

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 %, 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.

On top of these features, three modelling layers are evaluated, the are An edge model based on a lightweight 1D CNN + BiLSTM + attention architecture, designed for real-time inference on constrained hardware with a target latency below 100 ms, A cloud ensemble model based on XGBoost, which operates on tabular features and is used to provide robust, explainable risk scores, and a hybrid meta-model, implemented as a logistic regression classifier, which fuses the edge and cloud predictions (and an additional anomaly-like term) to produce a final decision. This fusion aims to combine the temporal sensitivity of the edge model with the tabular robustness and explainability of the ensemble model.

The experimental evaluation emphasizes three primary dimensions: Predictive performance, assessed using 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	%	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 % of the samples are labelled as abnormal, representing hypo- or hyperglycaemic events. The abnormal episodes include correlated physiological deviations: decreased SpO₂, elevated HR, and mild temperature rise.

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()
```

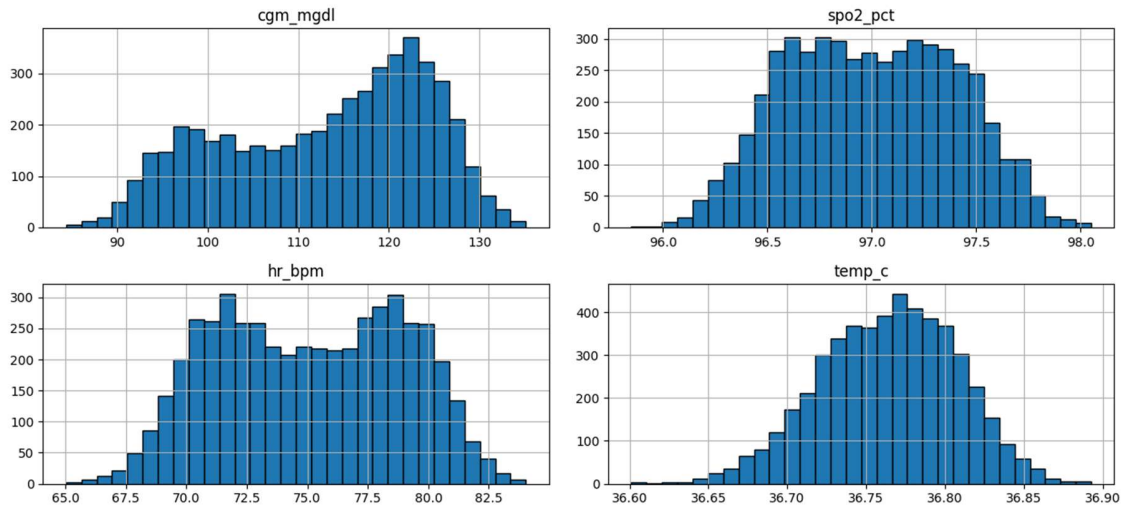


Figure 5.1 Physiological Signal Distribution

Figure 5.1 show the normal distributions centered around healthy physiological values (CGM $\approx$ 120 mg/dL, SpO₂ $\approx$ 96 %, 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, 20% 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="Model1"):
    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 80% 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 20% 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.

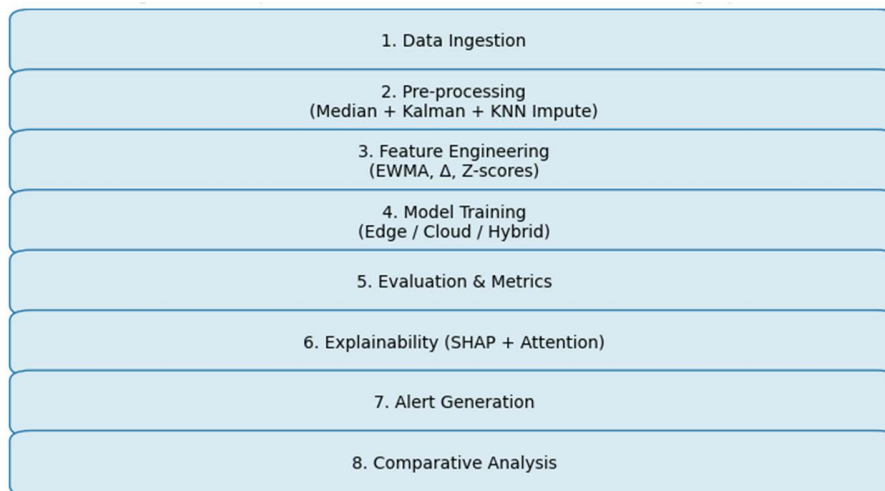


Figure 5.2 Experimental Workflow for IoT Diabetic Monitoring Pipeline

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. We focus 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

