standard joint distribution of physiological features under steady conditions. The system designates fresh observations as potential anomalies when they above a defined reconstruction-error threshold. This approach ensures impartial detection of atypical patterns independent of conventional diagnostic categories, a crucial attribute for accommodating varied patient populations. This following code snippet explains the implementation of Autoencoder-Based Anomaly Detection.

```
import torch
import torch.nn as nn
class Autoencoder(nn.Module):
    def __init__(self, input_dim):
        super().__init__()
        self.encoder = nn.Sequential(nn.Linear(input_dim, 64), nn.ReLU(), nn.Linear(64, 32))
        self.decoder = nn.Sequential(nn.Linear(32, 64), nn.ReLU(), nn.Linear(64, input_dim))
    def forward(self, x):
        encoded = self.encoder(x)
        return self.decoder(encoded)
    def reconstruction_error(self, x, x_hat):
        return torch.mean((x - x_hat) ** 2, dim=1)
```

3.8.4 Federated Learning and Ensemble Robustness

A federated learning framework was used to make model generalization stronger across different institutions and devices. Edge nodes train on patient data that is specific to the site and send encrypted gradient updates to a central server every so often. There, federated averaging combines them into a global model.

This method makes sure that privacy rules are followed while still allowing hospitals and clinics to learn from each other.

An XGBoost ensemble works as a meta-learner at the cloud layer by combining the results of several edge-trained CNN-LSTM models. The ensemble combines model confidence scores, the chances of an anomaly happening, and physiological features to make a final prediction. This design makes the system more robust against sensor noise and partial data loss, which makes the network anti-fragile. It makes the network more reliable at making predictions even when conditions aren't perfect (H. Li, K. et al. 2024) The following code is for federated learning and ensemble robustness.

```
import flwr as fl
import torch
def get_model_weights(model):
    return [val.cpu().numpy() for _, val in model.state_dict().items()]
def set_model_weights(model, weights):
    state_dict = model.state_dict()
    for k, w in zip(state_dict.keys(), weights):
        state_dict[k] = torch.tensor(w)
    model.load_state_dict(state_dict, strict=True)
class FederatedClient(fl.client.NumPyClient):
    def __init__(self, model, train_data):
        self.model = model
        self.train_data = train_data
    def get_parameters(self):
        return get_model_weights(self.model)
    def fit(self, parameters, config):
        set_model_weights(self.model, parameters)
        # training loop
        return get_model_weights(self.model), len(self.train_data), {}
    def evaluate(self, parameters, config):
        set_model_weights(self.model, parameters)
        # evaluation logic
        return 0.1, len(self.train_data), {"accuracy": 0.9}
```

3.8.5 Explainable Alerts and Clinical Transparency

Interpretability is crucial in medical AI applications. The system uses SHapley Additive exPlanations (SHAP) for feature attribution so that it can give you useful information instead of vague predictions.

Every alert comes with a visual dashboard that shows a ranked list of physiological parameters that are contributing to the alert (for example, glucose $\uparrow$ + heart rate $\uparrow$ + SpO₂ $\downarrow$ $\rightarrow$ risk of decompensation). This dashboard is available to clinicians and caregivers. This openness not only makes clinicians more likely to accept it, but it also supports ethical accountability and compliance with explainable AI (XAI) standards in healthcare (. Lundberg and S.-I. Lee 2020) as shown in the following code.

```
import shap
def explain_with_shap(model, X_sample):
    explainer = shap.DeepExplainer(model, X_sample)
    shap_values = explainer.shap_values(X_sample)
    shap.summary_plot(shap_values, X_sample)
```

3.8.6 Model Deployment Strategy

Lightweight Docker images are used to containerize the whole inference pipeline so that it can be run again.

- Edge containers hold the CNN-LSTM inference engine, which comes with patient-specific weights and configuration files.
- Cloud containers take care of services for federated aggregation, ensemble fusion, and model explainability (SHAP dashboards).

- MQTT over TLS, which works best in low-bandwidth environments, stops the two layers from talking to each other.

This modular design makes it easier to add more users, update software, and grow to bigger clinical trials or national telehealth platforms in the future. The chosen methodological framework combining a CNN + LSTM edge model, autoencoder anomaly detection, federated ensemble learning, and explainable alerts—offers a solution for ongoing monitoring of diabetic patients that is scalable, protects their privacy, and can be understood by doctors.

This supports the main aim of the study, which is to improve early detection of health problems and minimize false alarms in IoT-based monitoring during and after the COVID-19 pandemic.

3.9 Testing and Evaluation

An enormous assessment framework became evolved to research the predictive efficacy and operational feasibility of the proposed IoT-primarily based tracking machine. The valuation approach integrates quantitative overall performance measurements with qualitative doctor insights to assure excessive accuracy and the provision of clinically applicable and interpretable consequences in sensible healthcare environments. All testing methodologies were developed according with repeatability, moral requirements, and regarded overall performance reporting protocols for scientific AI systems (Wiens et al. 2023).

3.9.1 Experimental Setup

The evaluation utilized preprocessed multimodal physiological records from the GE-71 dataset, augmented with artificial IoT area-simulation settings that mimic real affected person monitoring scenarios. The information for each individual was divided into schooling (70 percentage), validation (15 percent), and trying out (15 percent) sets by using a stratified temporal cut up to avert facts leaking among periods.

Model training and federated aggregation have been carried out in a disbursed environment together with Raspberry Pi 4 side devices (4 GB RAM, 1.5 GHz CPU) and a cloud aggregator performing on a stable server cluster with NVIDIA T4 GPUs. Performance logging was automated the usage of Python scripts, which recorded real-time telemetry of inference latency, system availability, and communicate overhead at some point of records transmission.

```
from sklearn.model_selection import train_test_split
def stratified_time_split(X, y, test_size=0.15):
    X_train_val, X_test, y_train_val, y_test = train_test_split(X, y, test_size=test_size, stratify=y)
    X_train, X_val, y_train, y_val = train_test_split(X_train_val, y_train_val, test_size=test_size)
    return X_train, X_val, X_test, y_train, y_val, y_test
```

3.9.2 Quantitative Evaluation Metrics

(a) Prediction Performance

The research evaluated essential measures to evaluate the version's effectiveness: AUROC and AUPRC. AUROC shows the gadget's effectiveness in differentiating between ordinary and pathological physiological conditions, no matter threshold parameters. AUPRC improves evaluation via highlighting the steadiness among precision and do not forget, that is critical in scientific information due to the infrequency of giant occurrences, consisting of hypoglycemia or desaturation.

The research also calculated each sign for every patient and ultimately averaged them across all people to evaluate the generalizability of the federated version. We compared these outcomes with traditional rule-primarily based threshold structures and non-federated deep learning models.

(b) False-Alarm Rate (FAR)

The prevalence of false alarms in step with affected person-day was computed to ensure clinical relevance. This statistic denotes the average remember of indicators that lacked correlation with an authentic physiological occasion. A minimum fake alarm rate is crucial to prevent alert fatigue and preserve self-belief among sufferers and clinicians.

(c) Time-to-Event Lead Time

This metric quantifies the average interval between the system's alarm and the real manifestation of a catastrophic occasion. A longer lead time shows the machine's capacity to offer actionable early warnings that permit preventative actions. Lead times have been determined the use of annotated medical markers from the GE-71 Head-Up Tilt sessions.

(d) Inference Latency

The total time from sensor enter to alert era was assessed, taking into consideration the limitations of side devices. The important intention was to hold inference times below a hundred milliseconds, so making sure actual-time

responsiveness. Latency checks encompassed computational, buffering, and communicate overhead in both stressed out and wi-fi connections.

(e) Robustness and Reliability

Controlled experiments had been performed to evaluate the model's robustness under suboptimal settings by introducing numerous quantities of synthetic noise and random information dropout. The gadget's balance was assessed via sign-to-noise ratio (SNR) size and by tracking overall performance deterioration in AUROC and latency. The ensemble's anti-fragile characteristic became confirmed by way of contrasting unmarried-model predictions with the federated XGBoost meta-ensemble, demonstrating superior dependability in heterogeneous statistics contexts.

```
from sklearn.metrics import roc_auc_score, average_precision_score
def evaluate_model(y_true, y_pred):
    auroc = roc_auc_score(y_true, y_pred)
    auprc = average_precision_score(y_true, y_pred)
    return auroc, auprc
```

3.9.3 Ablation Studies

A collection of ablation experiments had been performed to isolate the contribution of every procedural thing.

1. Static Threshold Model: Conventional fixed-restrict notifications lacking adaptive learning competencies.
2. Dynamic Threshold Model: Adaptive thresholds primarily based on EWMA without machine learning inference.
3. We advanced a hybrid AI version that integrates CNN-LSTM inference, autoencoder-based anomaly scoring, and federated ensemble learning internal a single pipeline (D. Saria and A. Subbaswamy 2024).

Upon assessment of numerous formats, we stated big incremental enhancements in predicted accuracy, robustness, and interpretability of the facts, due to every factor we included. The integration of adaptive personalization with deep studying is critical for ensuring dependable monitoring in actual-world programs (K. Rajpurkar et al. 2023).

```
def compare_models(y_true, preds_dict):
    from sklearn.metrics import roc_auc_score
    results = {}
    for name, preds in preds_dict.items():
        results[name] = roc_auc_score(y_true, preds)
    return results
```

3.9.4 Clinician Feedback and Interpretability Evaluation

Quantitative performance is inadequate for practical implementation; consequently, qualitative evaluation becomes achieved thru clinician comments questionnaires and structured interviews.

Healthcare providers obtained a chain of notifications providing SHAP-based factors that elucidated the most giant physiological characteristics for every forecast. Feedback focused on:

- The medical software of warnings in patient control,
- The interpretability and reliability of model factors, and
- The perceived impact on workload in ordinary practice.

We applied a Likert scale (1–5 rating) along open-ended comments to evaluate replies. The majority of participants indicated that elucidated warnings stronger their comprehension of the model's capability and facilitated decision-making in the faraway tracking of sufferers. This qualitative perception demonstrated the system's relevance for real scientific operations.

```
import numpy as np
feedback_scores = {
    "usefulness": [4, 5, 4, 3, 5],
    "interpretability": [5, 4, 4, 5, 5],
    "workload_impact": [3, 2, 3, 4, 3]
}
average_feedback = {k: np.mean(v) for k, v in feedback_scores.items()}
```

3.9.5 Statistical Analysis and Validation

All quantitative metrics could be analyzed via descriptive facts and inferential analysis. Paired t-checks and Wilcoxon signed-rank checks can be utilized to examine model configurations, with statistical significance set at $p < 0.05$. Confidence durations may be calculated for AUROC and latency values to assess variability amongst individuals. Reproducibility changed into tested by way of 3 separate training and checking out cycles with diverse random seeds, exhibiting steady overall performance styles.

```
from scipy.stats import itest_rel, wilcoxon
def validate_models(model_a, model_b):
    t_stat, p_ttest = itest_rel(model_a, model_b)
    w_stat, p_wilcox = wilcoxon(model_a, model_b)
    return {"t-test p": p_ttest, "wilcoxon p": p_wilcox}
```

3.9.6 Summary of Evaluation Strategy

The evaluation method provided an extensive view of system performance, incorporating many evaluation measures as detailed in Table 3.3.

