Predict First, Sync When Needed: A Risk-Aware Approach to Adaptive Digital Twin Synchronization under Bandwidth Constraints

Yasith R. Wanigarathna*, Sasu Tarkoma*, and Roberto Morabito*†
*Department of Computer Science, University of Helsinki, Finland
†Department of Communication Systems, EURECOM, France
{yasith.wanigarathnaarachchige, sasu.tarkoma}@helsinki.fi, Roberto.Morabito@eurecom.fr

Abstract*—Maintaining real-time synchronization between physical assets and their Digital Twins over bandwidth-limited wireless links is a complex challenge. Existing approaches either transmit all sensed data, resulting in unnecessary network overhead, or rely solely on forecasting, which cannot guarantee fidelity when operating conditions change. In response, we present a risk-aware, event-driven synchronization framework that leverages predictive modeling to minimize network traffic while maintaining the fidelity of digital twins. Our approach utilizes a Gated Recurrent Unit-based quantile forecaster to predict key states with calibrated uncertainty. Additionally, it employs online quantile coverage and Kullback-Leibler divergence tests to monitor reliability and detect changes in distribution. The system operates in a predictive mode, fetching only selected states, and shifts to full synchronization and model retraining when deviations are detected. We conducted experiments using public LTE and 5G datasets, achieving $R^2 > 0.80$ in most cases, with a peak of 0.99. Real-world evaluations using a commercial smartphone showed network-traffic reductions of 5.4–7.0% versus continuous synchronization, along with mean absolute error (MAE) reductions of about 64% on average relative to simpler predictive methods. These findings demonstrate that predictive synchronization, coupled with awareness of drift, enables high-fidelity digital twinning even under strict bandwidth constraints.

***Index Terms*—Digital Twin, IoT, Adaptive Synchronization, Predictive Modeling, 5G Networks, Resource Efficiency**

I. Introduction

A Digital Twin (DT) is a virtual representation of a physical system that continuously reflects its state and behavior in real time [1]. Unlike static models, a DT maintains a bidirectional link with its Physical Twin (PT), receiving sensor data and sending back predictive insights or control signals [2]. This closed-loop interaction enables enhanced capabilities in the physical systems where DTs are utilized in various domains such as networking, manufacturing, and healthcare [3].

The emergence of the 6G paradigm envisions massive digital twinning, where DTs evolve from individual asset replicas to large-scale, system-level models that provide comprehensive visibility and coordinated control [4]. These deployments may involve thousands of interconnected DTs performing continuous monitoring, predictive analytics, and adaptive resource management. Achieving real-time synchronization on this scale remains a significant challenge [5].

This work was supported by the Digital Twinning of Personal Area Networks for Optimized Sensing and Communication project through the Business Finland 6G Bridge Program (8782/31/2022).

Most existing works assume a consistent, high-fidelity connection between PT and DT. However, such continuous synchronization is often impractical over bandwidth-limited wireless links. The frequency of synchronization introduces a crucial trade-off: frequent updates improve accuracy but increase communication overhead, while infrequent updates risk state divergence [6]. This motivates the need for adaptive approaches that intelligently balance fidelity with bandwidth consumption [7].

Beyond immediate state mismatches, delayed or missing updates can introduce concept drift, where the DT's internal models become misaligned with the evolving PT [8]. As the underlying data distribution changes, model performance degrades, limiting the effectiveness of analytics and control. Therefore, incorporating uncertainty estimation and drift detection is essential to maintain DT fidelity, especially in predictive or event-driven synchronization schemes. Rising predictive uncertainty is often correlated with distribution shifts, which can trigger uncertainty-driven alerts for drift detection [9]. Early detection of such deviations enables corrective actions, including retraining and recalibration, before errors accumulate.

To address these challenges, we propose a risk-aware, event-driven synchronization framework that integrates predictive modeling with uncertainty quantification and distributional drift detection. The DT utilizes a multi-output quantile-based time series forecaster to predict multiple PT states with uncertainty estimates. Simultaneously, Kullback–Leibler divergence is employed to detect distribution shifts by comparing current prediction distributions against a reference baseline. When predictive uncertainty increases or divergence exceeds a threshold, the system switches from predictive operation to direct synchronization and retrains the model with fresh data. This adaptive control loop reduces unnecessary transmissions while maintaining high fidelity under changing conditions.

We evaluate our framework using real-world LTE and 5G datasets and implement it on a commercial smartphone. The results demonstrate that our approach significantly reduces synchronization traffic compared to direct synchronization while preserving state accuracy over long durations. These enhancements arise from using uncertainty-aware forecasting and drift detection together, demonstrating that predictive synchronization can maintain high-quality digital twins even under strict bandwidth limitations, offering a promising direction for future DT implementations in 6G and beyond.

This paper makes the following key contributions. We introduce a risk-aware synchronization framework that *(i)* combines quantile-based uncertainty estimation with KL-divergence drift detection to decide when to synchronize vs. predict, *(ii)* performs event-driven retraining using synchronized data only when prediction reliability degrades, and *(iii)* is implemented and evaluated on a real smartphone DT using LTE and 5G datasets and live network traffic. The framework achieves about 64% MAE reduction on average over simpler predictive schemes while reducing synchronization traffic by 5–7% in real deployments.

II. Related Work

Existing research on DT synchronization has investigated various strategies to reduce communication overhead while maintaining fidelity. One approach focuses on event-driven or threshold-based synchronization, where updates are sent only when the deviation between the DT and its PT exceeds a predefined error bound. For instance, TwinSync [10] employs a lightweight shadow twin mechanism that transmits state updates only when a user-defined error threshold is about to be violated, significantly reducing bandwidth consumption without compromising application-level accuracy. Tan and Matta [7] frame synchronization as a stochastic control problem, demonstrating that adaptive policies that consider the system state outperform fixed-interval update schemes, emphasizing the importance of dynamic synchronization decisions.