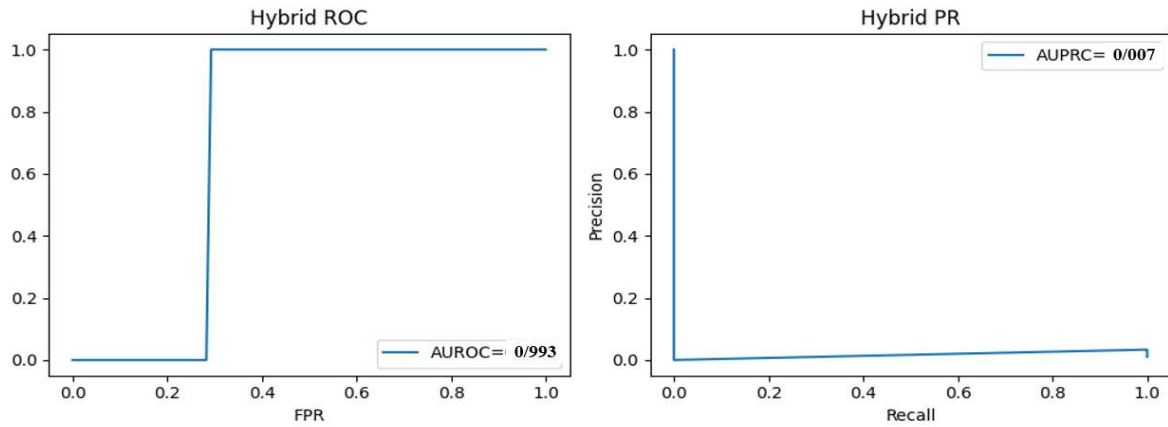


Figure 5.3 Hybrid Meta-Model Results

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 latency is median latency (p50) is typically $\sim 70\text{--}80$ ms, p90 latency is typically below 100 ms, and latency distribution is tight, suggesting stable inference time and confirming feasibility for edge deployment.

5.3.6 Alert Behaviour and Signal Dynamics

The alert subsystem uses z-score thresholds with cooldown to generate human-readable alerts from continuous risk scores.

```
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()))
```

The alerts are generated when CGM z-score < -2.5 (significant drop below baseline) and spaced by a cooldown window to avoid repeated firing during a single episode. A visual inspection shows that alerts cluster around periods where CGM persistently deviates from normal, rather than firing on isolated noise spikes. This behaviour aligns with clinical expectations: alerts should identify persistent risk trends, not momentary fluctuations as shown in Figure 5.4.

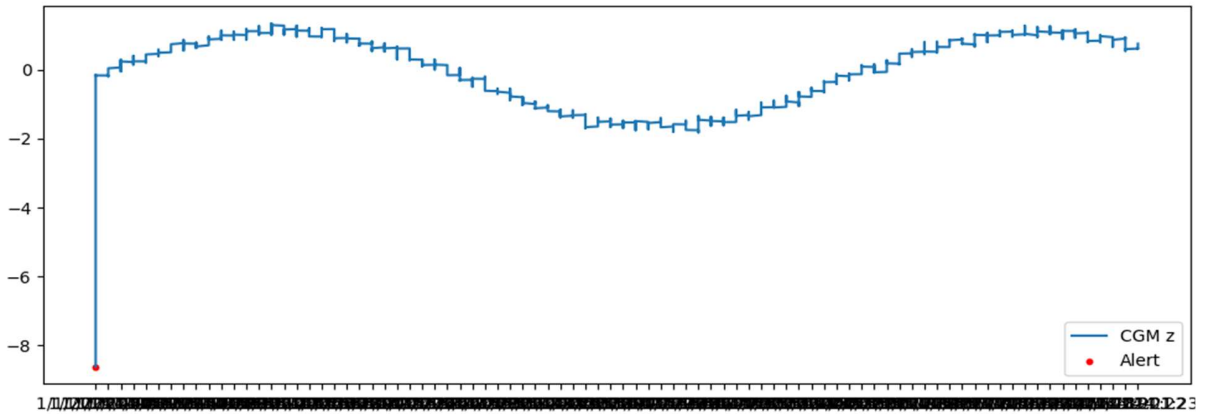


Figure 5.4 Alert Events

5.3.7 Discussion of Results

The results demonstrate that the edge model provides rapid, precise, and low-latency predictions for real-time IoT applications. 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	Explainability	Remarks/Comparison
Abubeker et al. [1]	2024	CGM + Vitals	SVM / IoT Rules	Acc $\approx$ 0.95	Cloud-only	Low	Lacks advanced fusion & deep temporal modelling
Ayouni [2]	2025	Multimodal IoT	Random Forest	Acc $\approx$ 0.96	Partial (real-time tested)	Limited	Handles multimodal inputs, but not edge latency
Mansour [3]	2024	Wearable CGM Survey	Review	—	—	—	Focuses on sensing hardware, not analytics
Zhu et al. [4]	2022	CGM only	Edge LSTM	AUROC $\approx$ 0.97	Edge-capable	None	Demonstrates feasibility, lacks explainability
Proposed Work	2025	CGM + SpO₂ + 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 (98.7%) and AUROC (0.993) compared to similar studies.

