

AutoEncoder-based Detection of Insulin Pump Faults in Type 1 Diabetes Treatment

IEEE Journal of Biomedical and Health Informatics (JBHI)

Published: Feb 2025

INTRODUCTION

- ❖ Patients with T1D depend on lifelong insulin therapy to keep blood glucose (BG) levels within a healthy range.
- ❖ Technological advances, such as **CGM** and **insulin pumps (CSII)**, have improved T1D management by providing real-time glucose readings and automated insulin delivery.
- ❖ However, these devices are still prone to **malfunctions!**

INTRODUCTION

❖ What is Insulin Pump Faults (IPFs)?

- ❖ IPFs = Malfunctions that stop or reduce insulin delivery without the patient being immediately aware.
- ❖ **Examples:** Infusion set occlusions, kinks in the catheter, or hardware/software errors that stop basal or bolus insulin delivery.

❖ Why detecting IPFs is hard?

- ❖ Insulin effect is delayed (~45 min), so BG looks normal at first.
- ❖ At night, faults are more dangerous since the patient is asleep.
- ❖ Some pumps raise alarms, but many “silent occlusions” go undetected.

INTRODUCTION

❖ **Why detecting IPFs is important?**

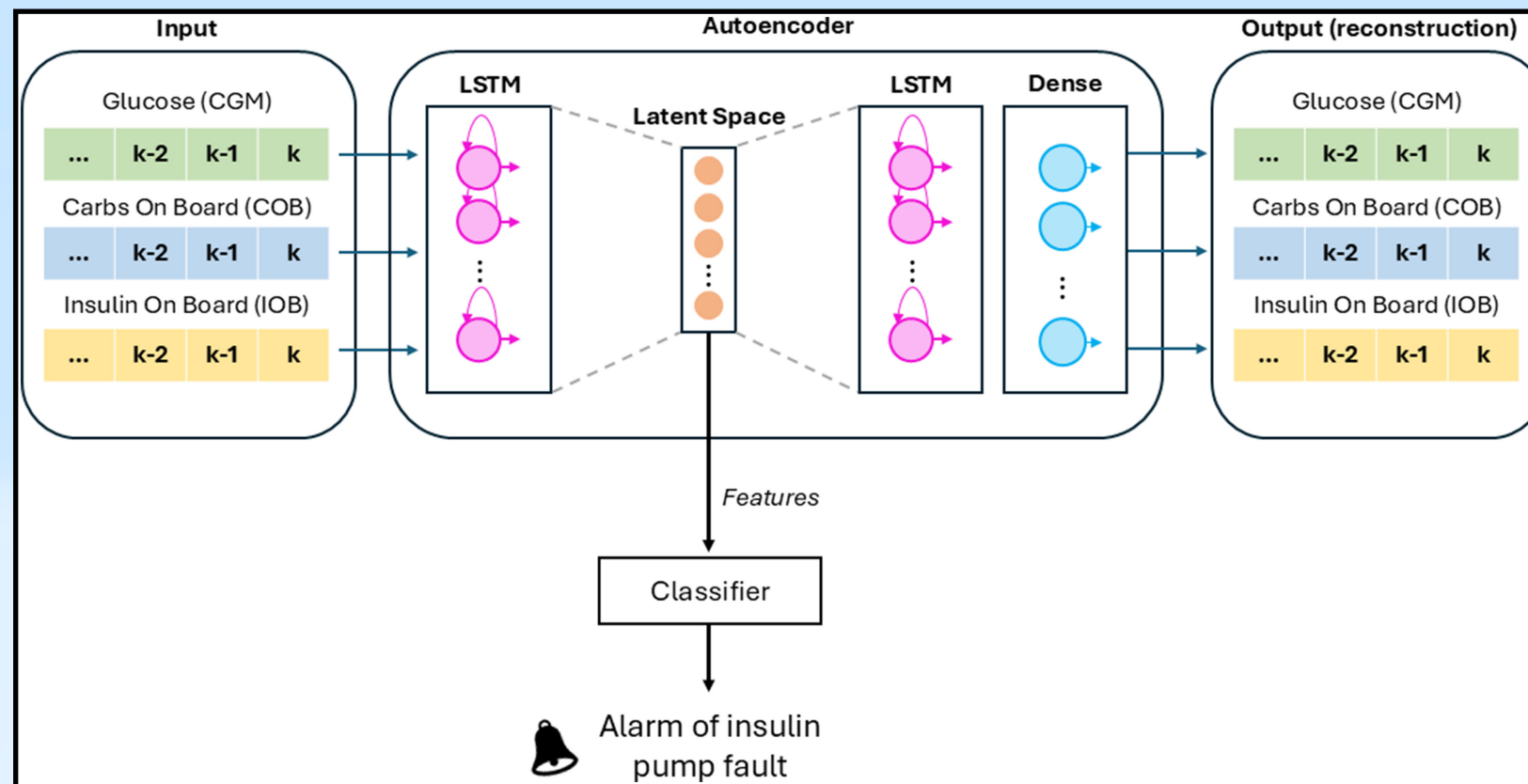
- ❖ Studies: 33–50% of patients experience undetected faults.
- ❖ Detecting IPFs early is critical for safety and preventing long-term damage.