The trade-off between synchronization accuracy and communication cost has also been studied in mobile and vehicular DT systems. Incentive-based and game-theoretic approaches have been proposed to coordinate resource sharing between vehicles and edge infrastructure under bandwidth constraints [11]. Other efforts address networking bottlenecks by modeling synchronization delays and optimizing communication parameters. For example, Cakir et al. [12] introduce the Twin Alignment Ratio to quantify synchronization timeliness and show how congestion control is essential to prevent performance degradation in dense wireless environments. Similarly, Yang et al. [13] frame synchronization-delay minimization as a joint optimization problem involving device selection, transmit power allocation, and data offloading, resulting in significant delay reductions compared to heuristic approaches. Forecasting-based synchronization schemes have also been explored, where sensor values are predicted locally, and the update frequency is adjusted based on error thresholds [14]. While these solutions improve scalability and efficiency, they generally lack mechanisms to quantify predictive uncertainty or detect when distributional shifts render predictions unreliable.

Quantile regression has emerged as a powerful tool for modeling predictive uncertainty. Unlike conventional point forecasting, it estimates conditional quantiles and produces prediction intervals with defined coverage probabilities [15]. These intervals are robust to outliers and heteroscedasticity, common in wireless telemetry data. Recurrent architectures such as LSTMs and GRUs enhance this approach by capturing nonlinear temporal dependencies, yielding well-calibrated uncertainty bands that reduce false alarms and improve decision-making in time-series prediction [16]. Additionally, tail quantile analysis has been used to indicate concept drift [17].Furthermore, composite quantile regression enables online updates without costly retraining by jointly estimating multiple quantiles using a single objective function [18]. These advances demonstrate that quantile-based forecasting can inform risk-aware decision-making. However, prior work has yet to combine these methods with adaptive DT synchronization.

Detecting distributional drift is another essential capability for maintaining DT fidelity. Metrics such as the Kullback–Leibler (KL) divergence offer a principled measure of divergence between observed and expected data distributions and have been widely used in anomaly detection, streaming analytics, and adaptive learning [19]. KL-based statistics, like the Population Stability Index (PSI), can detect subtle changes in data distributions before conventional systems fail [20]. Simultaneously, applications in wireless sensing and

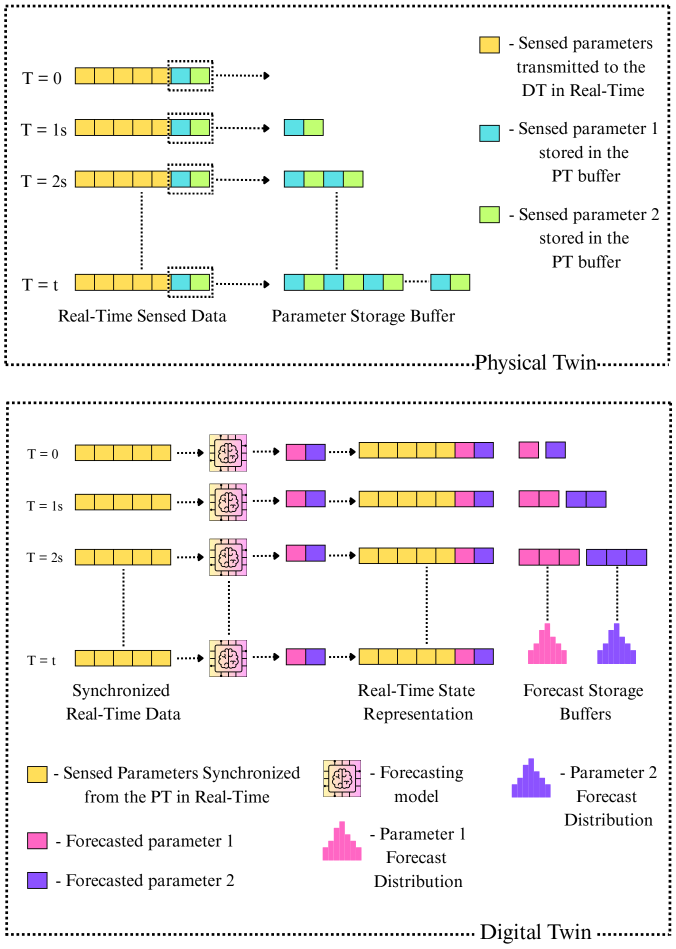
T = 0
T = 1s
T = 2s
T = t
Real-Time Sensed Data
Parameter Storage Buffer
- Sensed parameters transmitted to the DT in Real-Time
- Sensed parameter 1 stored in the PT buffer
- Sensed parameter 2 stored in the PT buffer
Physical Twin
T = 0
T = 1s
T = 2s
T = t
Synchronized Real-Time Data
Real-Time State Representation
Forecast Storage Buffers
- Sensed Parameters Synchronized from the PT in Real-Time
- Forecasting model
- Parameter 2 Forecast Distribution
- Forecasted parameter 1
- Parameter 1 Forecast Distribution
- Forecasted parameter 2
Digital Twin

Fig. 1. In adaptive mode, the PT synchronizes only a subset of parameters and buffers the rest. The DT forecasts the withheld parameters to reconstruct the full state. When prediction quality decreases, the framework triggers full synchronization at the next interval.

interference prediction demonstrate the effectiveness of these metrics in triggering model switching and enhancing prediction robustness [21]. KL divergence has also been applied in integrated sensing and communication systems to jointly optimize sensing and communication performance [22]. These studies confirm that divergence-based monitoring is a reliable signal for triggering adaptation but have not yet been tightly integrated into DT synchronization policies.

Previous work has addressed various facets of the synchronization problem, including adaptive scheduling, network optimization, incentive-driven coordination, and forecasting-based techniques. However, most approaches overlook predictive uncertainty or assume that the predictive model remains valid over time. Likewise, existing drift detection methods are not incorporated into the synchronization loop. Our work bridges these gaps by combining quantile-based forecasting with KL-divergence-driven drift detection to create an adaptive synchronization framework. This approach enables the DT to dynamically decide when to rely on local predictions and when to synchronize fully, thereby achieving high fidelity under bandwidth constraints.