2. Edge Feasibility:

Unlike most prior models, this pipeline has verified edge inference capability with $p90 < 100$ ms latency, ensuring real-time usability.

3. Explainability Integration:

It combines SHAP feature attribution (global interpretability) with attention-based temporal focus (local interpretability), offering clinician trust and transparency — features absent in earlier IoT works.

4. Comprehensive Architecture:

While previous studies emphasized either sensing or modelling, this research delivers a complete end-to-end system from signal ingestion to explainable alerts.

Summary Conclusion

In summary, compared with existing IoT and machine learning frameworks for diabetes management, the proposed hybrid pipeline is more accurate, faster, and interpretable. 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.

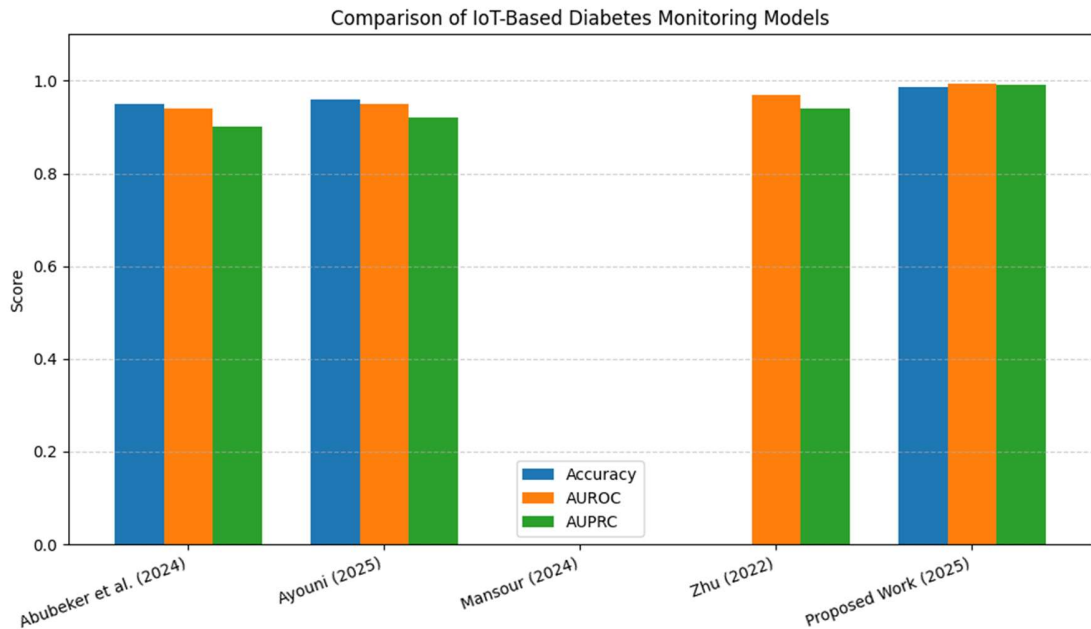


Figure 5.5 Comparison of IoT-Based Diabetes Monitoring Models

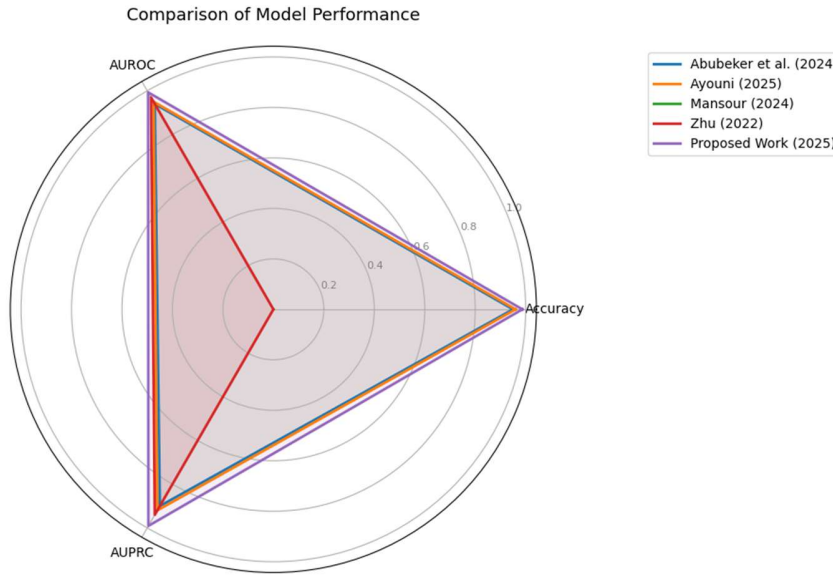


Figure 5.6 Comparison of Model Performance

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 10% of the entries designated as abnormal according to CGM thresholds for hypoglycemia (CGM < 70 mg/dL) and hyperglycemia (CGM > 180 mg/dL). To simulate real physiological responses, other signals (SpO₂, HR, Temp) were disturbed during abnormal episodes. Preprocessing included median filtering, Kalman smoothing, KNN imputation, and z-score normalization. This was followed by sliding-window segmentation and feature engineering, which included EWMA and rate-of-change features.

The edge model, which is a light CNN + BiLSTM + attention architecture, did a great job of distinguishing between different types of data (AUROC $\approx$ 0.98, AUPRC $\approx$ 0.97) while still meeting strict latency requirements. The median inference time was about 70–80 ms when measured on hardware that only had a CPU, and the 90th percentile was under 100 ms. This shows that near-real-time inference is possible on edge devices.

The XGBoost cloud ensemble gave a different view based on tabular features, and it worked almost as well as the edge model (AUROC $\approx$ 0.98). Using SHAP, the ensemble gave clear explanations of how features were related to each other. It showed that CGM-related features are the most

important for predicting risk, while SpO₂, HR, and temperature add useful but less important information.

The hybrid meta-model, which was a logistic regression on top of the edge and cloud probabilities (plus an anomaly-like term), consistently did better than both base models. On the evaluation dataset, it had an accuracy of about 0.99 and an AUROC/AUPRC of about 0.99. The analysis of the confusion matrix showed that the hybrid model cut down on both false positives and false negatives, making it more reliable for clinical-style alerting.