❖ **They addressed the problem of the real-time IPF detection by:**

- ❖ Developing a deep learning approach
- ❖ Based on a recurrent auto-encoder for the automatic feature extraction,
- ❖ And a random forest classifier

METHODS

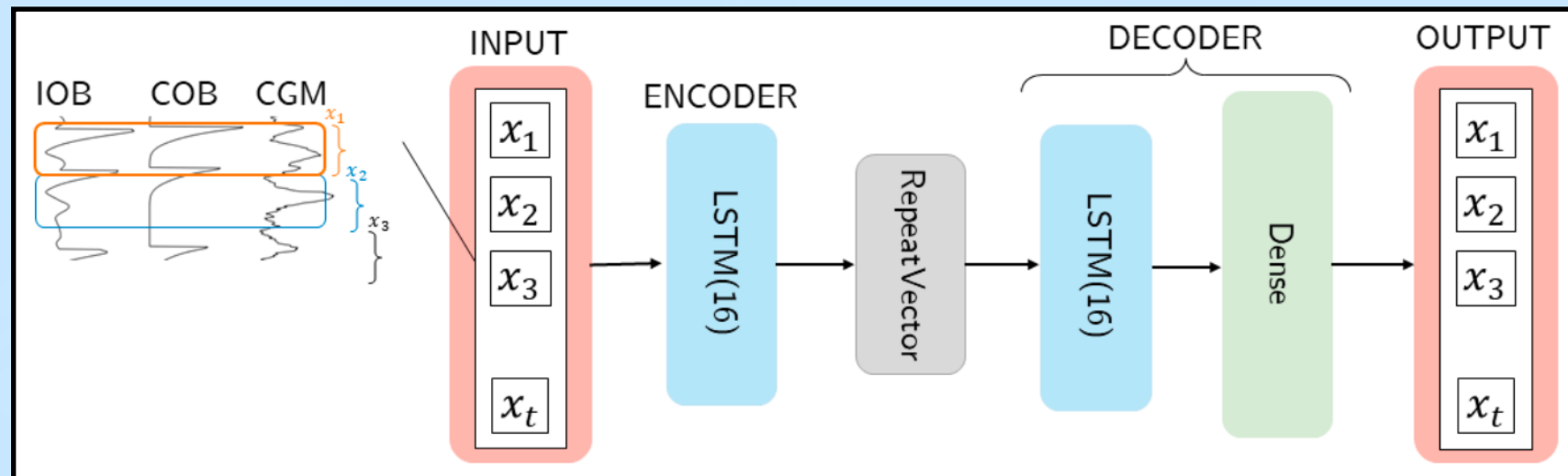
❖ Framework Overview



❖ Steps:

1. Data Preparation.
2. Feature Extraction with AE
3. Anomaly Detection with RF

AUTO-ENCODER



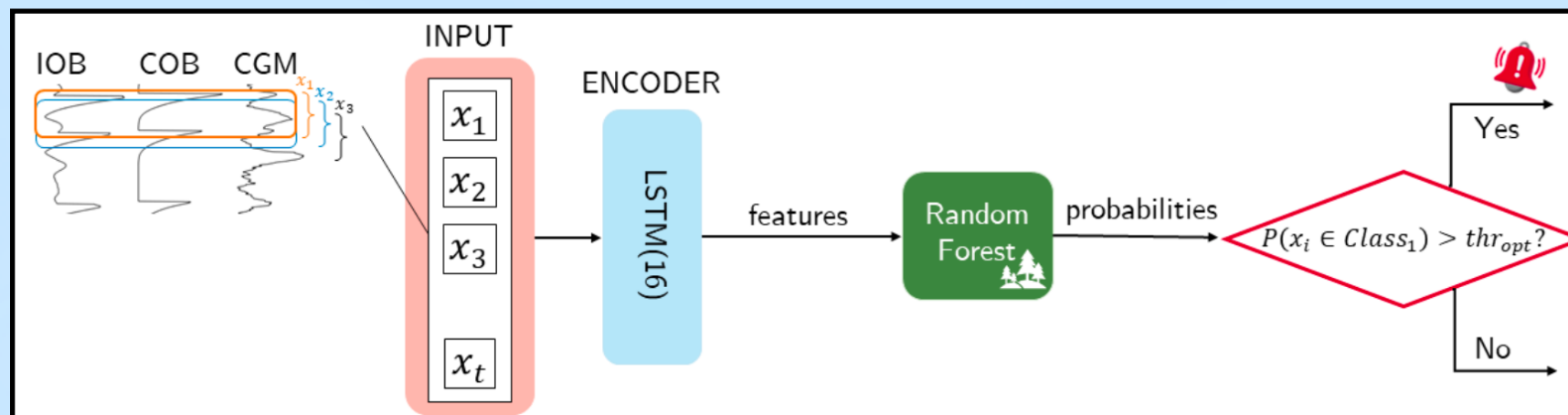
- ❖ They used an LSTM-based Auto-encoder (AE) to automatically extract meaningful, low-dimensional features from the multivariate time series.
- ❖ AE learns normal patterns of glucose dynamics without manual feature design.
- ❖ The input of the encoder part is the sequence of CGM, IOB and COB defined as $X_t^{enc} = [x_{t-L+1}, x_{t-L+2}, \dots, x_t] \in \mathbb{R}^{3 \times L}$ where each vector at time t is $x_t = [CGM(t), IOB(t), COB(k)]$

AUTO-ENCODER

❖ How it works?

- ❖ The AE is trained to reconstruct the input sequence as accurately as possible.
- ❖ The encoder compresses the input sequence into a latent representation (16 features), which captures the essential dynamics.
- ❖ After training, the decoder is discarded, and the encoder is used as a feature extractor.

ANOMALY DETECTION



- ❖ Extracted features from the encoder are fed to a Random Forest (RF) to classify each sequence as normal or faulty.
- ❖ RF outputs probabilities for both classes and raises an **alert** if the fault probability exceeds a tuned threshold.

ANOMALY DETECTION

- ❖ The optimal threshold thr_{opt} minimizes:

$$J(thr) = \sqrt{(1 - Recall(thr))^2 + (FP/day(thr))^2}$$

- ❖ If the probability $\geq thr_{opt}$, the sample is classified as faulty.
- ❖ This threshold obtained during the training phase using a simple grid search.

DATASET

- ❖ **Simulator:** UVA/Padova T1D Simulator (FDA-accepted) — generates realistic BGL responses to insulin & carbs.
- ❖ **Data:** 2 synthetic datasets:
 - ✓ 100 subjects × 3 months.
 - ✓ Meals at random times with random carbs.
 - ✓ Basal insulin: Model Predictive Controller (MPC).
 - ✓ Bolus insulin: patient-estimated carbs.
 - ✓ Measurement every 5 minutes (CGM noise modeled)
 - ✓ Dataset 2: 1 nocturnal fault/month → simulated by stopping insulin delivery (basal & bolus) completely for 6 hours at night (midnight–6AM).

DATASET

TABLE I
DATASET SPECIFICS

Metric	Data without IPF	Data with IPF
Body Weight [kg]	75.2 (12.1)	75.2 (12.1)
Age [years]	33.8 (9.6)	33.8 (9.6)
Time below range (TBR) [%]	5.6 (5.2)	5.6 (5.2)
Time in range (TIR) [%]	76.1 (9.8)	75.1 (9.7)
Time above range (TAR) [%]	18.3 (9.6)	19.2 (9.4)

