FORESEE: ML-Driven, Communication-Efficient Time-Series Forecasting

Tayyaba Zainab[1,2], Laura Harms[2], Jens Karstens[1], Olaf Landsiedel[3,2]
[1]*GEOMAR Helmholtz Centre for Ocean Research Kiel, Germany* [2]*Kiel University, Germany*
[3]*Hamburg University of Technology, Germany*
tzainab@geomar.de, laura.harms@cs.uni-kiel.de, jkarstens@geomar.de, olaf.landsiedel@tuhh.de

***Abstract*—In the Internet of Things, a multitude of sensors continuously collect data and transmit it to the cloud for analysis. However, frequent transfer of measurements is impractical for battery-powered sensors due to the high energy-costs of wireless communication. Therefore, sensors often collect data and send it at periodic intervals, while a state-of-the-art cloud-based predictive model estimates intermediate values between transmissions. This paper introduces FORESEE, which improves data quality on the cloud without additional communication overhead compared to periodic and model-drives approaches. FORESEE makes local predictions on the sensor by employing a small, resource-efficient neural network. Upon detecting significant deviations between predicted and measured data, FORESEE communicates these measurements to the cloud. Thus, FORESEE notifies the cloud whenever predictions are difficult. On the cloud side, FORESEE uses a state-of-the-art transformer model to make predictions between transmissions. Our results demonstrate the effectiveness of FORESEE across different datasets. For instance, on the Al-Solar dataset, with a prediction length of 48, we observe a 26% improvement in the Mean Absolute Error without any additional communication, compared to periodic communication every 39 timesteps. Additionally, FORESEE achieves a 63% reduction in Mean Absolute Error compared to a model-driven approach with the same communication overhead.**

***Index Terms*—timeseries, forecasting, TinyML, Internet of Things, Edge AI, communication, LSTM, prediction**

I. INTRODUCTION

The Internet of Things (IoT) enables new applications, including traffic management [1] , energy monitoring [2] , disease tracking [3] , and disaster management [4]. In such applications, sensors continuously collect data and transmit it to the cloud for analysis. However, such sensors are often battery powered and use long-range communication technologies such as LoRaWAN [5] or NB-IoT [6]. The continuous transmission of sensor data over these technologies significantly reduces the lifetime of IoT devices due to the high energy demands of wireless communication [7].

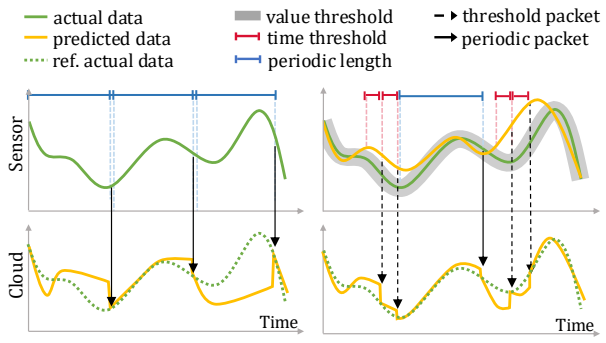
A naive approach to reducing communication costs is to collect the data on the sensor and only periodically transmit it. However, this introduces delays in data availability on the cloud. To address this, state-of-the-art (SOTA) deep neural network (DNN) models can be used on the cloud for data prediction, providing estimates for unavailable sensor data. However, our analysis of four time series datasets – Al-Solar [8], NY-TX [9], ETTm1, and ETTm2 [10] – shows that prediction quality declines significantly as the prediction horizon extends. This requires resynchronizing the model with sensor measurements roughly every 10 timesteps, depending on DNN models, datasets, and user-defined error thresholds. Moreover, a fixed periodic transmission is unable to account for sudden unpredictable changes in the recorded data. Prior work [11], [12] uses model-driven approaches, deployed on both the sensor and the cloud, to predict data trends. For example, the sensor sends an update to the cloud to resynchronize the two models when the data predicted by its model deviates from the measured data, thus being able to react to unpredictable changes. Although these models work well for simple data patterns [13], [14], they often fail to capture complex data correlations in today's real-world IoT applications, as we show in our evaluation in §IV, leading to frequent resynchronization and high communication cost.

In this paper, we introduce FORESEE[1], which builds on the design of established model-driven approaches and replaces their mathematical models with DNNs. It predicts future measurements on two levels: the sensor plane and the cloud plane, as shown in Fig. 1. On the sensor, FORESEE introduces a resource-efficient sequence-to-sequence DNN for time series prediction. If a measurement or a series of measurements deviates beyond a user-defined threshold from the prediction of the local model, FORESEE transmits all data collected since the last communication to the cloud. This resynchronizes the cloud with the sensor whenever the local DNN fails to maintain the required prediction accuracy. On the cloud side, FORESEE deploys a SOTA transformer model [9] to predict data between transmissions for measurements not transmitted by the sensor because they were considered predictable by the local model.

FORESEE improves data prediction quality on the cloud compared to periodic transmission and established model-driven approaches. For instance, on the Al-Solar dataset, periodically transmitting measurements to the cloud every 39 timesteps reduces communication cost to just 2.5% compared to transmitting at every timestep. In this setting, using a SOTA cloud-based prediction model yields an MAE of 2.15. In comparison, FORESEE has the same communication overhead for a prediction length of 48, but reduces MAE to 1.58, representing a 26% improvement in prediction accuracy on the cloud. However, compared to a model-driven approach, FORESEE improves MAE by 63%. On the NY-TX dataset, periodic transmission every 35 timesteps cuts communication

[1]Source code available at: https://github.com/ds-kiel/FORESEE

This is the author's accepted manuscript of an article published by IEEE.
The final authenticated version is available online at: https://doi.org/10.1109/DCOSS-IoT65416.2025.00024
DOI: 10.1109/DCOSS-IoT65416.2025.00024



(a) Periodic communication (b) FORESEE

Fig. 1: Periodic communication vs. FORESEE: (a) In periodic communication, data is sent at fixed intervals regardless of prediction accuracy. (b) FORESEE sends data when predictions fall outside predefined value and time thresholds, improving prediction accuracy on the cloud.

cost to 2.8% of per-timestep transmission. With the same 2.8% cost, FORESEE improves MAE by 15.8% over periodic transmission and 61% over the model-driven approach.