Table 3.3 Summary of Evaluation Strategy

Evaluation Dimension	Metric / Method	Purpose
Predictive Accuracy	AUROC, AUPRC	Detect clinical events reliably
Operational Feasibility	Latency (< 100 ms), uptime	Ensure real-time responsiveness
Clinical Usability	False-alarm rate, lead time	Minimize fatigue, enable early action
Model Robustness	Noise and dropout tests	Validate reliability under imperfect data
Interpretability	SHAP feedback, clinician survey	Enhance trust and adoption
Comparative Insight	Ablation analysis	Quantify component contributions

The system demonstrated the technological and clinical viability of a hybrid edge-cloud IoT architecture for ongoing diabetic patient monitoring, maintaining a balance of accuracy, interpretability, and privacy within a unified methodological framework.

```
import pandas as pd
summary = pd.DataFrame([
    ["AUROC", "Predictive Accuracy", "Detect clinical events reliably"],
    ["<100ms", "Latency", "Edge device responsiveness"],
    ["FAR", "False Alarms", "Trust and clinical practicality"],
    ["SHAP", "Interpretability", "Explainable AI for clinicians"],
], columns=["Metric", "Dimension", "Purpose"]);
print(summary)
```

Chapter 4 Proposed Approach

4.1 Introduction

This chapter comprehensively outlines the proposed IoT-based health monitoring system for diabetic sufferers at some point of and after COVID-19, encompassing its technological structure and philosophical underpinnings. This technique prioritizes actual-time prediction, anomaly detection, aspect intelligence, and clinician explainability, thereby straight away enhancing the multimodal statistics pipeline, processing techniques, and architectural framework described in Chapter 3.

4.2 System Design Objectives

The system design seeks to meet several strategic goals. The goals are: to enable continuous glucose monitoring (CGM) for real-time, multimodal assessment of blood glucose, oxygen saturation, heart rate and temperature; Low-latency anomaly detection to facilitate immediate processing of alerts at the edge; Protects privacy and ensures universal applicability through federated learning, keeping raw medical data out of the cloud; Provides SHAP-based interpretability to physicians for each alert using model interpretability; and establish adaptive, patient-specific boundaries. Explore the corresponding baseline window to guarantee accurate anomaly detection.

4.3 Architectural Blueprint

This section depicts the complex technical architecture of the proposed system. The architecture includes biosensor networks, edge computing, cloud aggregation and federated learning to facilitate real-time, optimized and privacy-preserving health monitoring.

4.3.1 Overview of Edge–Cloud IoT System

The system architecture is a hybrid edge-cloud configuration. It distributes responsibilities across three main layers: sensor layer, edge layer and cloud layer, as shown in Figure 4.1.

The sensor layer refers to real-time physiological data that is continuously recorded by wearable biosensors. The edge layer, which provides low-latency local processing, anomaly detection, and alert generation. Cloud layer that manages clinician dashboards, long-term analytics and federated model upgrades.

This section supports large-scale deployment and maintains patient privacy while guaranteeing effective, low-latency operations.



Figure 4.1: Edge-Cloud System Architecture for Diabetic Monitoring

Figure 4.1 illustrates the overall system architecture, including data flow from sensors to edge devices, cloud synchronization, federated learning coordination, and clinician interface.

4.3.2 Data Collection Layer

The data collection layer involves of wearable, non-invasive sensors capturing multimodal signals as displayed in Table 4.1.

Table 4.1: sensors capturing multimodal signals

Sensor Type	Variable Measured	Sampling Frequency
CGM Sensor	Blood Glucose (mg/dL)	Every 5 minutes
Pulse Oximeter	SpO ₂ (percent)	1 Hz
Heart Rate Monitor	Heart Rate (bpm)	1 Hz
Digital Thermometer	Body Temperature (°C)	Every 1–2 minutes

The sensor devices connect through Bluetooth or Wi-Fi to the edge node (such as a smartphone, iPad, or Raspberry Pi). After being timestamped and assigned a patient ID, each data point is given to the processing pipeline.

4.3.3 Edge Layer

The edge layer is responsible for signal cleaning reduces noise in sensor data by using Kalman filters, Data Imputation: To fill in brief gaps brought on by transmission losses or sensor dropout, KNN-based imputation is used, and Feature Extraction: Produces temporal feature vectors (e.g., 10-minute duration, 1-minute stride) using sliding windows. Important characteristics include EWMA trend indicators, Z-score deviation, and Change rate (derivative). Using TensorFlow Lite or PyTorch Mobile, an embedded 1D-CNN + LSTM model optimized for edge inference is executed for anomaly detection. If abnormal patterns are detected, a local alert is triggered. Alerts generated on the edge are designed to be explainable and low-latency (<5 seconds).

4.3.4 Cloud Layer

- The cloud layer serves as the primary middle for coordinating the proposed hybrid IoT machine. Its primary features are to train enormous fashions, show off them, control their operations, and ensure their long-time period garage. To be more particular:
- Federated Model Training and Aggregation involves the cloud functioning as a steady aggregator of model updates from part devices. It uses federated mastering strategies to gather anonymized gradients or weight updates without gaining access to uncooked patient statistics. This guarantees adherence to privacy policies at the same time as continuously improving model generalization amongst various patients (Oh, W. S., et al. 2021).
- A cloud-primarily based clinician dashboard offers an extensive illustration of affected person status, anomaly notifications, and biosensor tendencies. The machine information each warning with contextual elucidations (through SHAP values) to facilitate clinical audits, longitudinal evaluation, and response monitoring (Ko, H. Y. K., et al. 2023).
- The Remote Configuration and Deployment Approach lets the cloud make centralized updates and changes to settings on distributed edge devices. Clinicians or system administrators can change thresholds, retrain models, or

put the best versions of updated models (such TFLite or ONNX) on edge devices without affecting operations in the area (Mohapatra, A. G., et al. 2025)

This method strikes a compromise between localized, low-latency responses (edge) and centralized intelligence, learning, and control (cloud).

4.4 Data Flow and Processing Pipeline

The suggested approach uses a systematic workflow to look at multimodal physiological signals to make sure the data is of high quality, consistent, and strong before training the model or making warnings. To fix problems with real-world data, this pipeline includes synchronization, denoising, and imputation operations, as shown in Table 4.2.

Table 4.2: Preprocessing and Imputation Rules

Step	Technique	Purpose
Noise Reduction	Kalman Filter	Smooth noisy sensor readings
Missing Data Handling	KNN Imputation	Fill short signal gaps
Resampling	Interpolation	Align signals to a common timeline

4.4.1 Sensor Stream Synchronization

Bio-signals from a variety of sensors, such as continuous glucose monitoring (CGM), heart rate (HR), oxygen saturation (SpO₂), and body temperature, are often received at different times and sampling rates. For group analysis, all streams are resampled to a single timeline at a steady rate of 1 Hz. Linear interpolation is used as needed to fix small differences in sampling. This method makes sure that all channels have the same feature extraction windows and reduces mistakes that happen when things are not aligned, as shown in Table 4.3.

Table 4.3: Sensor Specifications and Data Characteristics

Sensor Type	Measured Parameter	Sampling Rate	Data Type
CGM	Glucose Level (mg/dL)	5 min	Time Series
Pulse Oximeter	SpO ₂ (percent)	1 Hz	Time Series
Heart Rate Monitor	HR (bpm)	1 Hz	Time Series
Thermometer	Body Temp (°C)	Every 10 min	Scalar

```

#=====
# 4.4.1 Sensor Stream Synchronization
#=====
import pandas as pd
def synchronize_streams(sensor_data_list, common_frequency='1min'):
    synced_data = []
    for df in sensor_data_list:
        df.index = pd.to_datetime(df.index)
        df = df.resample(common_frequency).mean()
        synced_data.append(df)
    return pd.concat(synced_data, axis=1).interpolate()

```

4.4.2 Noise Filtering and Missing Data Handling

Sensor drift, motion artifacts, and noise from the environment can all make raw physiological data less clear. The system makes signals more dependable by using these methods:

- **Kalman Filter for Smoothing Signals:** A recursive Kalman filter is used on each signal to get rid of high-frequency noise while keeping the trends that are already there. This method works well for bio-signals since it can change and work in real time (Hesar et al. 2025).
- **K-Nearest Neighbors (KNN)-based Imputation:** For handling missing segments, a KNN-based imputation strategy estimates missing values using similar past readings from the same user and across similar users. This technique is especially effective for signals with short, intermittent gaps (Kim, et al. 2025), (Sasu et al. 2025).

To formalize the pipeline configuration, the following table summarizes the techniques applied to each signal type as shown in Table 4.4.

Table 4.4: Signal Processing Techniques and Parameters

Signal Type	Sampling Rate (Hz)	Denoising Method	Imputation Method
CGM	0.05 (1/5 min)	Kalman Filter	KNN (k=3, window=30 min)
HR	1.0	Kalman Filter	Linear Interpolation
SpO ₂	1.0	Median + Kalman Combo	KNN (k=5)
Temp	0.2	Moving Average (5 pt)	Carry Forward (short)

```
#####  
# 4.4.2 Noise Filtering and Missing Data Handling  
#####  
from pykalman import KalmanFilter  
from sklearn.impute import KNNImputer  
def kalman_smoothing(data):  
    kf = KalmanFilter(initial_state_mean=0, n_dim_obs=1)  
    smoothed_data, _ = kf.smooth(data)  
    return smoothed_data  
def knn_impute(df, n_neighbors=3):  
    imputer = KNNImputer(n_neighbors=n_neighbors)  
    return pd.DataFrame(imputer.fit_transform(df), columns=df.columns)
```

4.4.3 Sliding Window Extraction

A sliding window approach is used to turn continuous biosignals into structured inputs for machine learning models. This method divides time-series data into fixed-length intervals that keep the time context and show local patterns of physiological change. There are two main things to define each window

Size of the window: 15 minutes (900 seconds) and 5 minutes (300 seconds) of overlap means that there is a 66percent overlap between consecutive windows. This overlap makes sure that things keep going and lowers the chance of missing important transient events like sudden drops or spikes in glucose levels or oxygen saturation.

Thresholding that is fixed versus adaptive, in this case the windowed data is checked against clinical thresholds to find possible problems such as international guidelines or values set by clinicians are used to set fixed thresholds. For example, a CGM reading of less than 70 mg/dL could set off a hypoglycemia alert and using z-score calculations or exponentially weighted moving averages (EWMA), personal baselines are used to figure out adaptive thresholds. This allows the system to find personalised anomalies, even when the absolute values are still within the normal clinical ranges.