III. Methodology and Framework

A. Proposed Adaptive Synchronization Framework

As illustrated in Figure 1, at each synchronization interval t, the PT transmits only a selected subset of N state variables; the DT infers the withheld variables through a multi-output predictor, providing a complete state vector for downstream analytics. Because the DT operates on predicted states rather than requiring all raw measurements, only the minimal necessary data is transmitted, making the framework inherently bandwidth- and privacy-efficient. For example, during a 2-second synchronization interval, the PT transmits only a subset of parameters (e.g., RSSI, SNR and RSRQ), while the DT predicts the remaining states (e.g., RSRP and Downlink bitrate) and proceeds with analytics without requiring raw measurements. These intervals typically range from one second to two minutes, depending on the application requirements. All forecasting, uncertainty monitoring, drift detection, and retraining decisions are executed at the DT. The PT remains lightweight: it only senses parameters and transmits data when requested, and maintains a small temporary buffer of recent measurements for retraining operations. The adaptive controller periodically evaluates forecast risk and may fall back to direct synchronization and retraining. The full decision logic is summarized in Section III-E. To remain effective under changing conditions, the predictor is retrained in an event-driven manner. Retraining is triggered only when drift is detected or when coverage remains below the threshold over a specified monitoring window. It utilizes the most recent synchronized data to recalibrate without unnecessary updates during stable periods.

For multi-parameter operation, we implement a shared buffer at the PT. Suppose any target is flagged as risky by the drift test. In that case, the system immediately reverts to direct synchronization for all targets in the next interval. This design leverages cross-parameter correlations, accelerates adaptability to evolving conditions, and shortens the time needed to establish a reliable retraining set compared to individual parameter buffers, which can delay model updates.

In practice, the multi-output forecaster achieves a high R^2 for withheld variables, indicating that the approximation error between validation points remains minimal. Overall, the framework effectively balances communication costs and accuracy by transmitting only necessary data during stable periods while quickly reverting to direct synchronization when risks increase.

B. RNN-Based Quantile Prediction Model with Monotonic Bounds

The first component of the framework is a recurrent model that forecasts selected PT parameters while quantifying predictive uncertainty. We employ a Gated Recurrent Unit (GRU) network to capture temporal dependencies efficiently, as GRUs require fewer parameters and train faster than LSTMs [23].

Unlike standard regressors that output a single value, the model predicts the mean and two quantiles (10th and 90th percentiles) for each target parameter. Training minimizes a composite objective combining the mean-squared error (MSE) for $\hat{\mu}$ and the pinball loss [15] for $\hat{q}_{0.1}$ and $\hat{q}_{0.9}$:

$$L_\tau(y,\hat{y}) = \begin{cases} \tau(y-\hat{y}), & y \geq \hat{y}, \\ (1-\tau)(\hat{y}-y), & y < \hat{y}, \end{cases} \tag{1}$$

with $\tau \in \{0.1, 0.9\}$. The total loss sums the two pinball terms and MSE across all outputs and is minimized via Back-Propagation Through Time.

To guarantee logical consistency among quantiles, we apply a monotonic constraint inspired by MCQRNN [24]. The network predicts an unconstrained lower quantile $\tilde{q}_{0.1}$ and positive offsets using softplus activations:

$$\hat{\mu} = \tilde{q}_{0.1} + \mathrm{softplus}(\Delta_\mu), \quad \hat{q}_{0.9} = \hat{\mu} + \mathrm{softplus}(\Delta_q), \tag{2}$$

ensuring $\hat{q}_{0.1} \leq \hat{\mu} \leq \hat{q}_{0.9}$. This non-crossing constraint produces well-ordered intervals that serve as calibrated uncertainty estimates.

Algorithm 1: Training the GRU Mean+Quantile Model

```
Input: Time-series $\{y_t\}_{t=1}^T$; quantiles $\tau_1$=0.1, $\tau_2$=0.9;
       learning rate $\eta$
Output: Trained model $M$
Initialize GRU parameters $\Theta$
for $e = 1$ to $E$ do
    for mini-batch $\mathbf{X}$ do
        $r \leftarrow \mathrm{GRU}_\Theta(\mathbf{X})$
        Apply monotonic bounds via softplus to get
          $(\hat{q}_{0.1}, \hat{\mu}, \hat{q}_{0.9})$
        Compute $L_{0.1}$, $L_{0.9}$ via Eq. (1) and MSE($\hat{\mu}, y$)
        $\mathcal{L} \leftarrow \frac{1}{3}(L_{0.1} + L_{0.9} + \mathrm{MSE})$
        Update $\Theta \leftarrow \Theta - \eta \nabla_\Theta \mathcal{L}$
    end
end
return $M(\cdot;\Theta)$
```

At inference, the model consumes synchronized PT features and outputs $\hat{q}_{0.1}(t)$, $\hat{\mu}(t)$, and $\hat{q}_{0.9}(t)$ for each target. The $[\hat{q}_{0.1}, \hat{q}_{0.9}]$ interval defines predictive uncertainty, guiding whether the DT should rely on model forecasts or request full synchronization. Using 10th–90th percentile bounds provides a balanced trade-off between reliability and sensitivity to outliers, yielding robust uncertainty calibration for adaptive synchronization [25].

C. KL Divergence Estimation via Kernel Density

The second component of the framework quantifies the distributional drift between the predictive output of the model and the reference (training) distribution using the Kullback–Leibler (KL) divergence.

$$D_{\mathrm{KL}}(P \parallel Q) = \int p(x) \log \frac{p(x)}{q(x)}\, dx, \tag{3}$$

where $p(x)$ denotes the reference density and $q(x)$ the density of recent predictions. A high D_{KL} indicates that the model's output distribution has diverged from the training distribution, indicating potential degradation of the reliability of the prediction or less-confident predictions.

To estimate $p(x)$ and $q(x)$, we employ Kernel Density Estimation (KDE), a non-parametric technique well suited for streaming contexts with limited samples [26]. Given n samples $x_1, \ldots, x_n$, the KDE is:

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\Big(\frac{x - x_i}{h}\Big), \tag{4}$$

where $K(\cdot)$ is a kernel (Gaussian in our case) and h is the bandwidth determined via Silverman's rule [27]:

$$h_{\text{Silverman}} = 1.06\, \sigma\, n^{-1/5}. \tag{5}$$

Since KDEs have no closed-form integral, we approximate D_{KL} numerically using a discrete Riemann sum over a common grid:

$$\widehat{D_{\mathrm{KL}}}(P \parallel Q) = \sum_j \hat{p}(x_j) \log \frac{\hat{p}(x_j)}{\hat{q}(x_j)}. \tag{6}$$

To ensure numerical stability, both $\hat{p}$ and $\hat{q}$ are smoothed with a small ε and renormalized to integrate to one.