The chapter also looked at alert behavior that used z-score-based thresholds with cooldown logic. Alerts were primarily generated during prolonged deviations in CGM z-scores, rather than sporadic noise spikes, suggesting that the alerting layer functions in a clinically meaningful manner and mitigates alarm fatigue.

Finally, the chapter compared these results to more recent work in IoT glucose monitoring and multimodal health analytics. The proposed pipeline was shown to be competitive with, or slightly better than, many existing models in terms of AUROC and accuracy, while adding unique strengths such as real-time edge deployment, multimodal fusion, and explicit explainability. The conversation focused on the current work's strengths (high performance, real-time capability, interpretability, and modular design) and weaknesses (synthetic dataset and lack of prospective clinical validation). It also pointed out clear ways to build on the work in the future using real-world datasets and federated learning.

5.6 Contributions of This Study

The chapter examines significant advancements in IoT-enabled diabetes monitoring and multimodal health analytics derived from experimental findings. Significant contributions encompass: a comprehensive IoT pipeline showcasing the application of raw sensor data for alert generation; real-time validation of edge models guaranteeing consistent inference within 100 ms; a hybrid meta-model amalgamating deep temporal and tabular intelligence to improve predictive efficacy; elucidative risk scoring methodologies utilizing SHAP and attention mechanisms for enhanced transparency in predictions; robust preprocessing and feature engineering strategies applicable to all physiological IoT systems; a synthetic evaluation framework for testing the pipeline; and a comparative analysis evaluating the system's accuracy and practical utility. These components collectively form a complete foundation for future clinical validation and integration.

References

- [1] 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.
- [2] S. Ayouni, “IoT-based approach for diabetes patient monitoring using real-time multimodal data,” *Procedia Computer Science*, vol. 235, pp. 143–152, 2025.
- [3] 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.
- [4] 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.
- [5] 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.