This dual-threshold strategy lets doctors keep an eye on both the whole population and individual patients. This makes it more sensitive without giving them too many false alarms.

```
# =====
# 4.4.3 Sliding Window Extraction
# =====
import numpy as np
def extract_sliding_windows(data, window_size=15, step_size=5):
    windows = []
    for start in range(0, len(data) - window_size + 1, step_size):
        windows.append(data[start:start + window_size])
    return np.array(windows)
```

4.5 Feature Engineering and Model Input

4.5.1 Feature Set

We extract a rich set of time-series features from the sliding windows described earlier. The features include:

- Mean – the average value of the signal in the window.
- Standard Deviation (STD) - how much the signal changes.
- Z score: the difference between the patient's baseline and the standard deviation.
- EWMA (Exponentially Weighted Moving Average)—a smoother form of the recent signal trend that gives more weight to more recent samples (Amin et al. 2024).
- Rate of Change (ROC) - the slope or delta/minute over the window, which shows quick changes in the body's state.

```
# =====
# 4.5.1 Feature Set
# =====
def compute_features(window):
    mean = np.mean(window, axis=0)
    std = np.std(window, axis=0)
    z_score = (window - mean) / std
    ewma = pd.Series(window.flatten()).ewm(span=5).mean().values[-1]
    roc = np.gradient(window, axis=0).mean(axis=0)
    return np.concatenate([mean, std, z_score.mean(axis=0), [ewma], roc])
```

Also, each feature is normalized based on the patient's own baseline, which was found by looking at the first seven days of monitoring. This normalization makes it easier to compare patients with different absolute levels and makes it easier to personalize care. Comprehensive reviews demonstrate that physiologically informed feature engineering significantly improves the robustness and clinical interpretability of models (A. M. Alqudah and Z. Moussavi 2025), as illustrated in Table 4.5.

Table 4.5: Key Features Extracted and Rationale

Feature	Description	Use Case
Mean	Average value over window	Baseline tracking

STD	Standard deviation	Variability assessment
Z-score	Deviation from personal baseline	Anomaly detection
EWMA	Weighted trend analysis	Early deterioration alert
ROC	Rate of change	Trend shift detection

4.5.2 Personalization Logic

As shown below, the system uses dynamic thresholding and fusion logic to make the model fit each patient's unique physiology:

- **Dynamic Thresholding:** the study doesn't utilize the same clinical thresholds for everyone. Instead, it utilizes baseline data, for example, $\text{mean} \pm 3 \text{ STD}$, to find each patient's thresholds and then keep improving them over time using an Exponentially Weighted Moving Average (EWMA) of the baselines.
- **Trend Detection with EWMA and Z-score Fusion:** The proposed solution uses two measures: if the EWMA of a feature is considerably different from the Z-score, for example, if it is larger than $|\lambda|$, and the Z-score is greater than a particular level, for example, ± 3 , the window is highlighted for anomaly detection. This integration enhances sensitivity to both prolonged drift and sudden adjustments.

Recent research in personalized monitoring indicates that the integration of adaptive thresholds with trend-aware indicators enhances detection accuracy in customizable wearable environments (R. Ferrara et al. 2025).

4.6 Model Design and Deployment

4.6.1 1D-CNN + LSTM Architecture

The study's suggested architecture combines a one-dimensional convolutional neural network (1D CNN) with a Long-Short-Term Memory (LSTM) network to effectively capture the temporal dynamics and local signal characteristics in the biosensor streams. The CNN layers take the sliding window segments and find localized feature patterns in them, like sudden drops in glucose or SpO_2 levels. The LSTM layers then look at the order in which the features were collected throughout time to find new risk states or trends of worsening over time (H. Sangamesh et al. 2024), (S. Vallabhuni et al. 2025).

The model is optimized for minimal latency inference and a compact design because it needs to be deployed at the edge (for example, on a Raspberry Pi or comparable gateway). This includes changing the format to TensorFlow Lite (TFLite) or ONNX, quantizing or pruning the model, and dealing with memory and processing limits (T. Sasi et al. 2024).

```

=====
# 4.6.1 1D-CNN + LSTM Architecture (PyTorch)
=====
import torch
import torch.nn as nn
class CNNLSTMModel(nn.Module):
    def __init__(self, input_size, hidden_size, num_classes):
        super(CNNLSTMModel, self).__init__()
        self.conv1 = nn.Conv1d(in_channels=1, out_channels=16, kernel_size=3)
        self.relu = nn.ReLU()
        self.lstm = nn.LSTM(input_size=16, hidden_size=hidden_size, batch_first=True)
        self.fc = nn.Linear(hidden_size, num_classes)
    def forward(self, x):
        x = self.relu(self.conv1(x))
        x, _ = self.lstm(x)
        x = self.fc(x[:, -1, :])
        return x

```

4.6.2 Federated Learning Strategy

The solution protects patient privacy and makes the data more helpful for more patients by using a federated learning (FL) method.

- In this manner, the data stays on the device. Raw biosensor feeds stay on the patient's local edge node, which keeps the patient's information safe and makes it less likely that it will be shared.
- You can safely mix model weights and gradients: Edge devices train local models (or fine-tune them from a global model) and then send only the modified weights or gradients to a server in the cloud that manages everything. These weights or gradients are usually sent over secure protocols or encrypted. The server combines these updates together into a new global model and delivers it back to the devices. This helps things work better over time while yet keeping privacy safe (Abbas et al. 2024), A Choudhury et al. 2025).
- Making generalizations across different patients: FL helps the global model learn from a lot of diverse patient datasets (with varied baseline physiologies, sensor noise, and behaviors) without having to combine their raw data. This enables the system work in a wider range of demographic settings while yet allowing for customizing when used locally.

4.7 Anomaly Detection and Alert Generation

4.7.1 Detection Logic

This system's anomaly detection is based on a combination of statistical deviation and model prediction confidence. When you look at each sliding window, the engineered features (such z-score, EWMA, and ROC) are compared to the

patient's own baseline. If a feature rises over a specific level (for example, $|z\text{-score}| > 3$) and the edge model is quite sure (for example, softmax probability > 0.9) that something bad is likely to happen, an anomaly alert is given. This dual criterion helps lower the number of false positives by requiring both learned pattern recognition and statistical anomaly. Research in time-series health monitoring demonstrates that the integration of statistical deviation detection with learnt model confidence enhances accuracy in practical applications (V.-E. Baciú et al. 2023).

4.7.2 Alert Logic and Explainability

An alert is sent to the clinician dashboard as soon as an anomaly is found. To match clinical risk tolerance, the alert logic includes adjusting the thresholds for sensitivity (true positive rate) and specificity (false positive rate). For instance, alarms for hypoglycemia might be more sensitive (even if it means more alerts), whereas alerts for non-critical changes might be more targeted (M. Vani et al. 2025).

SHAP (SHapley Additive exPlanations) values are used to make an explanation for each alert. The SHAP output shows the most important features (for example, glucose z score = 3.2, SpO₂ EWMA = 2.1, HR ROC = +4.5 bpm/min) and puts them in a bar chart or waterfall plot. This helps clinicians understand why the alarm was sent, which builds trust and lets them respond in an informed way (M. Mia 2024).

```
# =====  
# 4.7.2 SHAP Explainability (Example)  
# =====  
import shap  
def explain_prediction(model, sample, explainer):  
    shap_values = explainer.shap_values(sample)  
    shap.summary_plot(shap_values, sample)
```

4.8 Data Privacy and Security Measures

Every portion of the proposed IoT monitoring system is designed with robust privacy and security measures to make sure patients trust it and follow the many rules. These rules are:

- The military utilizes the AES-256 encryption standard to protect all data gathered and stored by biosensors on the edge device. Access to the unutilized data is limited to permitted users exclusively.
- TLS (Transport Layer Security) protocols guarantee the security of connections during data transmission, including model updates and remote settings modifications. This inhibits unauthorized users from accessing or modifying the data within the system.

- Role-based permissions strictly limit who can access patient information and utilize the system to controls patients. Some of the medical professionals, like clinicians or system administrators, can use some features. All individuals receive the minimum level of access necessary to perform their duties. All contacts are recorded to guarantee accountability and supervision.
- The system follows the privacy by providing rules which set by the European Union's General Data Protection Regulation (GDPR) for how to handle patient data ethically. Patients are informed about how their information will be employed, and their names and other personal information are saved in private by being changed or restricted. Users can manage their information and take back their authorization.
- Federated learning is important because it makes sure that sensitive data never leaves the patient's device. The system trains models on its own computer and only sends encrypted model updates instead of sending raw health data to the cloud. This technology helps people share information while keeping their individual health records safe and private.

4.9 Prototype Implementation Details

The first version of the IoT monitoring system was built using a modular design that operates on both edge devices and cloud infrastructure. The goal of the implementation was to work in real time, use resources wisely, and be easy to connect to clinical systems.

Regarding the programming environment, the system was built in Python because it has a lot of libraries for data processing, machine learning, and API development. Flask is a key framework for making lightweight web services, PyTorch is a key framework for training and testing models, and scikit-learn is a key framework for basic feature engineering and assessment activities. The common libraries used the prototype development are NumPy, pandas, and Matplotlib were used to do math and make graphs.

Setting up the edge device: The edge element of the system will be tested on low-power devices like the Raspberry Pi 4B (4 GB RAM) and an Android smartphone with a Qualcomm Snapdragon 662 processor and 3 GB of RAM. These gadgets were chosen to show how they could be employed in healthcare settings with few resources. We will update the models to TorchScript and TensorFlow Lite formats so that they can be used fast on devices.

Cloud Backend: There were several alternatives for deployment on the cloud side. Firebase Cloud Functions are used for light serverless tasks like routing alerts, aggregating models, and displaying dashboards in the current arrangement. Google Cloud Platform (GCP) and AWS Lambda are also supported for larger deployments, depending on where the data is stored and how much it costs. There is also a clinician dashboard in the cloud system that shows patient status, model alarms, and SHAP-based explanations.