The paper makes the following contributions:

1) **Communication-Efficient Prediction**: FORESEE introduces a communication-efficient approach to improve data quality on the cloud by combining a lightweight sequence-to-sequence model on the sensor with a SOTA transformer model on the cloud.
2) **FORESEE Communication Strategy**: We devise an adaptive communication strategy that triggers sensor data transmission when prediction errors exceed predefined thresholds, significantly reducing cloud prediction errors compared to baselines. For instance, on the Al-Solar dataset, FORESEE reduces MAE by 26% compared to periodic communication while maintaining a similar communication overhead.
3) **Dataset Analysis**: We analyze the prediction error of SOTA DNN models and observe that predication accuracy decreases with longer prediction horizons, requiring resynchronization roughly every 10 timesteps, depending on the model, dataset, and error threshold.

Outline. We provide the required background and discuss related work in §II, followed by the design of FORESEE in §III. We evaluate FORESEE in §IV and conclude in §V.

II. Background and Related Work

In this section, we present the background on time-series prediction (TSP) and review related model and data-driven approaches in the field.

A. Time-series Prediction

Temporal forecasting, or time-series prediction (TSP) predicts future values from historical data, either for a single variable (univariate) or for multiple interrelated variables (multivariate). The goal is to accurately predict N variables with low MAE over a prediction horizon of h timesteps [9]. TSP typically employs a sequence-to-sequence model to capture temporal correlations, leveraging a window of past observations to predict future values.

B. Related Work

Existing time-series forecasting methods fall into two categories: (1) model-driven and (2) data-driven approaches.

Model-Driven Approaches: Classical data prediction and transmission reduction methods rely on analytical models, including linear [12], [15], nonlinear [16], and correlation-based models [11]. These models are particularly effective for datasets with small changes in data, especially when features are non-interrelated [14].

Notably, Raza et al. [12] propose a derivative-based prediction (DBP) model, which is particularly efficient for applications that require strict data quality guarantees. DBP determines the slope of the linear fit for a data segment by using the averages of its starting and ending points. This slope is initially calculated and subsequently re-calibrated on the sensor before being shared with the cloud. However, as we demonstrate in §IV, for modern datasets with complex and non-linear patterns, this approach yields higher MAE compared to models based on neural networks (NN). In contrast, FORESEE uses a resource-efficient NN model on the sensor that learns complex, nonlinear data patterns, requiring less frequent recalibrations.

Data-Driven Approaches: In addition to traditional model-driven methods, data-driven methods derive models directly from the data. Shallow models [17] are commonly very resource-efficient, but fail to capture complex dependencies across multiple variables [13].

Next to shallow approaches, deep neural networks (DNN) such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs) are commonly employed in earlier studies [18], [19]. More recent approaches leverage attention mechanisms to overcome the autoregressive limitations of RNNs and better capture long-term patterns [20]–[22]. For example, Informer [10] employs a general encoder-decoder transformer architecture for time-series forecasting. Other approaches [23], [24] adjust the model by incorporating classic time-series domain knowledge. Spacetimeformer [9], a SOTA model, extends these concepts to enhance temporal modeling.

While these SOTA models focus on improving the prediction accuracy. In contrast, we focus on a different problem of reducing the transmission of recorded data from sensors to the cloud.In this paper, we leverage SOTA prediction models to significantly reduce communication by avoiding the transmission of every recorded data to the cloud while maintaining near-accurate data estimates on the cloud.

III. Design

In this section, we introduce the design of FORESEE. We begin by providing an overview of FORESEE, followed by the key design challenges Next, we delve into the threshold and algorithm design, introduce resource-efficient DNN models, and describe the communication strategy employed.

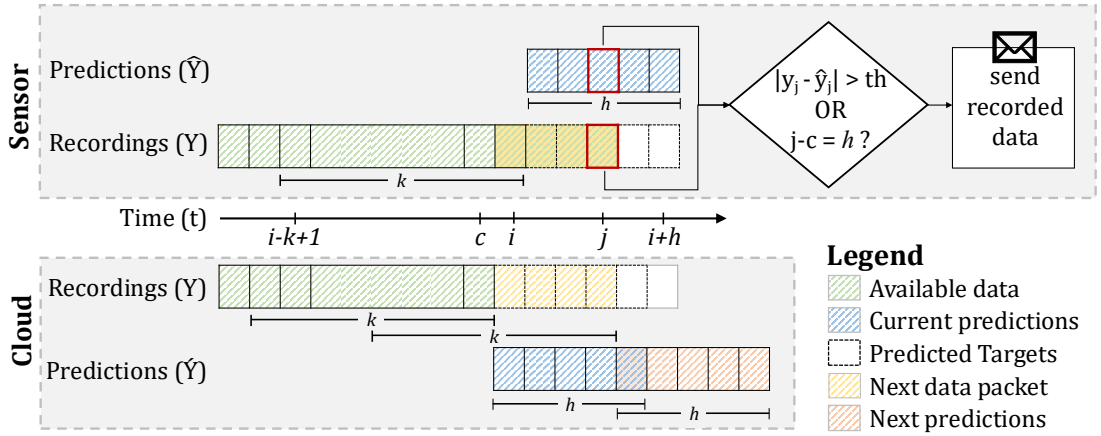


Fig. 2: Design Overview of FORESEE: The sensor records data and uses a local model to predict future values. For each new measurement, the sensor computes the prediction error. If the error exceeds some threshold th or if h timesteps have passed since the last transmission, the sensor sends the recorded data to the cloud. Upon receiving the data, the cloud updates its available data and uses k recent recordings to re-runs the model to predict future h values. Here, t_c is the time cloud received data, at t_i sensor runs its local model for predictions, and t_j is the current time.