Algorithm 2: KL Divergence Estimation via KDE

```
Input: Reference samples $A = \{a_i\}$, predicted samples
        $Z = \{z_j\}$
Output: Estimated $\widehat{D_{\mathrm{KL}}}(A \parallel Z)$
Compute KDEs $\hat{p}$ and $\hat{q}$ using Gaussian kernel (bandwidth
  via Silverman's rule)
Normalize to $\sum_j \hat{p}(x_j) = \sum_j \hat{q}(x_j) = 1$
Compute $\widehat{D_{\mathrm{KL}}} = \sum_j \hat{p}(x_j) \log \frac{\hat{p}(x_j)}{\hat{q}(x_j)}$
return $\widehat{D_{\mathrm{KL}}}$
```

This divergence is evaluated at each synchronization interval. If D_{KL} exceeds a predefined threshold, the system flags drift, triggers full synchronization, and retrains the predictive model. Otherwise, the DT remains in predictive mode. This simple yet effective approach enables real-time, drift-aware control without imposing heavy computational overhead. KL divergence is non-negative and unbounded, with $D_{\mathrm{KL}} = 0$ indicating identical distributions. Since D_{KL} is estimated from finite samples via KDE, we choose the drift threshold empirically and set it to 7.5 in all experiments based on observed distribution alignment.

D. Coverage Calculation for Prediction Intervals

The third component of the framework evaluates the empirical reliability of the model's predicted intervals through a coverage metric. Coverage is defined as the proportion of ground-truth observations that fall within the predicted quantile bounds $[\hat{q}_{0.1}(t), \hat{q}_{0.9}(t)]$. For each time step t, we define an indicator $I_t = 1$ if $\hat{q}_{0.1}(t) \leq y_t \leq \hat{q}_{0.9}(t)$, and $I_t = 0$ otherwise. The empirical coverage over a window of N samples is then:

$$\text{Coverage} = \frac{1}{N} \sum_{t=1}^{N} I_t. \tag{7}$$

Ideally, the empirical coverage should match the nominal interval (80%). However, due to sampling variability, slight deviations are acceptable [28]. We set a tolerance threshold of 0.75 (75%) to indicate adequate calibration. When coverage falls below this level, the model is deemed unreliable, triggering full synchronization until recalibration is achieved.

Coverage is recomputed periodically during direct synchronization using the most recent N ground-truth samples. Each non-overlapping block of samples is processed independently, with coverage serving as a key signal for deciding when to revert from direct to predictive mode. High coverage confirms that the model's uncertainty estimates are well-calibrated, allowing the twin to resume bandwidth-efficient operation.

Algorithm 3: Coverage Computation

```
Input: $N$, bounds $\hat{q}_{0.1}(k), \hat{q}_{0.9}(k)$, ground-truth $Y$
Output: Coverage $C$
$hits \leftarrow 0$
for $k = 1$ to $N$ do
    $hits \mathrel{+}= (\hat{q}_{0.1}(k) \leq Y_k \leq \hat{q}_{0.9}(k))$
end
$C \leftarrow hits/N$
return $C$
```

E. Adaptive Synchronization Decision Logic and Retraining Mechanism

The adaptive decision logic integrates the components introduced earlier; quantile-based forecasts, KL divergence, and empirical coverage to decide whether the DT should operate in PREDICTIVE or DIRECT mode at each synchronization interval. The goal is to minimize bandwidth use without compromising fidelity.

At each interval, if the DT is in predictive mode, the framework first computes the KL divergence between the predicted and reference distributions (Algorithm 2). If the divergence D_{KL} remains below the predefined threshold $D_{\max}$, predictions are deemed reliable, and the DT is updated without transmitting new data. Otherwise, the system switches to direct synchronization for the next interval, streaming all ground-truth values from the PT.

During direct mode, the framework evaluates the empirical coverage of recent prediction intervals using Algorithm 3. If the coverage C_t exceeds the threshold $C_{\min}$ (set to 0.75 in our experiments), the DT returns to predictive mode. If coverage remains below threshold across multiple intervals, the system triggers a retraining phase, using the newly synchronized data to fine-tune the GRU model. This retraining is event-driven, ensuring computational efficiency by updating only when significant drift or calibration loss is detected.

The logic naturally extends to multi-output forecasting: KL divergence is monitored per parameter, and a switch to direct mode occurs if any target violates its threshold. Retraining then leverages shared data buffers, enabling faster adaptation to the evolving PT dynamics. This cooperative design prevents prolonged degradation of accuracy while maintaining communication efficiency.

Algorithm 4: Adaptive Synchronization Decision Logic

```
Input: Predicted samples $Z$, ground-truth $Y$ (only if direct
       mode), model $M$, state $S_t \in \{\textsc{Predictive}, \textsc{Direct}\}$,
       thresholds $D_{\max}, C_{\min}$
Output: Next state $S_{t+1}$
if $S_t = \textsc{Predictive}$ then
    Compute $D_{\mathrm{KL}}$ via Algorithm 2
    if $D_{\mathrm{KL}} \leq D_{\max}$ then
        Update DT with predictions; $S_{t+1} \leftarrow \textsc{Predictive}$
    else
        Switch to direct mode; synchronize actual $Y$
        Compute coverage $C_t$ via Algorithm 3
        if $C_t < C_{\min}$ then
            Retrain $M$ using recent data;
        end
        $S_{t+1} \leftarrow \textsc{Direct}$
    end
end
else
    Update DT with synchronized $Y$
    Compute $C_t$; Retrain $M$ incrementally
    $S_{t+1} \leftarrow (C_t \geq C_{\min})?\textsc{Predictive} : \textsc{Direct}$
end
return $S_{t+1}$
```

This decision process ensures the DT maintains high accuracy while minimizing unnecessary data transfer. When the model exhibits stable performance (low divergence, high coverage), the system prioritizes predictive synchronization. Conversely, unexpected behavior or poor calibration triggers direct updates and selective retraining, allowing the twin to adapt swiftly to new operating conditions with minimal bandwidth overhead.