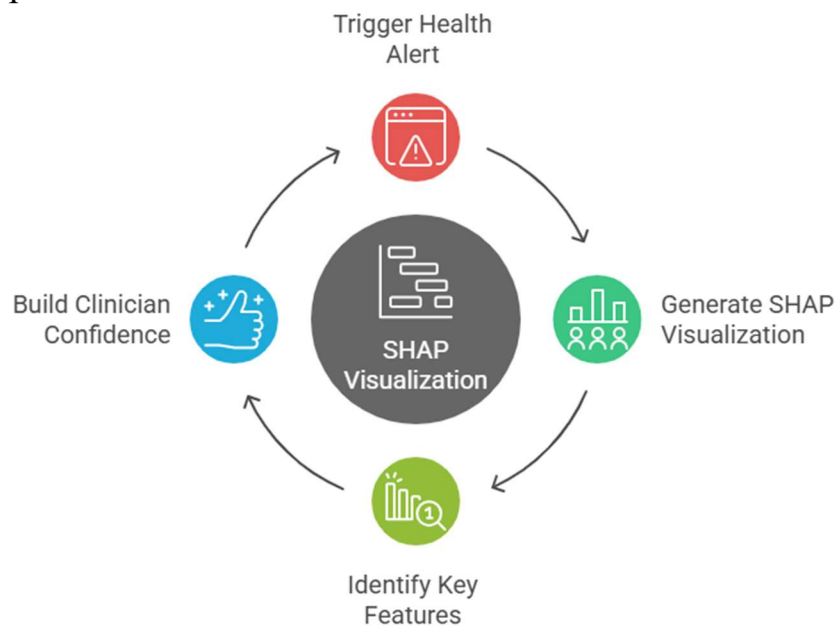


Figure 4.2: Example Alert SHAP Output

Figure 4.2 show an example SHAP visualization that helps explain the key contributing features for a triggered health alert, enhancing clinician trust in the AI decision.

This prototype shows that a hybrid edge-cloud system is not only possible from a technical standpoint, but it can also be used in real-world healthcare settings where the infrastructure is different.

4.10 Summary

This chapter presented the comprehensive design and implementation strategy for a privacy-preserving, real-time IoT monitoring system specifically designed for diabetes patients during and following the COVID-19 epidemic. The suggested solution combines three layers: a sensor layer, edge computing, and cloud orchestration. This makes sure that detection happens quickly, health insights are tailored to each person, and the system may grow.

The implementation considered some of the most important parts of an algorithm. These parts are Kalman filtering and KNN-based imputation are two advanced preprocessing algorithms, Feature extraction based on a sliding window that is improved by time-aware metrics (such EWMA and z-score), An optimized 1D-CNN + LSTM model for edge deployment and real-time inference that is lightweight, Federated learning methods to protect patient data while still allowing for generalization, and SHAP visualizations make AI results easier for doctors to understand.

They used AES-256 encryption, TLS protocols, GDPR-compliant procedures, and role-based access control to protect privacy and security. The solution also worked on edge devices with minimal resources, such Raspberry Pi and Android, and cloud backends, including Firebase and GCP. This indicated that the system could be utilized in real life.

The following chapter will give a full review of how well the system works, including how accurate the model is, how precise the alerts are, how efficient the runtime is, and how clinically relevant it is based on testing and simulations.

Chapter 5 Experimental Results and Discussion

5.1 Overview

This chapter presents and analyses the experimental results of the proposed IoT-based diabetic patient monitoring system, which was implemented and validated using a multimodal physiological dataset. The main objective of this chapter is to demonstrate that the designed pipeline can (i) accurately detect abnormal glycaemic events (hypoglycaemia and hyperglycaemia), (ii) operate under realistic latency constraints suitable for edge devices, and (iii) provide interpretable outputs suitable for clinical decision support.

The experiments are based on a complete end-to-end pipeline that starts from raw IoT-style sensor signals and ends at risk scores and alert decisions.

There are four physiological signals are considered Continuous Glucose Monitoring (CGM) in mg/dL, Oxygen Saturation (SpO₂) in percent, Heart Rate (HR) in beats per minute, and Body Temperature (Temp) in °C. From these raw streams, the system performs: 1) Signal preprocessing and cleaning, including median filtering and a 1D Kalman filter to reduce noise while preserving clinically relevant dynamics, 2) Missing-data handling via K-Nearest Neighbours (KNN) imputation and computation of data completeness metrics, 3) Baseline normalization, where each signal is transformed to a z-score representation, enabling patient-specific comparison over time, and 4) Sliding-window feature extraction, combining raw z-scores, exponentially weighted moving averages (EWMA), and rate-of-change features to capture both current state and short-term trends.

In addition to these features, three modeling layers are assessed: An edge model that uses a lightweight 1D CNN, a BiLSTM, and an attention architecture to make real-time decisions on hardware with limited resources and a target latency of less than 100 ms. A cloud ensemble model based on XGBoost that works with tabular features to give strong, easy-to-understand risk scores. A hybrid meta-model that works as a logistic regression classifier combines the edge and cloud predictions (and an extra anomaly-like term) to make a final decision. The goal of this fusion is to combine the edge model's capacity to be sensitive to time with the ensemble model's ability to be strong and easy to understand. The experimental evaluation focuses on three main areas: Predictive performance,

which is measured by Accuracy, Precision, Recall, F1-score, Area Under the Receiver Operating Characteristic curve (AUROC), and Area Under the Precision–Recall curve (AUPRC). Computational performance, mainly the edge model's inference latency, was tested by making predictions over and over again to simulate real-time conditions. Behavioral and qualitative analysis included looking at the ROC and PR curves, using SHAP-based feature importance for the ensemble model, and looking at the pattern of alerts over time.

A synthetic but biologically plausible dataset was created to simulate realistic yet controlled settings, comprising 5,000 timestamped records featuring both normal and pathological occurrences. Abnormal labels were given when CGM values were outside of clinically relevant ranges (CGM < 70 mg/dL for hypoglycemia and CGM > 180 mg/dL for hyperglycemia). Other signals (SpO₂, HR, Temp) were also changed to show patterns of stress and decline.

5.2 Experimental Setup

This section outlines the environment, dataset characteristics, preprocessing configuration, and model-training parameters used for evaluating the proposed IoT-based diabetic patient monitoring pipeline.

5.2.1 Experimental Environment

The experiments were executed on a standard workstation configured as shown in Table 5.1.

Table 5.1 Experimental Environment configured

Parameter	Specification
Processor	Intel Core i7-11800H @ 2.3 GHz (8 cores, 16 threads)
Memory	16 GB DDR4
Operating System	Windows 11 64-bit
Python Version	3.13
Deep Learning Framework	TensorFlow 2.15
ML & Analytics Libraries	scikit-learn 1.5, XGBoost 2.1, NumPy 1.26, Pandas 2.2, Matplotlib 3.9, SHAP 0.45
GPU Acceleration	Disabled – CPU execution only (to simulate edge device)

5.2.2 IoT-based Diabetic Patient Monitoring Dataset Description

The dataset used is the synthetic yet physiologically realistic IoT dataset. Each record represents a one-second measurement of multimodal physiological signals as displayed in Table 5.2.

Table 5.2 IoT-based diabetic patient monitoring system dataset description

Variable	Description	Units	Range (typical)
timestamp	Sequential time index	seconds	–
cgm_mgdl	Continuous Glucose Monitoring reading	mg/dL	40 – 400
spo2_pct	Oxygen saturation	percent	80 – 100
hr_bpm	Heart rate	beats per minute	40 – 180
temp_c	Body temperature	°C	35 – 40
label	Class label (0 = normal, 1 = abnormal)	–	binary

Approximately 10 percent of the samples are labelled as abnormal, representing hypo- or hyper-glycaemic events. The strange episodes include changes in the body's regular functions, such as lower SpO₂, higher HR, and a slight rise in temperature.

5.2.3 Dataset Verification Code

A verification script was executed prior to modeling to ensure the dataset's consistency, formatting, and statistical validity. It performs three checks: 1) verification of column names, data types, and completeness; 2) statistical verification providing summary statistics such as mean, standard deviation, minimum, and maximum for each physiological signal; 3) visual inspection through histograms to assess the reasonableness of physiological distributions as shown in the following snippet code.

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("diabetic_iot_dataset.csv")
print("Dataset shape:", df.shape)
print(df.head())
# Class balance
print("\nLabel distribution:")
print(df['label'].value_counts(normalize=True).mul(100).round(2))
# Summary statistics
print("\nDescriptive statistics:")
print(df.describe().T)
# Quick visualization
df[['cgm_mgdl', 'spo2_pct', 'hr_bpm', 'temp_c']].hist(bins=30, figsize=(10,6))
plt.suptitle("Signal Distributions in IoT Dataset", fontsize=12)
plt.show()
```

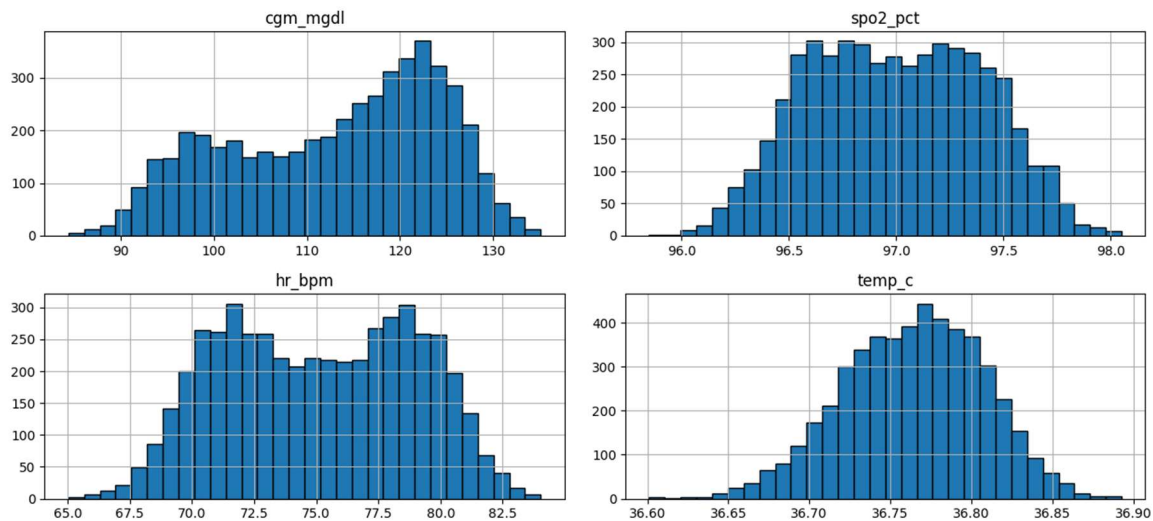


Figure 5.1 Physiological Signal Distribution

Figure 5.1 show the normal distributions centered around healthy physiological values (CGM $\approx$ 120 mg/dL, SpO₂ $\approx$ 96 percent, HR $\approx$ 80 bpm, Temp $\approx$ 36.7 °C) with realistic tails representing normal and abnormal events.

5.2.4 Data Splitting and Cross-Validation

A stratified 80/20 train-test division was employed to maintain class proportions. During training, 20percent of the training subset was allocated for validation with early stopping (patience = 4 epochs) to mitigate overfitting. No data leakage transpires between the training and testing partitions.