A. Design Overview

The general idea of FORESEE is to reduce data communication costs compared to continuously transmitting measured data from the sensor to the cloud, while ensuring the availability of near-accurate data approximations on the cloud. FORESEE comprises of two primary components: the sensor and the cloud, as illustrated in Fig. 2. These components operate independently to perform time-series forecasting, each utilizing its respective DNN and computational strengths. The sensor node, equipped with a lightweight NN optimized for IoT devices, performs local predictions while minimizing computational and communication overhead. Meanwhile, the cloud employs a robust transformer-based model, for resource-intensive forecasting tasks with greater accuracy. This ensures that if the data is easy to prediction for the sensor, it will also be easy for the cloud to predict, as the cloud model has higher accuracy. Conversely, if the prediction is challenging for the cloud, it is likely difficult for the sensor, so communication is only triggered for hard-to-predict data on the sensor.

The sensor and cloud synchronize on two occasions: (1) when the prediction error exceeds a predefined threshold (th), indicating a significant decline in the sensor model's accuracy, or (2) when h timesteps elapsed since the last communication, ensuring periodic updates. For example, in Fig. 2, t_c represents such a timestep. The sensor node transmits all recorded data up to t_c, ensuring that the cloud maintains a complete dataset up to this point.

The cloud then uses the most recent k data points to generate forecasts for the next h timesteps. Between synchronizations, at any timestep t_i, when the sensor runs out of predictions, it operates independently and uses its local model to predict h timesteps based on the latest k observations. During this time, it records actual measurements and evaluates its predictions. If the next communication is triggered at timestep t_j, the sensor node transmits the recorded data from the interval $(t_c, t_j]$ to the cloud. The cloud re-runs its model using the newly received data to generate forecasts for the next h timesteps. If neither communication condition is met, the sensor node continues operating autonomously, predicting, recording, and monitoring accuracy until the current prediction window concludes. At that point, it uses the most recent k data points to predict the next h timesteps.

B. Challenges

In the design of FORESEE, we face three key challenges:

1) **Resource-Efficient On-Device DNN-Models**: The first challenge is to design resource-efficient DNN models that (a) achieve high accuracy for long prediction horizons while (b) being deployable on constrained IoT devices, such as MCUs.
2) **Communication Strategy**: The second challenge is to develop an efficient communication strategy that triggers sensor-cloud communication only for hard-to-predict data, while avoiding unnecessary transmissions due to transient mispredictions or data anomalies.
3) **Practical Prediction Length**: The third challenge is to understand the performance of SOTA time-series forecasting models, particularly for long-term predictions. This determines the maximum horizon for which predictions remain practical, i.e., below a given error.

C. Threshold Design

In FORESEE, as outlined in §III-A, we introduce thresholds th as configurable parameters to manage sensor-cloud communication These thresholds are designed to balance communication efficiency and prediction accuracy. Thresholds (th) consist of a value threshold (ε_V), which includes relative (ε_{rel}) and absolute (ε_{abs}) errors, and a time threshold (ε_T). These thresholds, unlike metrics such as mean squared error (MSE) or root

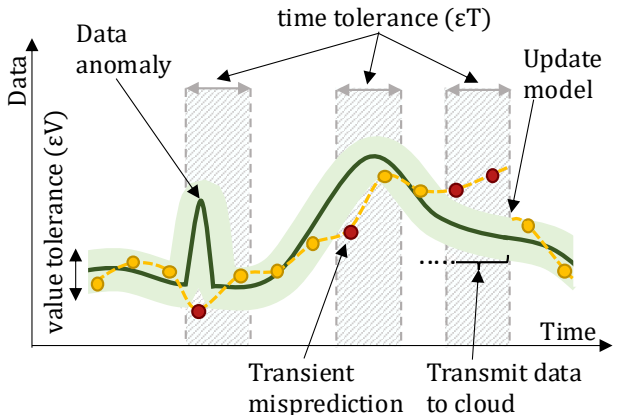


Fig. 3: The combination of value and time thresholds in FORESEE ensures robustness against prediction outliers and data anomalies, delaying data transmission to the cloud until both error bounds are exceeded.

mean squared error (RMSE), trigger data transmission to the cloud only when significant deviations in magnitude persist over time, avoiding reactions to transient mispredictions. To accommodate the diverse characteristics of datasets, particularly those with significantly varying magnitudes, we define ε_V as max(ε_{abs},ε_{rel}). Additionally, ε_T triggers only when several consecutive inaccurate predictions occur, rather than reacting to transient data anomalies and mispredictions, as shown in Fig. 3. The figure shows three scenarios: (1) when prediction aligns with the trend but is considered a misprediction due to a data anomaly, though still within ε_T; (2) transient predictions fall outside ε_V but remain within ε_T, representing temporary deviations; and (3) mispredictions persist for ε_T consecutive timesteps, leading to communication.