TABLE I
DIGITAL TWIN PARAMETER SYNCHRONIZATION MODES

Parameter	Synchronization Mode
Reference Signal Received Power (RSRP)	Adaptive Sync.
Downlink bitrate	Adaptive Sync.
Uplink bitrate	Direct Sync.
Longitude of user equipment	Direct Sync.
Latitude of user equipment	Direct Sync.
Speed of user equipment	Direct Sync.
Channel Quality Indicator (CQI)	Direct Sync.
Received Signal Strength Indicator (RSSI)	Direct Sync.
Signal-to-Noise Ratio (SNR)	Direct Sync.
Reference Signal Received Quality (RSRQ)	Direct Sync.

F. Experiment Design and Overhead Analysis

To evaluate the proposed framework, we implement the adaptive synchronization system on our developed DT of a smartphone operating in a cellular environment, as described in [29]. The smartphone DT models radio and network performance indicators in real time, enabling predictive analytics, energy-efficient optimization, and dynamic network adaptation.

This approach can extend to Bluetooth and Wi-Fi, as noted in [29], provided reliable inter-parameter correlations and accurate prediction models exist. However, scalability depends on correlation strength and predictive accuracy: expanding to more parameters without sufficiently accurate models can reduce DT fidelity, trigger frequent direct-synchronization fallbacks, and diminish overall gains.

The multi-output forecasting model simultaneously predicts two parameters: Reference Signal Received Power (RSRP) and downlink bitrate at the DT, while the remaining eight Key Performance Indicators (KPIs) are synchronized periodically. These two variables were selected based on a correlation study showing strong dependence on other metrics such as SNR, CQI, and RSSI. Table I summarizes the synchronization assignment.

To quantify efficiency, we define synchronization overhead as the proportion of parameters transmitted in direct mode. The reduction in synchronization overhead is:

$$\begin{matrix}\text{Overhead}\\ \text{Reduction}\end{matrix} = 1 - \frac{\begin{matrix}\text{Directly synchronized}\\ \text{parameter count}\end{matrix}}{\begin{matrix}\text{Total parameter count}\\ (P_{\text{total}})\end{matrix}}. \tag{8}$$

where the total parameter count is:

$$P_{\text{total}} = P_{\text{sample}} \times S_{\text{sync}} \times N_{\text{sync}}, \tag{9}$$

with P_{sample} the number of parameters per sample, S_{sync} the samples per synchronization interval, and N_{sync} the total number of intervals. All parameters are assumed to have uniform payload size; control overhead and retransmissions are omitted for clarity.

TABLE II
PREDICTION R^2 AND OVERHEAD REDUCTION FOR PREDICTIVE-ONLY VS. PROPOSED ADAPTIVE APPROACH (MULTI-OUTPUT MODEL)

Dataset	R^2 (RSRP)		R^2 (DL bitrate)		Reduction (%)	
	Pred.	Proposed	Pred.	Proposed	Pred.	Proposed
Car (LTE)	0.941	0.995	0.793	0.973	20	2.15
Static (LTE)	0.619	0.900	0.356	0.856	20	14.89
Train (LTE)	0.829	0.951	0.774	0.915	20	10.00
Bus (LTE)	0.863	0.960	0.656	0.984	20	4.35
Pedestrian (LTE)	0.932	0.983	0.906	0.985	20	4.81
Car (5G)	0.709	0.725	0.436	0.477	20	19.13

TABLE III
PREDICTION MAE FOR PREDICTIVE-ONLY VS. PROPOSED ADAPTIVE APPROACH (MULTI-OUTPUT MODEL)

Dataset	MAE (RSRP) [dBm]		MAE (DL rate) [kbit/s]	
	Pred.	Proposed	Pred.	Proposed
Car (LTE)	1.71	0.16	5095	535
Static (LTE)	2.15	0.73	1257	379
Train (LTE)	4.17	1.46	3405	1376
Bus (LTE)	2.88	0.74	4896	587
Pedestrian (LTE)	2.36	0.59	1741	368
Car (5G)	4.03	3.81	21877	20313

When two of ten parameters are managed adaptively, the theoretical maximum payload reduction is 20%. However, the proposed event-driven framework achieves nearly a similar reduction while maintaining a higher fidelity through dynamic switching and retraining, unlike a predictive-only scheme that risks long-term drift. More generally, if M of N parameters are reliably predicted, the payload reduction approaches M/N, provided update frequencies are similar.

IV. EXPERIMENTS AND RESULTS

A. Prediction Accuracy

To validate the framework in a setting that resembles the smartphone DT use case described in the Methodology section, we conducted experiments on two publicly available cellular network datasets. These datasets captured the same KPIs, as listed in Table I, with one dataset for 4G LTE and the other for 5G NR. Both were collected under static and mobile conditions [30], [31].

The model was benchmarked against a predictive-only baseline, which performs continuous forecasting without adaptive retraining or divergence-based switching. The performance was assessed using the coefficient of determination (R^2), Mean Absolute Error (MAE), and synchronization overhead reduction compared to direct synchronization (default: $D_{\mathrm{KL}} = 7.5$, $C_{\min} = 75\%$, $T = 60$ s).

Tables II and III show the adaptive strategy outperforms predictive-only across LTE and 5G scenarios. In all LTE mobility patterns, the adaptive model achieves higher R^2 values and lower MAE for both RSRP and downlink bitrate, while providing significant synchronization overhead reduction. The improvement is most pronounced in the train and static LTE cases, where the MAE decreases by more than 59% and the R^2 exceeds 0.85 for both targets. The results confirm that the proposed multi-output approach effectively captures cross-parameter correlations and maintains high-fidelity digital twinning.