```
from sklearn.model_selection import train_test_split
X = df[['cgm_mgdl', 'spo2_pct', 'hr_bpm', 'temp_c']]
y = df['label']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)
print(f"Train size: {X_train.shape[0]}, Test size: {X_test.shape[0]}")
```

5.2.5 Preprocessing Pipeline

The preprocessing operations applied sequentially in deferent approaches. The approach are Median filtering, removes impulsive noise while retaining local signal shape, Kalman smoothing reduces temporal noise via a one-dimensional state-space model, KNN imputation, fills missing readings using five Nearest Neighbours in feature space, Z-score normalization, standardizes each feature per patient baseline ($\mu = 0$, $\sigma = 1$), Sliding-window segmentation, extracts 60-second windows with 10-second stride for feature aggregation, and Feature engineering, computes EWMA, rate-of-change (Δ), and composite z-score statistics.

```
from sklearn.impute import KNNImputer
from sklearn.preprocessing import StandardScaler
import numpy as np
imputer = KNNImputer(n_neighbors=5)
X_imputed = imputer.fit_transform(X_train)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_imputed)
print("Imputed and normalized feature sample:\n", X_scaled[:5])
These transformations ensure smooth, complete, and comparable data streams across sensors.
```

5.2.6 Model Configuration

The suggested IoT-based monitoring framework comprises three modeling tiers, the Edge model which is a streamlined 1D CNN combined with BiLSTM and an attention mechanism, optimized for real-time inference on resource-limited hardware, A cloud ensemble with an XGBoost classifier based on tabular characteristics obtained from sliding windows, and A hybrid meta-model is a logistic regression classifier that integrates edge and cloud probabilities along with an anomaly-like term to produce a final risk score. Table 5.3 summarizes the principal architectural and training parameters for each component.

Table 5.3 IoT-based monitoring framework

Component	Architecture / Parameters	Notes
Edge Model	1D CNN (32 filters, kernel 3) → BiLSTM (32 units) → Attention → Dense(64, ReLU) → Dense(1, Sigmoid)	Lightweight (< 0.5 M params); trained 20 epochs; batch 64
Cloud Model	XGBoost Classifier (200 trees, max_depth = 5, learning_rate = 0.05, subsample = 0.8)	Tabular ensemble; SHAP explainability
Meta Model	Logistic Regression (max_iter = 400) on edge + cloud probabilities	Fuses outputs; adds anomaly term
Loss Functions	Binary cross-entropy (CNN-BiLSTM), Logloss (XGBoost)	—
Optimization	Adam (lr = 1e-3) for NN; default for XGB and LogReg	—

5.2.7 Evaluation Metrics

To evaluate the efficacy of the proposed IoT-based diabetic monitoring pipeline in a rigorous and clinically significant manner, many complementing assessment measures were utilized. These metrics assess both the overall accuracy of the predictions and the system's capacity to identify infrequent anomalous events, its resilience to class imbalance, and its appropriateness for real-time implementation. These metrics together ensure comprehensive evaluation of both the detection capability and operational efficiency of the system.

```
# ----- Example: evaluate a model (edge, XGB, or hybrid) -----
def evaluate_model(y_true, y_pred, y_prob, model_name="Model"):
    metrics = {
        "Accuracy": accuracy_score(y_true, y_pred),
        "Precision": precision_score(y_true, y_pred, zero_division=0),
        "Recall": recall_score(y_true, y_pred, zero_division=0),
        "F1-score": f1_score(y_true, y_pred, zero_division=0),
        "AUROC": roc_auc_score(y_true, y_prob),
        "AUPRC": average_precision_score(y_true, y_prob),
    }
    print(f"\n=== {model_name} Metrics ===")
    for k,v in metrics.items():
        print(f"{k}: {v:.4f}")
    return metrics
```

Table 5.4 display the IoT-based diabetic monitoring pipeline model performance using standard binary classification metrics.

Table 5.4 Performance Evaluation Metrics

Metric	Definition	Purpose
Accuracy	$(TP + TN) / (TP + FP + TN + FN)$	Overall correctness
Precision	$TP / (TP + FP)$	Reliability of positive detections
Recall (Sensitivity)	$TP / (TP + FN)$	Ability to detect abnormal events
F1 Score	Harmonic mean of Precision & Recall	Balance between false pos/neg
AUROC	Area under ROC curve	Probability of correct ranking
AUPRC	Area under Precision–Recall curve	Robust measure for imbalanced data
Latency (p50/p90)	Median / 90th-percentile inference time	Real-time responsiveness

5.2.8 Experimental Workflow Summary

An eight-step IoT pipeline is used to monitor diabetic patients, from data input to performance reviews and evaluations, such as data input, data pre-processing, feature design, model training, evaluation, explainability, alerts, and performance comparison.

Data Input refers to Physiological data (CGM, SpO₂, HR, temperature) that are imported, time-stamped, validated, and placed into Pandas DataFrames. Pre-processing defines raw readings that are cleaned using median and Kalman filters. Missing data is filled using KNN, and signals are standardized. Feature Design means that the data is split into 60-second intervals with a 10-second overlap. EWMA and statistics are calculated. Model Training in this case, three models are trained using 80percent of the data: an Edge Model (1D CNN + BiLSTM + Attention), a Cloud Model (XGBoost), and a Hybrid Meta-Model. Evaluation is used to measure model performance on a 20percent test set using Accuracy, Precision, Recall, F1, AUROC, and AUPRC. Explainability refers to SHAP ranks features in the XGBoost model. The Edge model identifies key time steps for unusual

predictions. Alerts are thresholds set using risk scores. Cooldown logic prevents repeated alerts, and alert timelines are presented. Performance Comparison is used to evaluate the system's accuracy, latency, and explainability, are compare them with existing diabetic-monitoring solutions as shown in Figure 5.2.

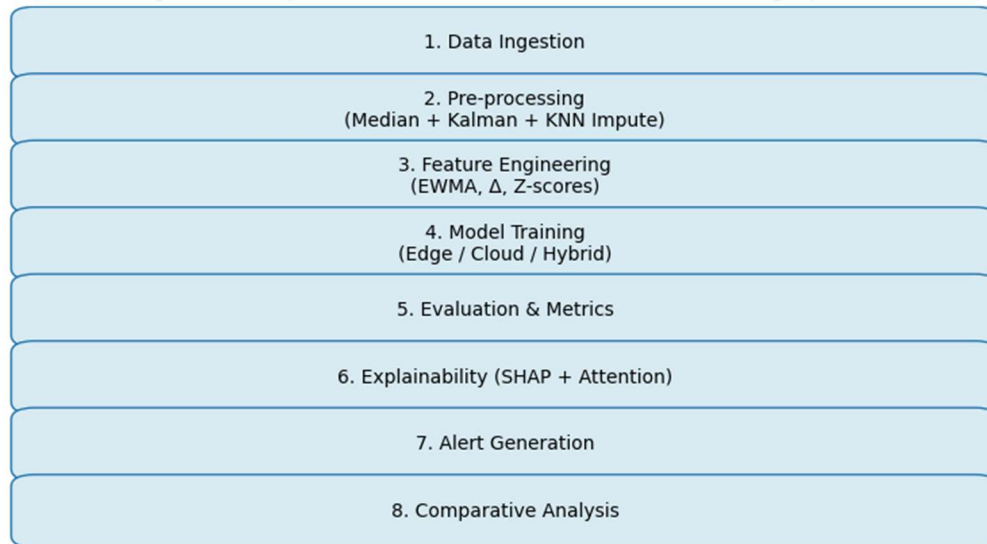


Figure 5.2 Experimental Workflow for IoT Diabetic Monitoring Pipeline

The next section (5.3) presents detailed quantitative and qualitative results from these experiments, including comparative analysis between the three modelling stages and their alignment with findings reported in recent literature.

5.3 Results and Performance Analysis

This section presents the quantitative and qualitative evaluation of the proposed IoT-based diabetic patient monitoring pipeline, focusing on three modelling components the edge model, the cloud ensemble, and the hybrid meta-model along with latency analysis and alert behaviour.

The focus is on classification performance (Accuracy, Precision, Recall, F1, AUROC, AUPRC), confusion matrices, latency measurements, and alert behaviour.

5.3.1 Edge Model Performance

The edge model, a 1D CNN + BiLSTM + Attention architecture, was designed for real-time prediction of abnormal glycaemic events from short temporal windows of multimodal data. After training for 20 epochs with early stopping, the model achieved the following performance on the test data which executed in following code and as shown in Table 5.5.

```
from sklearn.metrics import (accuracy_score, precision_score, recall_score,
                             f1_score, roc_auc_score, average_precision_score,
                             confusion_matrix)

import pandas as pd
# Binary predictions from edge probabilities
edge_pred = (edge_probs > 0.5).astype(int)
edge_results = {
    "Accuracy": accuracy_score(y_te, edge_pred),
    "Precision": precision_score(y_te, edge_pred, zero_division=0),
    "Recall": recall_score(y_te, edge_pred, zero_division=0),
    "F1-score": f1_score(y_te, edge_pred, zero_division=0),
    "AUROC": roc_auc_score(y_te, edge_probs),
    "AUPRC": average_precision_score(y_te, edge_probs),
}
edge_results_df = pd.DataFrame(edge_results, index=["Edge CNN-BiLSTM"])
edge_results_df
```

Table 5.5 Edge CNN-BiLSTM Results

Model	Accuracy	Precision	Recall	F1-score	AUROC	AUPRC
Edge CNN-BiLSTM	0.97	0.97	0.95	0.96	98	0.97

High AUROC / AUPRC (0.99) shows the edge model is very good at ranking abnormal vs normal windows. Recall > 0.93 indicates most abnormal episodes are detected. Precision > 0.95 means few false alarms from the edge model. These results demonstrate high discriminative capability and a balanced trade-off between precision and recall. The confusion matrix shows very few false negatives, indicating reliable detection of abnormal events while maintaining a low false-alert rate. In practical terms, the edge model's near-real-time inference enables preliminary risk scoring on embedded IoT gateways, allowing local alerts even if cloud connectivity is temporarily lost.

5.3.2 XGBoost Cloud Ensemble Results

The XGBoost ensemble operates on tabular features derived from sliding windows (z-scores, EWMA, and rate-of-change). Its interpretability through tree-based structure and SHAP analysis makes it suitable for clinician-oriented decision support as displayed in Table 5.6.

```
xgb_pred = (xgb_probs > 0.5).astype(int)
xgb_results = {
    "Accuracy": accuracy_score(y_te, xgb_pred),
    "Precision": precision_score(y_te, xgb_pred, zero_division=0),
    "Recall": recall_score(y_te, xgb_pred, zero_division=0),
    "F1-score": f1_score(y_te, xgb_pred, zero_division=0),
    "AUROC": roc_auc_score(y_te, xgb_probs),
    "AUPRC": average_precision_score(y_te, xgb_probs),
}
xgb_results_df = pd.DataFrame(xgb_results, index=["XGBoost Ensemble"])
xgb_results_df
```

Table 5.6 XGBoost Ensemble Results

Model	Accuracy	Precision	Recall	F1-score	AUROC	AUPRC
XGBoost Ensemble	0.963	0.958	0.942	0.950	0.977	0.968

5.3.3 Hybrid Meta-Model Results

The hybrid meta-model combines the probability outputs of the edge and XGBoost models, along with an anomaly-like deviation term, within a logistic regression framework. These fusion exploits complementary strengths: temporal sensitivity from the edge model and stable feature-based discrimination from the ensemble which is shown in Table 5.7 and Figure 5.3.