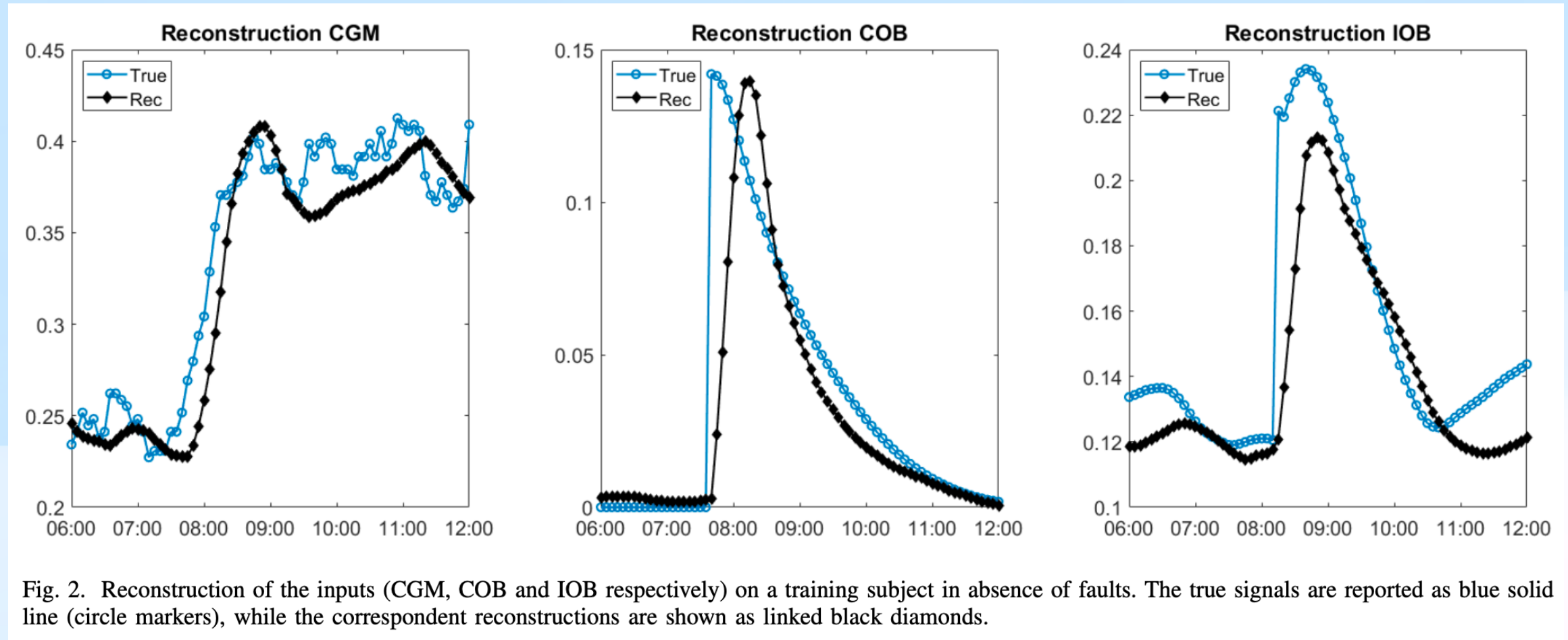
TABLE II
SUMMARY: DATASET PARTITIONING

Number of Subjects	Type of data	Step of the pipeline
Dataset1 100 subjects	No Faults	Autoencoder Training
Dataset2 80 subjects	1 IPF/month	Random Forest Training and Threshold Selection
Dataset2 20 subjects	1 IPF/month	Test of the Pipeline