TABLE IV
IMPACT OF THE KL DIVERGENCE THRESHOLD (D_{KL})
($C_{\min} = 75\%, T = 60$ S)

D_{KL}	R^2		MAE		Overhead
	RSRP	DL rate	RSRP (dBm)	DL rate (kbit/s)	Reduction (%)
1.0	0.998	0.990	0.07	105	0.62
2.5	0.986	0.991	0.32	175	1.25
5.0	0.951	0.915	1.46	1376	10.00
7.5	0.951	0.915	1.46	1376	10.00
10.0	0.829	0.774	4.17	3405	20.00

B. Effects of Adaptive Synchronization Criteria

To analyze the sensitivity of our framework to its adaptive synchronization criteria, we conducted experiments on the LTE train dataset while varying the KL divergence threshold (D_{KL}), the coverage threshold ($C_{\min}$), and the base synchronization interval. These parameters determine when the DT transitions between predictive and direct synchronization modes. The evaluation quantifies the trade-off between prediction accuracy and synchronization overhead reduction, with R^2 and MAE serving as accuracy measures.

KL Divergence Threshold (D_{KL}): As shown in Table IV, a low threshold (for instance, $D_{\mathrm{KL}} = 1.0$) results in frequent synchronization, producing near-perfect accuracy ($R^2 = 0.998$ for RSRP and 0.990 for downlink bitrate) but with minimal reduction in communication overhead (0.62%). Increasing the threshold allows the DT to remain in predictive mode for longer periods, improving payload efficiency but introducing greater model drift. At $D_{\mathrm{KL}} = 5.0$, the framework achieves a balanced performance, with $R^2 = 0.951$ for RSRP and 0.915 for bitrate, and approximately a 10% reduction in overhead. Very high thresholds ($D_{\mathrm{KL}} = 10.0$) further lower overhead to about 20%, although accuracy degrades noticeably ($R^2 \approx 0.83$). Therefore, moderate thresholds in the range of 5.0 to 7.5 provide the best trade-off between fidelity and efficiency for the dataset considered.

A plateau occurs for $5.0 \leq D_{\mathrm{KL}} \leq 7.5$: Table IV shows identical R^2, MAE, and overhead reductions. With $C_{\min} = 75\%$ and 60 s intervals, increasing D_{KL} in this range does not change drift-trigger times because decisions are made only at discrete instants, yielding the same synchronize decision.

Coverage Threshold ($C_{\min}$): As summarized in Table V, higher coverage thresholds lead to more frequent synchronizations and higher accuracy but reduce overhead savings. A strict threshold of $C_{\min} = 90\%$ maintains excellent fidelity ($R^2 \approx 0.96$) but limits overhead reduction to 3.75%. In contrast, setting $C_{\min} = 75\%$ allows a modest decline in accuracy

TABLE V
IMPACT OF THE COVERAGE THRESHOLD
($C_{\min}$) ($D_{\text{KL}} = 7.5, T = 60$ s)

$C_{\min}$ (%)	R^2		MAE		Overhead
	RSRP	DL rate	RSRP (dBm)	DL rate (kbit/s)	Reduction (%)
70	0.875	0.872	2.98	2222	16.25
75	0.951	0.915	1.46	1376	10.00
80	0.961	0.960	0.95	657	3.75
85	0.961	0.960	0.95	657	3.75
90	0.961	0.960	0.95	657	3.75

TABLE VI
IMPACT OF THE BASE SYNCHRONIZATION INTERVAL (T)
($D_{\text{KL}} = 7.5, C_{\min} = 75\%$)

T (s)	R^2		MAE		Overhead
	RSRP	DL rate	RSRP (dBm)	DL rate (kbit/s)	Reduction (%)
5	0.978	0.956	0.74	766	5.50
30	0.870	0.879	2.97	2021	16.00
60	0.951	0.915	1.46	1376	10.00
90	0.959	0.946	1.07	870	5.71
120	0.959	0.951	1.06	786	5.00

($R^2 = 0.951$ for RSRP, 0.915 for bitrate) while increasing overhead reduction to 10%. Lowering the threshold further to 70% increases overhead reductions to 16.25% but introduces greater prediction errors. Hence, a coverage threshold around 75% offers the most balanced configuration for the dataset we consider.

A plateau is observed for $C_{\min} \geq 80\%$: Table V reports identical R^2, MAE, and overhead reductions. With $D_{\text{KL}} = 7.5$ and 60 s intervals, increasing $C_{\min}$ beyond 80% does not change trigger times because coverage is checked only at discrete decision instants, yielding the same synchronization decision.

Base Synchronization Interval: This shows how varying the synchronization interval (T seconds) affects the balance between DT accuracy and communication efficiency. The results in Table VI indicate a non-linear dependency. Very short intervals (e.g., $T = 5$ s) ensure high accuracy ($R^2 = 0.98$ for RSRP, 0.96 for bitrate) but yield only a 5.5% reduction in overhead due to frequent drift triggers caused by smaller sample sizes. As T increases to 30 s, synchronization frequency decreases and overhead reduction improves to 16%, albeit with lower accuracy ($R^2 \approx 0.87$). The optimal balance occurs near $T = 60$ s, where accuracy remains above $R^2 = 0.91$ with a 10% overhead reduction. Longer intervals (90–120 s) again slightly reduce efficiency due to accumulated model drift requiring more corrective synchronizations.

Joint Effect and Design Trade-Offs: Across all experiments, we observe a consistent trade-off between synchronization efficiency and model accuracy. Stricter policies (low D_{KL}, high $C_{\min}$, or short synchronization intervals) maintain near-perfect fidelity but incur higher communication costs, whereas more relaxed configurations substantially reduce synchronization overhead at the expense of moderate drift in the DT state. The optimal operating point depends on the application's tolerance for latency and accuracy degradation. For the analyzed dataset, moderate configurations with $D_{\text{KL}} \in [5, 7.5]$, $C_{\min} = 75\%$, and $T = 60$ s provide a balanced regime, achieving $R^2 > 0.91$ for both target variables and approximately 10% reduction in synchronization overhead. These results confirm that the proposed adaptive synchronization framework can be tuned flexibly to meet the performance requirements of diverse DT deployments.