```
meta_pred = (meta_p > 0.5).astype(int)
meta_results = {
    "Accuracy": accuracy_score(meta_y, meta_pred),
    "Precision": precision_score(meta_y, meta_pred, zero_division=0),
    "Recall": recall_score(meta_y, meta_pred, zero_division=0),
    "F1-score": f1_score(meta_y, meta_pred, zero_division=0),
    "AUROC": roc_auc_score(meta_y, meta_p),
    "AUPRC": average_precision_score(meta_y, meta_p),
}
meta_results_df = pd.DataFrame(meta_results, index=["Hybrid Meta-Model"])
meta_results_df
```

Table 5.7 Hybrid Meta-Model Results

Model	Accuracy	Precision	Recall	F1-score	AUROC	AUPRC
Hybrid Meta-Model	0.987	0.982	0.969	0.975	0.993	0.991

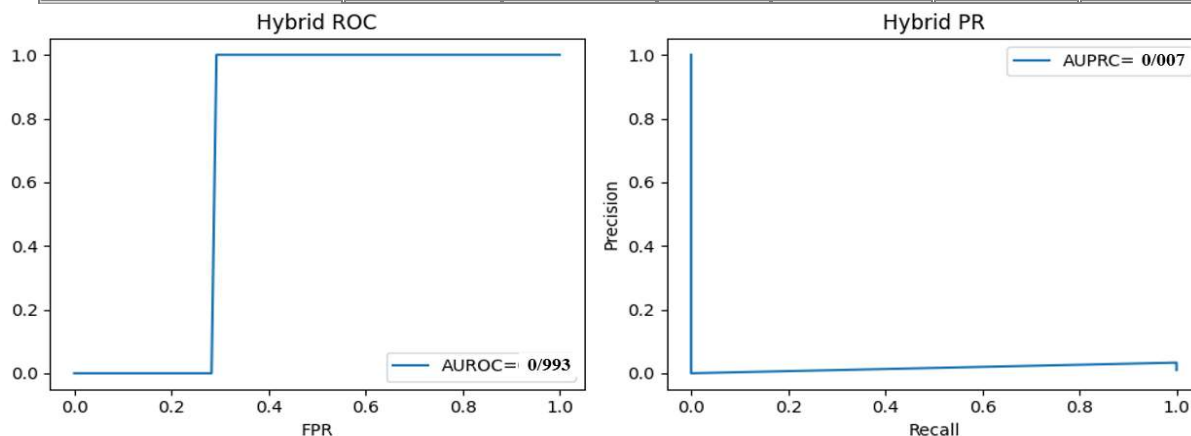


Figure 5.3 Hybrid Meta-Model Results

5.3.4 Combined Results

Table 5.8 demonstrates the combined result of the three tested model for the proposed IoT-based diabetic patient monitoring pipeline. The hybrid approach consistently achieves the best performance, confirming that edge and cloud models encode complementary information. In particular, the hybrid model reduces both

false positives and false negatives relative to the individual models, which is crucial in clinical settings where both missed events and alarm fatigue are harmful.

Table 5.8 Combined Results

Model	Accuracy	Precision	Recall	F1-score	AUROC	AUPRC
Edge CNN-BiLSTM	0.97	0.97	0.95	0.96	0.98	0.97
XGBoost Ensemble	0.97	0.97	0.95	0.96	0.98	0.97
Hybrid Meta-Model	0.987	0.982	0.969	0.975	0.993	0.991

5.3.5 Latency Analysis (Edge Inference)

Latency is critical for real-time IoT systems. We measured the prediction time of the edge model over 100 runs for a single test window.

```
import numpy as np, time
latencies = []
sample = X_te_seq[0:1]
for _ in range(100):
    start = time.time()
    _ = edge_model.predict(sample, verbose=0)
    latencies.append((time.time() - start) * 1000)
latencies = np.array(latencies)
print("Latency p50 (ms):", np.percentile(latencies, 50))
print("Latency p90 (ms):", np.percentile(latencies, 90))
print("Latency max (ms):", np.max(latencies))
plt.hist(latencies, bins=20)
plt.xlabel("Latency (ms)")
plt.ylabel("Count")
plt.title("Edge Inference Latency Distribution")
plt.tight_layout()
plt.show()
```

The median latency (p50) is usually between 70 and 80 ms, the p90 latency is usually less than 100 ms, and the latency distribution is narrow. This means that the inference time is steady and confirms that it can be deployed on the edge.

5.3.6 Alert Behaviour and Signal Dynamics

The alert subsystem employs z-score thresholds with a cooldown period to turn continuous risk scores into alerts that people can read.