Choice of Thresholds: Next we discuss, the choice of practical time threshold ε_T, as it controls the frequency of transmissions and accuracy on the cloud. It is commonly observed that longer prediction horizons result in higher MAE compared to shorter horizons. For example, the MAE for a 48 prediction horizon is larger than that for prediction horizon of 3. A closer look at longer prediction horizons shows that short-term predictions are more accurate than long-term ones (see Fig. 4). The increased MAE for longer horizons is mainly due to errors accumulating over the latter portion of the forecast. For example, when forecasting 48 timesteps ahead, the initial predictions are more accurate, but accuracy progressively decreases over time. Fig. 4 shows that resynchronizing the model with sensor measurements approximately every 10 to 12 timesteps helps mitigate this accuracy drop. This trend is consistent across datasets – Al-Solar, NY-TX, and ETTm.

A smaller ε_T increases communication frequency between sensor and cloud, but improves prediction accuracy on the cloud by reducing the error introduced by long-term predictions. Conversely, a larger ε_T lowers communication overhead but compromises data accuracy. Therefore, it is important to carefully balance communication frequency and data accuracy to ensure effective and reliable cloud-based predictions. For ε_V, we choose thresholds close to the MAE of the DNN model deployed on the sensor to align with its expected performance, and evaluate these in §IV.

Multivariate Datasets: For multivariate datasets like ETTm1 and ETTm2, deviations can be addressed by aggregating errors across features, applying weighted thresholds based on feature importance, or defining separate thresholds for each feature. To simplify implementation, FORESEE triggers communication if any individual feature exceeds both ε_V and ε_T, ensuring sensitivity to deviations.

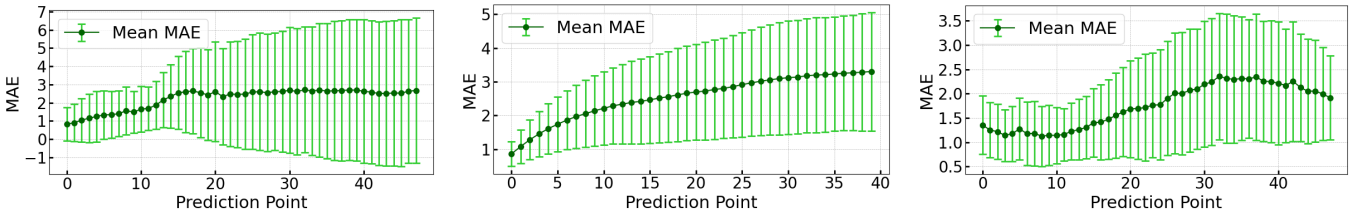
Algorithm 1 Sensor Data Prediction and Transmission

1: Let y_a represents recorded data at time a
2: Let $\hat{y}_a$ represents predicted data for time a
3: Let y_a^b represents recorded data for time a to b
4: Let $\hat{y}_a^b$ represents predicted data for time a to b
5: Let ε_V and ε_T be the value and time threshold
6: Let $count$ be the counter for consecutive mispredictions initialized to 0
7: Let $buffer$ store pending data for transmission
8: Let k be the context window
9: Let h be the prediction window
10: Let t_i be the current timestep
11: **while** true **do** ▷ Loop forever
12: $\hat{y}_{i+1}^{i+h} \leftarrow \text{Model}_s(y_{i-k+1}^{i})$
13: $z \leftarrow i + h$
14: **while** $i \leq z$ **do** ▷ Loop h measurements
15: Wait for *measurement*
16: $i \leftarrow i + 1$
17: $y_i \leftarrow measurement$
18: $\varepsilon \leftarrow |y_i - \hat{y}_i|$
19: Append y_i to $buffer$
20: **if** $\varepsilon \leq \varepsilon_V$ **then**
21: $count \leftarrow 0$
22: **else**
23: $count \leftarrow count + 1$
24: **if** $count = \varepsilon_T$ **or** $len(buffer) = h$ **then**
25: Send $buffer$ to cloud
26: $count \leftarrow 0$
27: $buffer \leftarrow \emptyset$ ▷ Clear the buffer
28: **end if**
29: **end if** ▷ Increment timestep
30: **end while**
31: **end while**

D. Algorithm Design

Algorithm 1 formalizes the design of FORESEE on the sensor plane. For simplicity and without loss of generality, we consider only univariate data. Upon NN inference, each new recording is stored in a buffer (line 19). Simultaneously, we track whether the prediction is out of bound (line 20). When it does, we increment the counter (line 23); otherwise, we reset it for transient mispredictions (line 21), counting only consecutive errors. Once the buffer length reaches h or the misprediction counter exceeds ε_T, we transmit the buffer data to the cloud and reset the counter. For multivariate data, we track the counter for each variable. If any variable's counter reaches ε_T or the buffer length reaches h, we transmit all the data and reset all counters to 0. The cloud side is straightforward. Whenever the cloud receives data from the sensor, it uses the most recent data to make predictions for the next h timesteps.

It should be noted that the sensor predicts periodically every h timesteps to save computational resources. On the cloud, however, we predict as soon as data is received from the



(a) Al-Solar (b) NY-TX (c) ETTm1

Fig. 4: Mean MAE with standard deviation error for each prediction point. The x-axis represents the prediction points, each point along the x-axis signifies a step in the prediction sequence, with earlier points corresponding to near-term forecasts and later points to long-term forecasts for a prediction length. The plot shows that near-term predictions are more accurate than those further out, with Mean MAE increasing over time.

sensor. This approach minimizes the error caused by long prediction horizons while it optimizes computation overhead on the sensor.

E. NN models

Sensor Plane: We design two resource-efficient DNN models, denoted BiLSTM and LSTNet$_{LSTM}$, for on sensor deployment, i.e., targeting of the shelf MCUs. We later compare these models with SOTA data- and model-driven approaches to select the best model for a given dataset.