TABLE VII
LATENCY AND RESOURCE UTILIZATION OF THE PROPOSED FRAMEWORK

Mode	Total latency (ms)	CPU utilization (%)	Memory utilization (%)
Predictive (no retraining)	26	2.1	5.0
Predictive (with retraining)	769	11.1	5.0
Direct (with retraining)	433	7.4	5.0

C. Runtime Efficiency of the Proposed Framework

We evaluated the computational efficiency of the proposed framework using the static LTE dataset on a mid-range laptop equipped with an Apple M2 processor, 16 GB RAM, and macOS 15.5 (24F74). The profiling results, summarized in Table VII, quantify total latency and average resource utilization for the main operating modes of the system.

Across all modes, the total processing latency remains well below one second, indicating that the framework can support synchronization periods significantly shorter than those typically required in operational deployments. Predictive synchronization without retraining exhibits the lowest latency of 26 ms, as it only performs divergence tests on the model's predictive distributions. When retraining is enabled, latency increases to approximately 0.77 s due to the additional steps of divergence estimation, coverage verification, and model updating. Direct synchronization, which includes both coverage evaluation and retraining, incurs a moderate latency of 0.43 s.

Resource utilization follows a similar trend: predictive synchronization with retraining averages about 11% CPU usage, direct mode about 7%, and predictive mode without retraining only 2%. Memory consumption remains stable at roughly 5% across all modes, demonstrating consistent and lightweight operation.

D. Empirical Network-Traffic Evaluation

We validated the network traffic savings using our developed DT solution [29], where a commercial Android smartphone (PT) transmitted KPI streams over an LTE network to a remote DT endpoint. The phone replays KPI payloads identical to those in the labeled dataset, and no real-time ground-truth labels are collected during deployment; hence, no on-device accuracy is measured. Accuracy (MAE/R^2) is evaluated offline on the labeled dataset (Tables II and III), while trace replay is used only to analyze DT synchronization overhead.

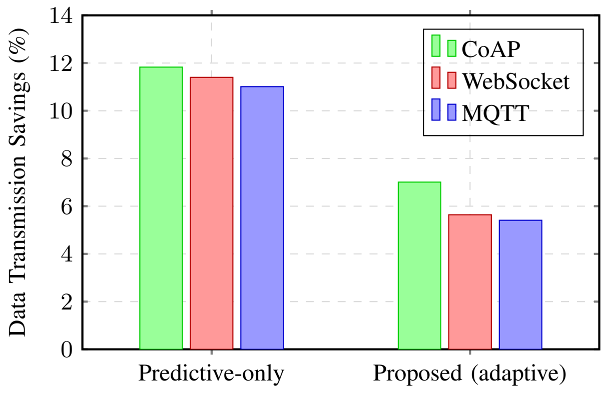

Fig. 2. Data transmission savings relative to direct synchronization. Across all protocols, the adaptive method consistently reduces network traffic while preserving fidelity. The predictive-only baseline achieves higher savings but leads to substantially degraded accuracy (cf. Table III).

In these experiments, the DT server executed no control logic in order to isolate the communication overhead. We evaluated two synchronization strategies: predictive only (suppressing two target KPIs) and the proposed adaptive approach, and compared both against the overhead of direct synchronization. The experiments were conducted across three transport protocols commonly used in DT systems (MQTT, WebSocket, and CoAP) [32]–[34]. Each trial lasted 47 minutes, using the same static dataset payload as in Tables II and III.

As shown in Fig. 2, across all protocols the adaptive method reduces traffic relative to direct synchronization (MQTT: 5.41%, WebSocket: 5.64%, CoAP: 7.01%). The predictive-only baseline saves slightly more bytes (11–12%) but substantially degrades fidelity (cf. Table III); across LTE scenarios, the adaptive approach reduces MAE by about 74% (RSRP) and 77% (downlink bitrate) relative to predictive-only.

Compared to the analytical payload-only reduction in Table II, empirical savings are smaller (predictive-only: 55–59% of the theoretical figure; proposed-adaptive: 36–47%), primarily due to protocol headers and JSON wrappers omitted from the analytical model (Eq. (8)) and because the byte length of sensed parameters varies with value. In practice, the adaptive strategy strikes the intended balance: it identifies risky predictions via KL divergence and coverage checks, triggers direct synchronization and retraining when needed, and thereby maintains close alignment with PT states while still delivering tangible network savings across diverse DT transport protocols.

E. Operational Cost and Scalability Analysis

Wireless data transmission dominates power consumption in most constrained IoT nodes, particularly in low-power wide-area network (LPWAN) technologies such as LoRa and NB-IoT [35]. As each transmitted bit incurs an energy cost, even modest traffic reductions translate into nearly proportional power savings, effectively extending device lifetime. Reducing data volume by 5-7% yields a comparable reduction in radio activity, which not only conserves energy but also mitigates the impact of data caps and bandwidth quotas in pay-as-you-go IoT subscriptions. Because these plans charge directly by data usage, small traffic reductions can produce measurable financial savings.

TABLE VIII
Cost Savings per Device under Various Cellular IoT Operators

Operator	Rate (per MB)	Saving (per month)	Saving (per year)
Telnyx	$0.01	$0.32	$3.84
Hologram	$0.03	$0.96	$11.52
ThingsMobile	$0.10	$3.20	$38.40

We quantified the cost benefits of the adaptive synchronization framework using three global IoT service providers: Telnyx, Hologram, and ThingsMobile, each employing pay-per-megabyte billing. Their respective rates are $0.01/MB, $0.03/MB, and $0.10/MB [36]–[38]. Based on our CoAP evaluation, the adaptive framework reduces traffic by approximately 31.996MB per month for each device. This reduction translates to monthly savings of $0.32 with Telnyx, $0.96 with Hologram, and $3.20 with ThingsMobile, corresponding to annual savings of $3.84, $11.52, and $38.40, respectively (Table VIII). Even modest reductions in data traffic, on the order of a few tens of megabytes, can therefore result in meaningful cost savings, particularly in large-scale IoT deployments.