EVALUATION CRITERIA

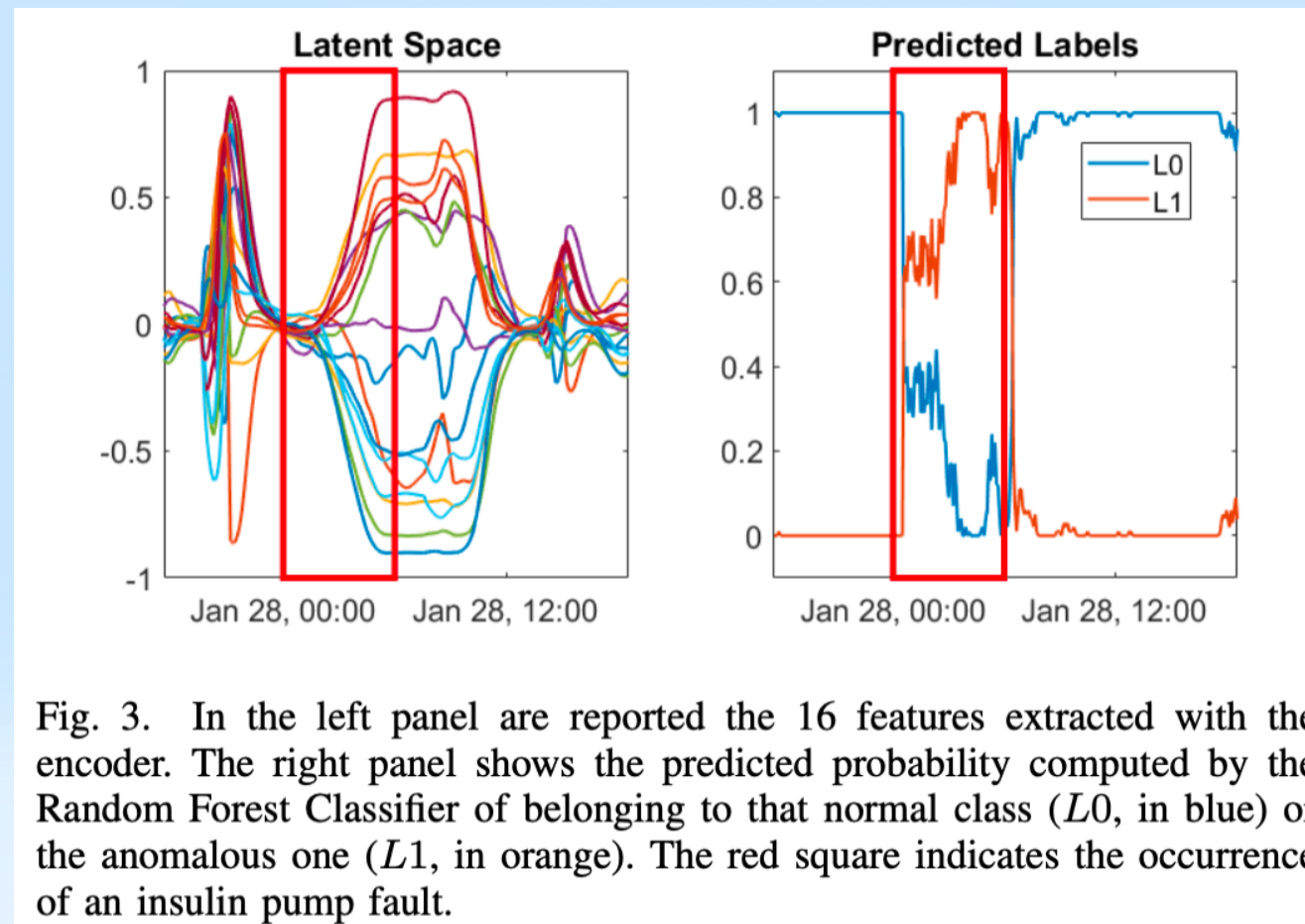
- ❖ **True Positive (TP):** Alarm during a fault.
- ❖ **False Negative (FN):** No alarm during a fault.
- ❖ **False Positive (FP):** Alarm without fault.
- ❖ **FP/day:** False positives per day.
- ❖ **Detection delay:** Time from fault start to alarm.
- ❖ **Recall (Sensitivity):** $r = \frac{TP}{TP + FN}$

RESULTS



- ❖ Figure 2 reported the output of the AE where the reconstructed inputs are shown together with the original signals during 6 monitoring hours of a subject in the training set.

RESULTS



- ❖ The **left panel** shows the 16 latent features extracted by the encoder, which diverge sharply from normal patterns during a fault.
- ❖ The **right panel** shows the Random Forest's output probabilities, where the fault probability rises and crosses the threshold, triggering an alert.

RESULTS

TABLE IV
RESULTS OF THE K-FOLD CROSS VALIDATION

Fold	Recall []	FP/day []
1	0.93 (0.17)	0.02 (0.04)
2	0.81 (0.20)	0.07 (0.10)
3	0.98 (0.07)	0.03 (0.06)
4	0.85 (0.26)	0.08 (0.10)
5	0.95 (0.22)	0.06 (0.09)
Average	0.90	0.05

TABLE V
COMPARISON WITH THE STATE-OF-ART.

Algorithm	Recall []	FP/day []	Dataset
AE-RF	0.93	0.02	Simulator v2018 [21]
Random Forest [17]	0.82	0.21	Simulator v2018 [21]
IForest [16], [17]	0.80	0.06	Simulator v2018 [21]
Manzoni et al [15]	0.91	0.12	Simulator v2018 [21]
Herrero et al [39]	0.80	0.08	Simulator v2014 [40]
Howsmon et al [10]	0.73-0.71	0.27-0.28	Real data

- ❖ They employed **5-fold cross-validation**:
- ❖ Dataset is **randomly partitioned** into 5 equally sized folds
- ❖ Each fold is used as a test set once
- ❖ The **remaining four folds** are used for training.
- ❖ This process is repeated 5 times, with each fold acting as the test set exactly once.
- ❖ **Recall**: Out of all actual faults, how many did the algorithm catch?
- ❖ Their approach is able to recognize the **90% of the IPF on average** while generating about 4 false alarms in 3 months.

RESULTS

- ❖ On average, **the algorithm detects a fault in ~220 minutes.**
- ❖ This delay is similar to clinical studies, where detection can take up to 4 hours.
- ❖ **In reality:**
 - ❖ If you have **accurate pump logs**, you can detect the fault **immediately** because the log shows “no insulin delivered.”
 - ❖ But if you only look at **BGL**, you’ll notice the effect only **~2 hours later**, because insulin already in the body keeps working for a while.

Thank you for your attention