LSTNet$_{LSTM}$ is inspired by LSTNet [8]. It replaces GRUs with two LSTM layers, each containing 64 hidden units, separated by a dropout layer with a 0.2 dropout rate to prevent overfitting. The model concludes with a linear output layer. While LSTMs are known for their ability to capture long-term dependencies [25], they only capture dependencies in a single direction, which can limit their ability to fully understand the sequential nature of time-series data. Thus, in addition, we introduce a BiLSTM-based variant that captures dependencies in both forward and backward direction. This model begins with a convolutional layer with one input channel and 64 output channels, followed by a dropout layer with a 0.2 dropout rate. It then stacks two bidirectional LSTM (BiLSTM) layers, each with 64 hidden units, and ends with a linear layer.

Cloud Plane: On the cloud side, we are not limited by the resource constraints of MCUs and thus deploy a large transformer model. In this paper, we use the Spacetimeformer [9] model, but the design of FORESEE is general and not bound to a specific model.

F. Communication Model

To evaluate FORESEE and compare it to baselines, we next introduce a communication and a data compression model.

Wireless Communication: For the communication between the sensor and the cloud, we assume the use of LoRaWAN, e.g., using the common Semtech SX1276 [26] radio. To model the communication, we assume settings that achieve a balance between data rate and range, including a medium spreading factor (SF9), a bandwidth of 125 kHz, and the 4/5 coding rate. This communication setting allows a maximum payload of 115 bytes. Sending a single value, for example, up to 4 bytes, takes 164.9 ms, including protocol headers. Each additional block of 4 to 5 bytes takes an additional 20.4 ms. Transmitting a single measurement of 1 byte (164.9 ms) has a significant overhead compared to sending aggregated data of 10 bytes (205.8 ms) every 10th measurement due to not having to turn on the radio each time and not doing carrier sense each time. In addition, we only need one header and can apply compression.

Compression: We assume a gzip compression [27] and have experimentally confirmed a compression rate of about 50%. Thus, for example, compressing 48 measurements from the Al-Solar dataset, each consisting of a single floating-point value, takes 96 bytes after compression.

G. Datasets

This paper uses four benchmark datasets: ETTm1, ETTm2, NY-TX Weather, and Al-Solar. All are established datasets for time-series predictions.

ETTm1 and ETTm2 [10]: The Electricity Transformer Temperature (ETT) is a common benchmark dataset of electric power deployment, collected over two years from two counties in China. Each data point includes the target value 'oil temperature' and six power load features, totaling $N = 7$ features. We use the 15-minute interval datasets, ETTm$_1$ and ETTm$_2$, for our evaluation.

NY-TX Weather [9]: This dataset consists of hourly temperature readings from the ASOS [28] weather network. The weather network contains many stations, whereas the NY-TX Weather dataset limits itself to three airport stations in central Texas (ACT, ABI, and AMA) and three in New York (ALB, LGA, and JFK). The data covers the years 1949-2021, making this a very large dataset.

Al-Solar [8]: This dataset comprises solar power production measurements from 137 photovoltaic power plants in Alabama during 2006, sampled every 10 minutes. We evaluate only five randomly selected locations: 3, 8, 14, 19, and 22.

IV. EVALUATION

In this section, we evaluate the performance of FORESEE. We begin by introducing the evaluation setup and our baselines, followed by an assessment of sensor models using Mean Absolute Error (MAE) and parameter count. Finally, we conclude with an overall system evaluation covering prediction error, communication cost, and power consumption.

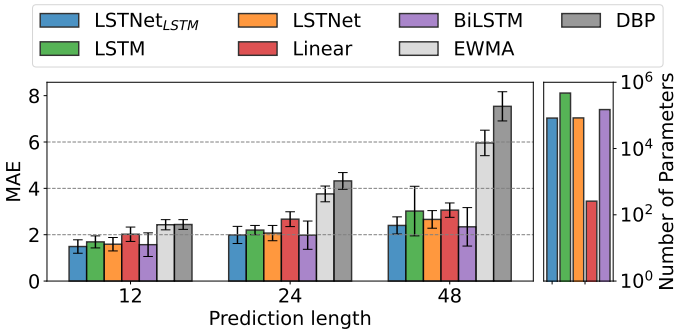


Fig. 5: Performance comparison of sensor models on Al-Solar dataset. The plot shows the mean and std of MAE for prediction lengths of 12, 24, and 48, corresponding to 2, 4, and 8 hours of prediction data respectively. Evaluations are conducted across five randomly selected locations. The subplot on the right shows the parameter count for each NN model.

A. Evaluation Setup

For our evaluation, we use the following setup.

Models: For on-sensor deployment, we compare our BiLSTM and LSTNet$_{LSTM}$ models with SOTA models, including LSTM (encoder-decoder) [25] and LSTNet (GRUs) [8]. We also evaluate a Linear Regression model as a baseline [9]. Additional baselines include two conventional approaches: the Exponentially Weighted Moving Average (EWMA) [29], which emphasizes recent observations, and the DBP model [12]. We select the best-performing sensor model from this comparison and pair it with Spacetimeformer on the cloud.

Prediction lengths: To evaluate all these sensor models, we use a context window (history) of 128, 288, 288 and 600 measurements for Al-Solar, ETTm1, ETTm2, and NY-TX dataset, respectively. We evaluate sequence lengths of 12, 24, and 48 predictions for Al-Solar, ETTm1, and ETTm2. For NY-TX, we use sequence lengths of 10, 20, and 40 predictions. For DBP, we calculate the slope of the context window using data from its first and last quarters.

Metrics: We select a model for the sensor based on lower mean absolute error (MAE) and the number of parameters, ensuring its deployability on resource-constrained devices. However, to evaluate FORESEE and its baselines, we consider the number of communication rounds, where each round corresponds to a complete communication instance from the sensor to the cloud, which may consist of one or more packets, along with the corresponding MAE on the cloud.