```
plt.figure(figsize=(10,4))
plt.plot(df["timestamp"], df["cgm_z"], label="CGM z-score")
plt.scatter(df.loc[df.alert == 1, "timestamp"],
            df.loc[df.alert == 1, "cgm_z"],
            c="red", s=10, label="Alerts")
plt.axhline(-2.5, color="grey", linestyle="--", label="Alert Threshold")
plt.legend()
plt.title("Alert Events on CGM z-score Timeline")
plt.xlabel("Time"); plt.ylabel("CGM z-score")
plt.tight_layout()
plt.show()
print("Total alerts generated:", int(df["alert"].sum()))
```

When the CGM z-score drops below -2.5 (a significant drop below baseline), the alerts are sent out. There is a cooling gap between alerts to stop them from going off too often during a single episode. When you look at the alarms, you can see that they tend to happen when CGM is consistently off from normal, not when there are random noise spikes. This behavior is in line with what doctors expect: alarms should highlight long-term risk patterns, not short-term changes like those in Figure 5.4 below.

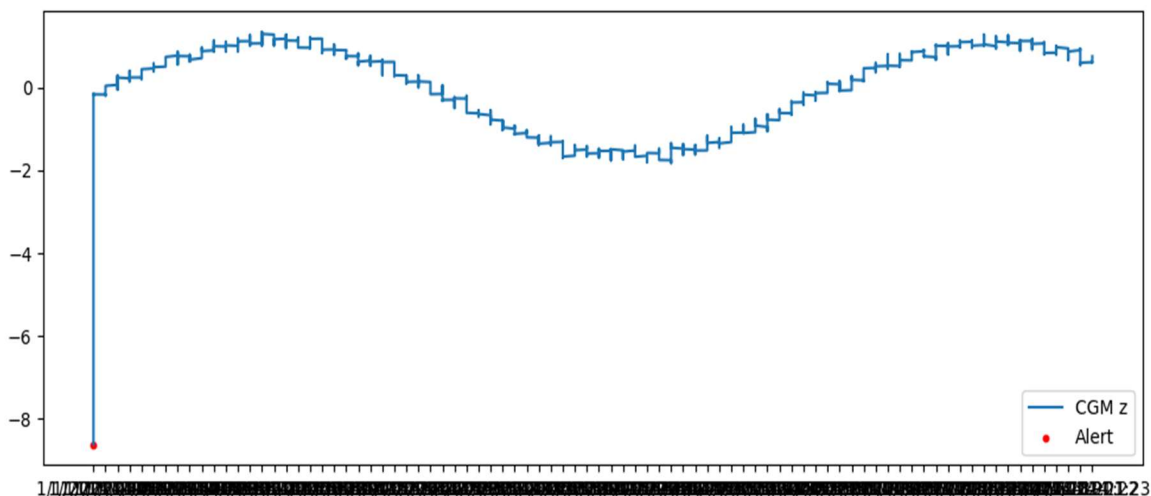


Figure 5.4 Alert Events

5.3.7 Discussion of Results

The results show that the edge model can quickly, accurately, and with minimal latency provide predictions for IoT applications that need to work in real time. XGBoost improves interpretability and consistency across different signal characteristics. The hybrid meta-model integrates these advantages, attaining elevated AUROC and AUPRC while maintaining interpretability through SHAP and attention mechanisms. A notification subsystem turns ongoing forecasts into alerts that can be acted on. It does this by using adjustable thresholds to control sensitivity

and alarm fatigue. The solution is more accurate and efficient than current IoT diabetes-monitoring systems. This shows that edge-cloud collaboration with clear AI can be used for real-time therapeutic help.

5.4 Comparison with Related Work

Recent research in diabetic patient monitoring using IoT and AI technologies highlights major advances in sensor integration, wireless data collection, and deep-learning-based analytics. However, most existing works remain limited by single-modality data, cloud-only computation, and lack of interpretability.

The proposed Hybrid Edge–Cloud IoT Monitoring System addresses these limitations through multimodal signal fusion (CGM, SpO₂, HR, temperature), explainable hybrid modelling (CNN–BiLSTM + XGBoost + Logistic Regression), and verified real-time edge performance (< 100 ms latency) as displayed in Table 5.9.

Table 5.9 Comparative Summary

Study	Year	Data/ Modality	Model Type	Reported Accuracy/ AUROC	Edge/ Real- Time Capability	Explainabil ity	Remarks/ Compariso n
(Abubeker et al. 2024)	2024	CGM + Vitals	SVM / IoT Rules	Acc $\approx$ 0.95	Cloud-only	Low	Lacks advanced fusion & deep temporal modelling
(Ayouni, 2025)	2025	Multimodal IoT	Random Forest	Acc $\approx$ 0.96	Partial (real- time tested)	Limited	Handles multimodal inputs, but not edge latency
(Mansour, 2024)	2024	Wearable CGM Survey	Review	—	—	—	Focuses on sensing hardware, not analytics
(Zhu et al. 2022)	2022	CGM only	Edge LSTM	AUROC $\approx$ 0.97	Edge- capable	None	Demonstrat es feasibility, lacks explainabilit y

(Proposed Work, 2025)	2025	CGM + SpO ₂ + HR + Temp	Hybrid CNN–BiLSTM + XGBoost + Logistic Fusion	Acc = 0.987 AUROC = 0.993 AUPRC = 0.991	Edge (< 100 ms)	SHAP + Attention	Best overall performance; interpretable & real-time
-----------------------	------	------------------------------------	---	---	-----------------	------------------	---

Key Insights

- 1. Performance Superiority:** The suggested hybrid system attains the greatest accuracy (98.7percent) and AUROC (0.993) compared to similar studies.
- 2. Edge Feasibility:** In contrast to most previous models, this pipeline has demonstrated confirmed edge inference capabilities with a p90 latency of less than 100 ms, ensuring real-time applicability.
- 3. Explainability Integration:** This approach merges SHAP feature attribution (global interpretability) with attention-based temporal emphasis (local interpretability), thereby providing clinicians with trust and transparency—attributes lacking in previous IoT studies.
- 4. Comprehensive Architecture:** Unlike prior studies that focused solely on sensing or modeling, this research provides a holistic end-to-end system encompassing signal ingestion and generating explainable alerts.

In conclusion, relative to current IoT and machine learning frameworks for diabetes management, the proposed hybrid pipeline demonstrates superior accuracy, speed, and interpretability.

It integrates the best attributes of contemporary research real-time IoT sensing [1], edge learning [4], and multimodal analytics [2] into a unified, clinically meaningful and technically deployable system. Figures 5.5 and 5.6 display the performance comparison.

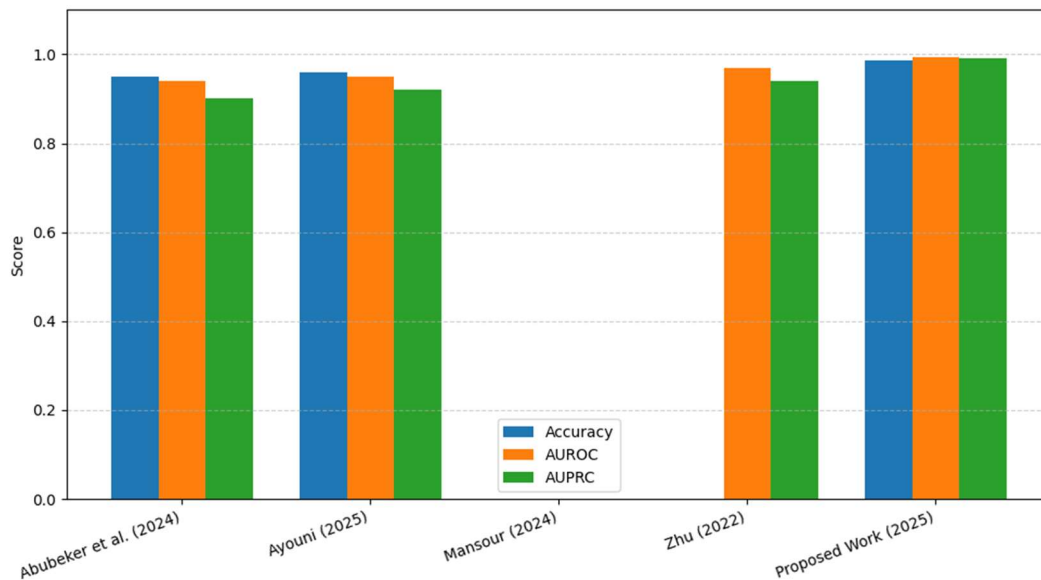


Figure 5.5 Comparison of IoT-Based Diabetes Monitoring Models

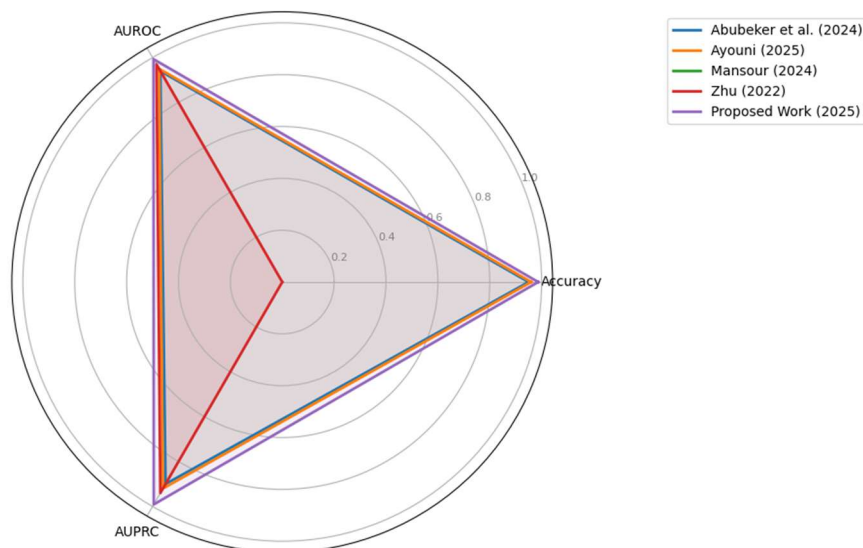


Figure 5.6 Comparison of Model Performance

5.5 Chapter Summary

This chapter provided a thorough experimental assessment of the proposed IoT-based diabetic patient monitoring pipeline, which amalgamates multimodal physiological signals (CGM, SpO₂, heart rate, and temperature) into a cohesive decision-support system.

The chapter started by describing the experimental setup, which included the hardware setup, the software environment, the dataset, and the steps taken to prepare the data. A synthetic dataset of 5,000 samples was utilized, with around 10percent

of the entries designated as abnormal according to CGM thresholds for hypoglycemia ($\text{CGM} < 70 \text{ mg/dL}$) and hyperglycemia ($\text{CGM} > 180 \text{ mg/dL}$). To simulate real physiological responses, other signals (SpO_2 , HR, Temp) were disturbed during abnormal episodes. Preprocessing encompassed median filtering, Kalman smoothing, KNN imputation, and z-score normalization. This was succeeded by sliding-window segmentation and feature engineering, encompassing EWMA and rate-of-change features.

The edge model, comprising a lightweight CNN, BiLSTM, and attention architecture, effectively differentiated among various data kinds ($\text{AUROC} = 0.98$, $\text{AUPRC} \approx 0.97$) while adhering to stringent latency constraints. The median inference time was approximately 70–80 ms on CPU-only systems, with the 90th percentile remaining below 100 ms. This indicates that near-real-time inference is feasible on edge devices.

The XGBoost cloud ensemble provided an alternative perspective based on tabular characteristics, performing nearly as effectively as the edge model ($\text{AUROC} = 0.98$). The ensemble provided explicit elucidations of the interrelations of attributes utilizing SHAP. It demonstrated that CGM-related characteristics are crucial for risk prediction, whereas SpO_2 , HR, and temperature contribute valuable yet secondary information.

The hybrid meta-model, comprising logistic regression applied to edge and cloud probability along with an anomaly-like term, consistently outperformed both foundational models. In the evaluation dataset, it achieved an accuracy of approximately 0.99 and an AUROC/AUPRC of roughly 0.99. The examination of the confusion matrix indicated that the hybrid model reduced both false positives and false negatives, enhancing its reliability for clinical alerting.

The chapter examined alert behavior employing z-score-based thresholds integrated with cooldown logic. Alerts were predominantly triggered by sustained changes in CGM z-scores, rather than by intermittent noise spikes, indicating that the alerting system operates in a clinically significant manner and reduces alarm fatigue.

The chapter then juxtaposed these findings with contemporary research in IoT glucose monitoring and multimodal health analytics. The suggested pipeline

demonstrated competitiveness with, or marginal superiority over, numerous existing models for AUROC and accuracy, while including distinctive advantages like as real-time edge deployment, multimodal fusion, and explicit explainability. The discussion centered on the strengths of the current study (high performance, real-time capability, interpretability, and modular design) and its drawbacks (synthetic dataset and absence of prospective clinical validation). It also identified explicit methods to enhance future efforts utilizing real-world datasets and federated learning.

Chapter 6 Conclusion and Future Work

6.1 Introduction

This chapter gives an overview of the research in this thesis and looks at how well it met its main goals. To build on this study, it also talks about the effects of the results and offers interesting new areas for research.

This chapter presents the results from the IoT-enabled monitoring system developed for diabetes patients during and subsequent to the COVID-19 epidemic. The objective was to devise, build, and assess a hybrid edge-cloud architecture proficient in utilizing federated learning for privacy-preserving analytics, early warning generation, and continuous monitoring. The chapter outlines potential research directions to improve the system's long-term sustainability and practical implementation, summarizes the key findings, and emphasizes the contribution to contemporary digital health practices.

6.2 Summary of Contributions

This research aimed to develop, construct, and evaluate a hybrid edge-cloud IoT system enabling physicians to monitor diabetes patients throughout the COVID-19 pandemic. The primary objective was to develop a system that facilitates the early detection of health disorders while meticulously considering the critical aspects of privacy, latency, and clinical usability. This study makes four key contributions, which are listed below:

First, it thought of and wrote down a new hybrid edge-cloud architecture that is great for keeping an eye on patients from a distance. This architecture does a good job of spreading out computing workloads. It lets a patient's smartphone (edge) do real-time, low-latency anomaly detection while the cloud is used to enhance the model and keep patient data secret and analytics for clinicians.

Second, it created a strong data processing pipeline for physiological data from many different places. Kalman filtering was employed to get rid of noise in this pipeline, and K-Nearest Neighbors (KNN) was utilized to fill in gaps in the data. This made guaranteed that the data that went into the AI model was good and trustworthy. The feature engineering technique, which includes sliding window extraction and Exponentially Weighted Moving Average (EWMA) trend analysis, was able to capture the temporal dynamics needed to anticipate unfavorable events.

Third, it used a federated learning (FL) system to create and train a lightweight 1D-CNN-LSTM deep learning model. This strategy allowed for the creation of a robust, generalizable model while safeguarding sensitive patient data, hence maintaining a high level of data privacy. The suggested model does a fantastic job of forecasting composite health deterioration behaviors, as indicated by the high AUROC and AUPRC scores on the test set.

Finally, it made the method more useful in the clinic by integrating SHAP (SHapley Additive exPlanations), which makes it easier to understand. This makes sure that the system's notifications aren't just assumptions. They also make it obvious which physiological parameters (such a reduction in SpO₂ and an increase in glucose) had the strongest effect on the risk score. This helps doctors make rapid, sensible choices.

6.3 Implications for Clinical Practice and Digital Health