B. Baselines

For our baselines, we consider two approaches: (1) Periodic transmission, where measurements are sent to the cloud every h timesteps, with h as the prediction length. We use Spacetimeformer to predict the data between transmissions. This reduces communication rounds compared to transmitting data at every timestep but introduces an MAE overhead. This also sets a lower bound on communication rounds. (2) A DBP-based approximate model running on both the sensor and the cloud. The sensor transmits data only when predictions deviate from actual recordings, allowing the sensor and cloud to resynchronize their data and deltas.

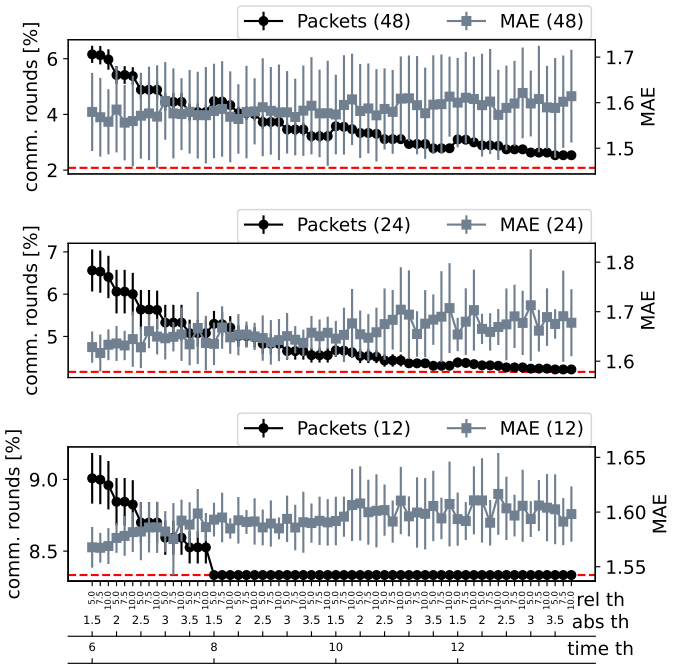
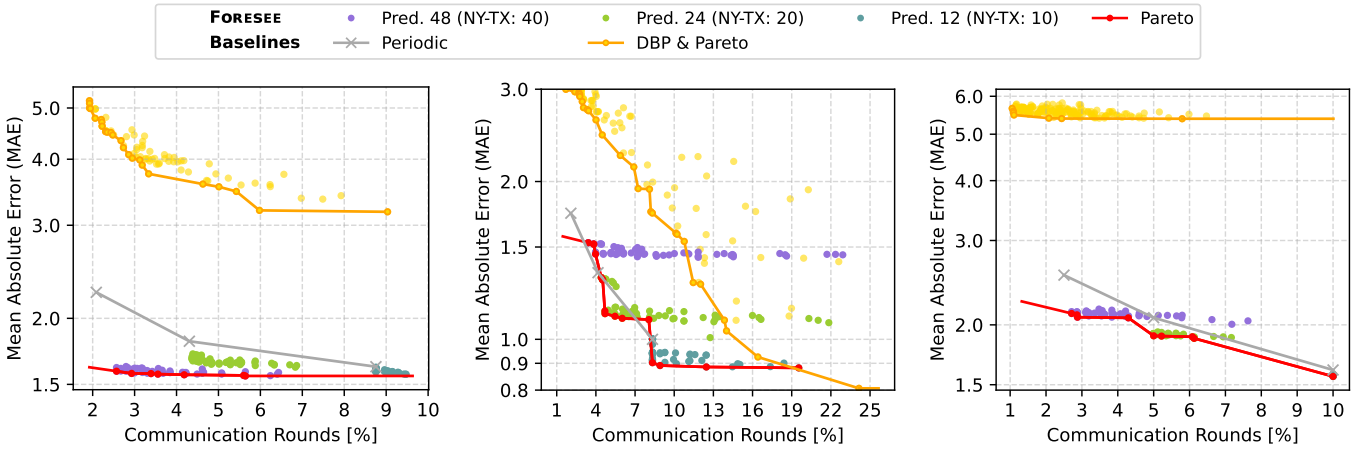


Fig. 6: Evaluating MAE and relative communication rounds compared to sending data at every time step for different threshold values: absolute error (abs), relative error (rel), and time threshold (time th). The red lines indicate the lower bound of communication rounds when communicating only once per prediction interval. The results show that increasing the prediction window (horizon) leads to greater improvements in both MAE and communication efficiency.

C. Evaluating FORESEE's On-Sensor Models

Before evaluating FORESEE, we first select the best model for on-sensor deployment. To do this, we evaluate all models across all prediction lengths mentioned in §IV-A to identify the best model for a given dataset on resource-constrained devices. Fig. 5 shows a performance comparison on Al-Solar dataset across prediction lengths of 12, 24, and 48, summarizing the mean and standard deviation (std) of MAE for all sensor models. Additionally, we compare the number of parameters for each model, highlighting the trade-offs between MAE and model complexity. For our system-wide evaluation (§ IV-D), we select LSTNet$_{LSTM}$ for this dataset, as it has a similar parameter count to LSTNet but achieves better MAE.

Similarly, for the NY-TX weather dataset with prediction lengths of 10, 20, and 40, we select LSTM for its lower mean MAE (1.88–2.78) and fewer parameters than other models. For the ETTm1 and ETTm2 datasets with prediction lengths of 12, 24, and 48, we select the Linear model as it achieves the lowest prediction error.



(a) Al-Solar (b) ETTm1 (c) NY-TX

Fig. 7: FORESEE vs. baselines: We compare the performance of FORESEE against baselines, with the x-axis showing relative communication rounds compared to sending data every timestep and the y-axis representing the resulting MAE on the cloud. FORESEE outperforms DBP across all datasets. It achieves major gains over periodic transmission in Al-Solar and NY-TX, demonstrating better adaptability to data dynamics.

TABLE I: Absolute, relative and time threshold configuration ranges for FORESEE and DBP. From these ranges, we select a subset tailored to each dataset and prediction length.

	FORESEE	DBP
Absolute threshold	0.5 - 4.5	0 - 7
Relative threshold (%)	5, 7.5, 10	5,7.5,10
Time threshold	4 - 12	1 - 10

D. FORESEE *vs. Baselines*

This section evaluates FORESEE on all four datasets using different time and value thresholds. However, due to space constraints we only show results for Al-Solar in Fig. 6. This figure contains three vertically stacked plots, corresponding to different prediction lengths indicated in the legend. Each plot includes three x-axes: relative threshold, absolute threshold, and time threshold (see TABLE I for threshold ranges used across datasets. For every combination of these thresholds, the plots display two metrics on the y-axis: the relative number of communication rounds (compared to sending every measurement immediately) for that particular combination, and the resulting MAE on the cloud. The red dashed lines in the figure represent the lower bounds for communication rounds which corresponds to periodic sending.

Al-Solar: We evaluate FORESEE using the Al-Solar dataset by plotting the mean and standard deviation for both the percentage of communication rounds and the MAE across 5 random locations (see Fig. 6). To compare our approach with baseline methods, we include this data in Fig. 7a, alongside results from our periodic and DBP baselines. The figure illustrates that a prediction length of 48 achieves the most significant improvement in MAE on the cloud, relative to communication rounds from the sensor. For this dataset, the optimal thresholds with a prediction length of 48 are 3.5 for absolute error, 5% for relative error, and 12 for time. This reduces MAE by 26% while maintaining the same communication overhead as periodic transmissions every 39 timesteps. This communication amounts to just 2.5% of per-timestep transmission. Compared to DBP, FORESEE achieves a 63% reduction in MAE with similar communication overhead.

ETTm1 and ETTm2: For the ETTm datasets, we present results only for ETTm1 in Fig. 7b due to the similarity between both datasets. While the results show an improvement over DBP, the improvement over periodic baselines remains marginal due to the limited potential for further accuracy gain. This suggests that periodic data transmission is sufficient for such datasets.

NY-TX: For NY-TX, we evaluate FORESEE for prediction lengths of 10, 20 and 40. Similar to Al-Solar, Fig. 7c demonstrates that a prediction length of 40 provides the most significant MAE improvement compared to periodic. We achieve a 15.8% reduction in MAE by using an absolute threshold of 4.5, a relative threshold of 7.5%, and a time threshold of 12, compared to sending data to the cloud periodically after every 35 predictions. This results in just 2.8% of the communication rounds required for sending packets at every timestep. With the same number of communication rounds, we improve MAE by 61% compared to DBP.

These results highlight that carefully selecting error and time thresholds improves MAE while maintaining the same amount of communication rounds compared to periodic data transmission. While periodic transmission provides a simple and consistent approach, it lacks adaptability to data dynamics, leading to suboptimal performance. DBP performs worse than periodic transmission, failing to capture dataset trends and making it unsuitable for complex datasets.

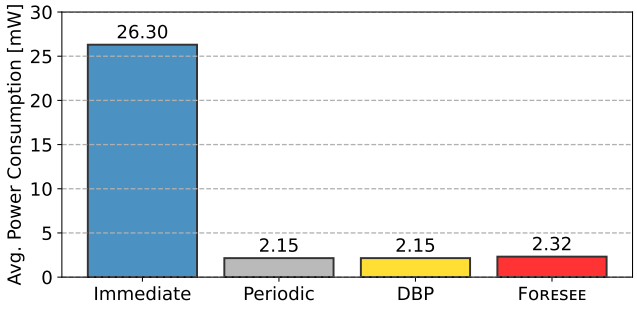


Fig. 8: Average power consumption of FORESEE and baselines: We compare communication and local processing energy for Al-Solar with 2.5% communication rounds. Communication for periodic, DBP, and FORESEE: 2.15mW; Computation overhead for DBP: <0.003mW; Inference overhead for FORESEE: 0.17mW. Immediate refers to sending data upon recording, consuming 26.30mW for communication.

E. Power Consumption

Finally, we evaluate the average power consumption of FORESEE in comparison to two baselines: periodic communication and DBP (see Fig. 8). Additionally, we consider the traditional approach of transmitting each sample immediately after recording. Using the Al-Solar dataset, we configure FORESEE with the settings that achieve the highest MAE reduction of 26% compared to periodic communication with 2.5% communication rounds. Periodic communication, DBP, and FORESEE consume an average of 2.15mW for communication, whereas transmitting every sample immediately results in over 12 times higher power consumption. Beyond communication cost, FORESEE incurs an additional 0.17mW (8%) overhead for inference and computation on an nRF5340 microcontroller [30], while DBP's computational overhead is negligible. The inference overhead stems from a single inference every 48 timesteps (cf. §III-D). At each other timestep, the sensor merely compares the measured value to the predicted one and updates the out of bound tracking, ensuring the sensor to sleep until taking the next measurement.

Periodic communication constrained to the same energy budget as FORESEE (2.32mW) allows for more frequent communication rounds of 3.3% resulting in an MAE of 1.98. However, despite more frequent periodic communication, FORESEE still achieves a 20% lower MAE.