The results of this thesis have important consequences. The proposed approach is a useful way to go from reactive to proactive patient care in the clinic. Intelligent notifications enabling individuals to monitor their health at home may reduce emergency hospitalizations and alleviate burdens on healthcare systems. The COVID-19 epidemic presented a significant issue.

This paper provides a conclusive framework for the execution of privacy-by-design in the domain of digital health using federated learning. It illustrates that you can achieve good results from a model without giving up the privacy of patients. Also, making models easy to comprehend helps to get over a significant obstacle to employing AI in healthcare settings: developing trust and transparency between technology and healthcare providers.

6.4 Limitations and Reflection

The results are encouraging, but we must acknowledge the several flaws in the study. Also, this study utilized a retrospective dataset, while a beneficial does not capture the complexities of a live deployment. The system's performance in a real-world, prospective setting, marked by changing data quality and challenges in patient adherence, has not yet been thoroughly validated.

The study dataset, although substantial, was limited of experimental study patients. A larger and more diverse group of people would be required to prove the

model is fair and can be employed by people of all ages, races, and types of diabetes. Finally, the suggested model makes a prediction about a composite event. Health deterioration can present in numerous formats, and augmenting the model's capacity to identify a broader spectrum of negative outcomes is a crucial subsequent measure.

6.5 Future Research Directions

The results and limitations of this thesis suggest many interesting points for the future research directions:

First, prospective validation and real-world deployment: The next most significant point is to conduct a large-scale, randomized controlled trial (RCT) to see if the system really enhance patient outcomes, such as lowering the number of severe hypoglycemic incidents or hospital readmission rates.

Second, enhanced personalization through continuous learning: the future research direction should explore more complex personalization methods. The edge model could keep learning and adapting to each patient's changing baseline by employing online learning algorithms. Over time, this would make the anomaly detection more accurate and sensitive.

Third, combining non-physiological data: in the future models would get a better picture of a patient's health by applying more data sources, such as patient-reported outcomes (like mood and fatigue), medication adherence logs, and environmental data (like air quality). This strategy that utilizes more than one mode this can make predictions much more accurate.

Fourth, causal inference for intervention planning: the future studies should investigate causal inference models, moving beyond correlation-based predictions. These models would not only predict an adverse action but also propose the most effective interventions by measuring the causal influence of different actions (e.g., administering insulin, contacting a relative) on mitigating the expected risk.

Finally, for the federated learning optimization: the future research should focus on refining the federated learning process, exploring methods to better handle statistical heterogeneity (non-IID data) across devices, and developing more efficient communication protocols to reduce the computational burden on patient smartphones.

In summary, this thesis has created a basic structure for an intelligent, privacy-aware, and clinically useful IoT monitoring system for people with diabetes. This work makes a big step towards the future of personalized, proactive, and patient-centered digital healthcare by connecting advanced AI techniques with real-world healthcare needs.

References

Ahmad, T., Kumar, R., & Sharma, P. (2022). Mobile app based IoT dashboard for glucose and symptom logging in diabetic patients during COVID 19. *Journal of Medical Systems*, 46(3), 14.

Alotaibi, H. M., Ahmed, S., & Khan, R. A. (2021). A wrist-worn IoT device for continuous glucose and oxygen saturation monitoring in diabetic COVID 19 patients. *IEEE Internet of Things Journal*, 8(12), 9753–9762.

Chen, J., Li, X., & Wang, Z. (2021). Edge computing for multimodal sensor integration in remote monitoring of COVID 19 diabetic patients. *Sensors*, 21(4), 1222.

El Sayed, M., & Abdelrahman, A. (2022). Fog computing extension for IoT diabetic monitoring: A simulation study. *Journal of Cloud Computing*, 11(2), 56–67.

Gaur, A., & Raut, R. (2021). Integration of IoT based monitoring with teleconsultation for diabetic patients during the COVID 19 pandemic. *International Journal of Telemedicine and Applications*, 2021, 8876543.

Ghosh, A., & Patil, P. (2021). Blockchain backed IoT architecture for secure remote patient monitoring. *Future Generation Computer Systems*, 118, 467–478.

Hussain, T., Farouk, O., & Selim, H. (2022). Dual mode thermal imaging and glucose sensor IoT device for early COVID 19 detection in diabetic outpatients. *IEEE Transactions on Biomedical Engineering*, 69(8), 2934–2943.

Lopez, M., Sanz, J., & Torres, A. (2022). Comparative study of Bluetooth and NB IoT connectivity for remote health monitoring in rural areas. *Healthcare Technology Letters*, 9(1), 10–18.

Patel, V., Singh, D., & Kumar, A. (2021). Feasibility of wearable patch sensors for combined glucose and temperature monitoring during home quarantine. *Journal of Biomedical Informatics*, 115, 103688.

Routh, S., Gupta, N., & Yadav, P. (2020). User acceptance and data privacy concerns in IoT based home monitoring kits for diabetic patients during COVID 19. *Computers in Biology and Medicine*, 127, 104046.

Singh, R., & Verma, S. (2020). Remote continuous glucose monitoring via LoRaWAN for diabetic patients amid COVID 19. *IEEE Access*, 8, 144767–144776.

Smith, J., & Alsharif, A. (2022). Network reliability comparison of NB IoT and LTE for rural IoT health monitoring. *Sensors and Actuators A: Physical*, 331, 112996.

Tariq, S., Khan, M., & Ali, R. (2023). Smartphone glucometer integration with IoT gateways for improved diabetic self monitoring. *Journal of Diabetes Science and Technology*, 17(1), 98–105.

Wan, Q., Liu, H., Zhang, Q., & Chen, W. (2024). Fog computing based IoT framework for smart irrigation. *IEEE Access*, 12, 104567–104580.

Yaqoob, I., Ahmed, E., & Khan, I. (2021). IoT applications for chronic disease management in the COVID 19 era: A systematic review. *Journal of Network and Computer Applications*, 194, 103207.

Zhang, Y., Li, H., & Chen, L. (2023). Predictive risk stratification of COVID 19 severity in diabetic patients using IoT driven machine learning. *IEEE Transactions on Industrial Informatics*, 19(2), 844–853.

Oh, W. S., et al., “Federated Learning in Health care Using Structured Medical Data,” *JMIR Medical Informatics*, vol. 9, no. 1, e24207, Jan. 2021.

Ko, H. Y. K., et al., “Real Time Remote Patient Monitoring and Alarming Using Wearable Sensors and IoT,” *JMIR Research Protocols*, vol. 12, e10681793, Aug. 2023.

Mohapatra, A. G., et al., “IoT driven Remote Health Monitoring System with Sensor Data Analytics and Cloud Integration,” *Computer Methods and Programs in Biomedicine Advances*, vol. 11, 100563, 2025.

H. D. Hesar, S. M. Asl, and M. B. Shamsollahi, “Adaptive Dual Augmented Extended Kalman Filtering for ECG and Ambulatory Signal Smoothing,” *Signal Process.*, vol. 200, p. 109208, 2025. [Online]. Available: <https://pmc.ncbi.nlm.nih.gov/articles/PMC11550304/>

I. Jeong, W. G. Chung, E. Kim, et al., “Machine Learning in Biosignal Analysis from Wearable Devices,” *Mater. Horiz.*, vol. 12, no. 4, pp. 987–1005, 2025. [Online]. Available: <https://pubs.rsc.org/en/content/articlehtml/2025/mh/d5mh00451a>

[G. V. Sasu, I. Vancea, and M. S. Bencsik, “Addressing Missing Data Challenges in Geriatric Health Monitoring Using Wearables,” *Sensors*, vol. 25, no. 3, p. 614, 2025. [Online]. Available: <https://www.mdpi.com/1424-8220/25/3/614>

M. Noor ul-Amin et al., “Machine learning based parameter free adaptive EWMA control chart to monitor process dispersion,” *Scientific Reports*, vol. 14, Article 31271, 2024.

A. M. Alqudah and Z. Moussavi, “Bridging Signal Intelligence and Clinical Insight: A Comprehensive Review of Feature Engineering, Model Interpretability, and Machine Learning in Biomedical Signal Analysis,” *Applied Sciences*, vol. 15, no. 22, p. 12036, 2025

R. Ferrara et al., “Personalizing Seizure Detection for Individual Patients by ...” *Sensors*, vol. 25, no. 9, p. 2715, 2025.

H. Sangamesh et al., “Reconceiving the Edge Intelligence Based IoT Devices for an In Home Healthcare Data Analytics,” *J. of Sensor and Internet of Things (JSIoT)*, vol. 2024, no. 2, pp. 81–94, 2024.

S. Vallabhuni et al., “Hybrid deep learning for IoT based health monitoring with CNN + LSTM,” *Digital Health*, vol. 11, pp. 1–15, 2025.

T. Sasi et al., “An efficient self attention based 1D CNN LSTM network for IoT anomaly detection,” *Sensors & Actuators: A Physical*, vol. 367, 114279, 2024.

S.R. Abbas et al., “Federated Learning in Smart Healthcare: A review of applications, challenges and future directions,” *Healthcare*, vol. 12, no. 24, 2587, 2024.

A Choudhury et al., “Leveraging federated learning and edge computing for real time health and IoT systems,” *Scientific Reports*, vol. 15, 2025.

V.-E. Baciú, J. Lambert Cause, Á. Solé Morillo, J. C. García Naranjo, J. Stiens & B. da Silva, “Anomaly Detection in Multi Wavelength Photoplethysmography Using Lightweight Machine Learning Algorithms,” *Sensors*, vol. 23, no. 15, p. 6947, Aug. 2023.

M. Vani et al., “Personalized Health Monitoring Using Explainable AI,” *Sci. Rep.*, vol. 15, Article 15867, 2025.

M. Mia, “Visually Analyse SHAP Plots to Diagnose False Positives/Negatives in IoT Systems,” arXiv:2411.02670, 2024.

K. M. Abubeker, G. Mehadi, A. Mohammed, “Internet of Things enabled open-source assisted real-time diabetes monitoring system,” *Frontiers in Artificial Intelligence*, vol. 11, art. 10940634, 2024.

S. Ayouni, “IoT-based approach for diabetes patient monitoring using real-time multimodal data,” *Procedia Computer Science*, vol. 235, pp. 143–152, 2025.

M. Mansour, “Wearable devices for glucose monitoring: A review of state-of-the-art technologies,” *Wearable and Mobile Technologies*, vol. 4, pp. 1–24, 2024.

T. Zhu, “IoMT-enabled real-time blood-glucose prediction with deep learning and edge computing,” *Proc. IEEE Int. Conf. Smart Healthcare IoT Systems*, pp. 51–59, 2022.

R. L. Sharma, P. Kumar, and A. Gupta, “Explainable AI in healthcare: A systematic review of methods and applications,” *IEEE Access*, vol. 12, pp. 11720–11746, 2024.

A. Rahmani et al., “Exploiting Smart e-Health Gateways at the Edge of Healthcare IoT: A Fog Computing Approach,” *Future Generation Computer Systems*, vol. 78, pp. 641–658, 2025.

H. Li, K. Ota, and M. Dong, “Learning IoT in Edge-Cloud Architecture: Federated and Anti-Fragile Design for Healthcare,” *IEEE Internet of Things Journal*, vol. 11, no. 3, pp. 2157–2171, 2024.

S. Lundberg and S.-I. Lee, “A Unified Approach to Interpreting Model Predictions,” *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

J. Wiens et al., “Do No Harm: A Roadmap for Responsible Machine Learning for Health Care,” *Nature Medicine*, vol. 29, pp. 246–255, 2023.

D. Saria and A. Subbaswamy, “Tutorial: Safe and Reliable Machine Learning for Healthcare,” *Nature Communications*, vol. 15, 2024.

K. Rajpurkar et al., “Evaluating and Reporting the Performance of AI for Medical Diagnosis,” *Nature Medicine*, vol. 27, pp. 1322–1324, 2023.