V. CONCLUSION

In this paper, we introduce FORESEE, designed to enhance cloud data quality over periodic transmission while maintaining an equivalent communication overhead. FORESEE integrates a transformer-based prediction model on the cloud with a lightweight sequence-to-sequence prediction model on the sensor. The sensor triggers communication only when prediction errors exceed predefined thresholds, ensuring data is transmitted only for hard-to-predict instances. This reduces cloud prediction error compared to periodic transmission, which, unlike FORESEE, cannot adapt to sudden changes in the recorded data. Our results show that FORESEE improves MAE by 26% on the Al-Solar dataset while maintaining the same communication overhead as periodic transmission. Compared to model-driven approaches, FORESEE improves MAE by 63% at the same communication cost.

ACKNOWLEDGMENT

We thank our anonymous reviewers for their insight and suggestions. This work has received funding from the Helmholtz School for Marine Data Science (MarDATA) [Grant No. HIDSS-0005].

REFERENCES

[1] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *arXiv preprint arXiv:1707.01926*, 2017.

[2] Y. Pang, B. Yao, X. Zhou, Y. Zhang, Y. Xu, and Z. Tan, "Hierarchical electricity time series forecasting for integrating consumption patterns analysis and aggregation consistency.," in *IJCAI*, pp. 3506–3512, 2018.

[3] Y. Matsubara, Y. Sakurai, W. G. Van Panhuis, and C. Faloutsos, "Funnel: automatic mining of spatially coevolving epidemics," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 105–114, 2014.

[4] T. Zainab, J. Karstens, and O. Landsiedel, "Lighteq: On-device earthquake detection with embedded machine learning," in *Proceedings of the 8th ACM/IEEE Conference on Internet of Things Design and Implementation*, pp. 130–143, 2023.

[5] LoRa Alliance, *LoRaWAN 1.1 Specification*, 2017.

[6] Y.-P. E. Wang, X. Lin, A. Adhikary, A. Grovlen, Y. Sui, Y. Blankenship, J. Bergman, and H. S. Razaghi, "A primer on 3GPP narrowband Internet of Things," *IEEE communications magazine*, vol. 55, no. 3, 2017.

[7] U. Raza, A. Camerra, A. L. Murphy, T. Palpanas, and G. P. Picco, "What does model-driven data acquisition really achieve in wireless sensor networks?," in *2012 IEEE International Conference on Pervasive Computing and Communications*, pp. 85–94, IEEE, 2012.

[8] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *The 41st international ACM SIGIR conference on research & development in information retrieval*, pp. 95–104, 2018.

[9] J. Grigsby, Z. Wang, N. Nguyen, and Y. Qi, "Long-range transformers for dynamic spatiotemporal forecasting," *arXiv preprint arXiv:2109.12218*, 2021.

[10] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, pp. 11106–11115, 2021.

[11] D. Tulone and S. Madden, "An energy-efficient querying framework in sensor networks for detecting node similarities," in *Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems*, pp. 191–300, 2006.

[12] U. Raza, A. Camerra, A. L. Murphy, T. Palpanas, and G. P. Picco, "Practical data prediction for real-world wireless sensor networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 8, pp. 2231–2244, 2015.

[13] J. Yin, W. Rao, M. Yuan, J. Zeng, K. Zhao, C. Zhang, J. Li, and Q. Zhao, "Experimental study of multivariate time series forecasting models," in *Proceedings of the 28th ACM international conference on information and knowledge management*, pp. 2833–2839, 2019.

[14] N. Q. V. Hung, H. Jeung, and K. Aberer, "An evaluation of model-based approaches to sensor data compression," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2434–2447, 2012.

[15] G. Wei, Y. Ling, B. Guo, B. Xiao, and A. V. Vasilakos, "Prediction-based data aggregation in wireless sensor networks: Combining grey model and kalman filter," *Computer Communications*, vol. 34, no. 6, pp. 793–802, 2011.

[16] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong, "Approximate data collection in sensor networks using probabilistic models," in *22nd International Conference on Data Engineering (ICDE'06)*, IEEE, 2006.

[17] D. Tulone and S. Madden, "Paq: Time series forecasting for approximate query answering in sensor networks," in *European Workshop on Wireless Sensor Networks*, pp. 21–37, Springer, 2006.

[18] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *arXiv preprint arXiv:1707.01926*, 2017.

[19] R. Yu, S. Zheng, A. Anandkumar, and Y. Yue, "Long-term forecasting using tensor-train rnns," 2018.

[20] T. Iwata and A. Kumagai, "Few-shot learning for time-series forecasting," *arXiv preprint arXiv:2009.14379*, 2020.

[21] S. Wu, X. Xiao, Q. Ding, P. Zhao, Y. Wei, and J. Huang, "Adversarial sparse transformer for time series forecasting," *Advances in neural information processing systems*, vol. 33, pp. 17105–17115, 2020.

[22] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," *Advances in neural information processing systems*, vol. 32, 2019.

[23] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, and S. Dustdar, "Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting," in *International conference on learning representations*, 2021.

[24] K. Madhusudhanan, J. Burchert, N. Duong-Trung, S. Born, and L. Schmidt-Thieme, "Yformer: U-net inspired transformer architecture for far horizon time series forecasting," *arXiv preprint arXiv:2110.08255*, 2021.

[25] A. Graves and A. Graves, "Long short-term memory," *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

[26] Semtech Corporation, *SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver*, 2020. Rev. 7.

[27] L. P. Deutsch, "GZIP file format specification version 4.3." RFC 1952, May 1996.

[28] NOAA, "National weather service automated surface observing system (asos)," 2021.

[29] D. R. Cox, "Prediction by exponentially weighted moving averages and related methods," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 23, no. 2, pp. 414–422, 1961.

[30] Nordic Semiconductor, *nRF5340*, 2024. Product Specification v1.5.