In large-scale IoT networks, such per-device savings accumulate rapidly. For example, a deployment of 250 devices, each saving approximately 32MB per month, results in a cumulative reduction of around 8GB per month. This reduction corresponds to annual network-wide cost savings of about $960 with Telnyx, $2,880 with Hologram, and $9,600 with ThingsMobile. In addition to lowering operational costs, reduced data transmissions help minimize channel contention, airtime occupancy, and interference, which are critical considerations for bandwidth-constrained IoT systems. Fewer transmissions also shorten radio-on time, thereby reducing per-node power consumption and extending battery life across the device fleet.

V. Conclusion

This paper presented an adaptive synchronization framework for DTs designed to maintain high-fidelity state representation under constrained network conditions. The key idea is to predict DT states that exhibit temporal stability and synchronize only when forecasts become unreliable. Overall, the adaptive synchronization strategy achieves measurable cost and energy efficiency gains without compromising DT fidelity. By transmitting only essential updates and dynamically retraining the predictive model upon drift detection, it minimizes communication overhead while maintaining alignment between the DT and PT. These advantages make the framework particularly suitable for cost-sensitive, energy-constrained IoT deployments, enabling scalable and sustainable DT ecosystems.

References

[1] S. Mihai, M. Yaqoob, D. V. Hung, W. Davis, P. Towakel, M. Raza, M. Karamanoglu, B. Barn, D. Shetve, R. V. Prasad, H. Venkataraman, R. Trestian, and H. X. Nguyen, "Digital twins: A survey on enabling technologies, challenges, trends and future prospects," *IEEE Communications Surveys Tutorials*, vol. 24, no. 4, pp. 2255–2291, 2022.

[2] M. Singh, E. Fuenmayor, E. P. Hinchy, Y. Qiao, N. Murray, and D. Devine, "Digital twin: Origin to future," *Applied System Innovation*, vol. 4, no. 36, 2021.

[3] M. Singh, R. Srivastava, E. Fuenmayor, V. Kuts, Y. Qiao, N. Murray, and D. Devine, "Applications of digital twin across industries: A review," *Applied Sciences*, vol. 12, no. 5727, 2022.

[4] M. A. Uusitalo, P. Rugeland, M. R. Boldi, E. Calvanese Strinati, P. Demestichas, M. Ericson, G. P. Fettweis, M. C. Filippou, A. Gati, M.-H. Hamon, M. Hoffmann, M. Latva-Aho, A. Pärssinen, B. Richerzhagen, H. Schotten, T. Svensson, G. Wikström, H. Wymeersch, V. Ziegler, and Y. Zou, "6g vision, value, use cases and technologies from european 6g flagship project hexa-x," *IEEE Access*, vol. 9, pp. 160 004–160 020, 2021.

[5] F. Fouquet, T. Hartmann, C. Cecchinel, and B. Combemale, "Greycat: A framework to develop digital twins at large scale," in *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, ser. MODELS Companion '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 492–495. [Online]. Available: https://doi.org/10.1145/3652620.3688265

[6] H. Yang, Y. Li, K. Yao, T. Sun, and C. Zhou, "A systematic network traffic emulation framework for digital twin network," in *2021 IEEE 1st International Conference on Digital Twins and Parallel Intelligence (DTPI)*, 2021, pp. 94–97.

[7] B. Tan and A. Matta, "The digital twin synchronization problem: Framework, formulations, and analysis," *IISE Transactions*, vol. 56, no. 6, pp. 652–665, 2024. [Online]. Available: https://doi.org/10.1080/24725854.2023.2253869

[8] R. Gupta, B. Tian, Y. Wang, and K. Nahrstedt, "TWIN-ADAPT: Continuous learning for digital twin-enabled online anomaly classification in IoT-driven smart labs," *Future Internet*, vol. 16, no. 239, 2024.

[9] J. L. Lobo, I. Laña, E. Osaba, and J. Del Ser, "On the connection between concept drift and uncertainty in industrial artificial intelligence," in *2023 IEEE Conference on Artificial Intelligence (CAI)*, 2023, pp. 171–172.

[10] D. Kalasapura, J. Li, S. Liu, Y. Chen, R. Wang, T. Abdelzaher, M. Caesar, J. Bhattacharyya, J. Kim, G. Wang, G. Kimberly, J. Eckhardt, and D. Osipychev, "Twinsync: A digital twin synchronization protocol for bandwidth-limited iot applications," in *2023 32nd International Conference on Computer Communications and Networks (ICCCN)*, 2023, pp. 1–1.

[11] J. Zheng, T. H. Luan, Y. Zhang, R. Li, Y. Hui, L. Gao, and M. Dong, "Data synchronization in vehicular digital twin network: A game theoretic approach," *IEEE Transactions on Wireless Communications*, vol. 22, no. 11, pp. 7635–7647, 2023.

[12] L. V. Cakir, S. Al-Shareeda, S. F. Oktug, M. Özdem, M. Broadbent, and B. Canberk, "How to synchronize digital twins? a communication performance analysis," in *2023 IEEE 28th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2023, pp. 123–127.

[13] Z. Yang, M. Chen, Y. Liu, and Z. Zhang, "Optimizing synchronization delay for digital twin over wireless networks," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 9106–9110.

[14] L. V. Cakir, C. J. Thomson, M. Özdem, B. Canberk, V.-L. Nguyen, and T. Q. Duong, "Intelligent digital twin communication framework for addressing accuracy and timeliness tradeoff in resource-constrained networks," *IEEE Transactions on Cognitive Communications and Networking*, pp. 1–1, 2024.

[15] I. Takeuchi, Q. V. Le, T. D. Sears, and A. J. Smola, "Nonparametric quantile estimation," *Journal of Machine Learning Research*, vol. 7, no. 45, pp. 1231–1264, 2006.

[16] A. I. Tambuwal and D. Neagu, "Deep Quantile Regression for Unsupervised Anomaly Detection in Time-Series," *SN Computer Science*, vol. 2, no. 6, p. 475, Sep. 2021.

[17] L. B. Lima, F. Cribari-Neto, and D. P. Lima-Junior, "Dynamic quantile regression for trend analysis of streamflow time series," *River Research and Applications*, vol. 38, no. 6, pp. 1051–1060, 2022.

[18] K. Wang, D. Zhang, and X. Sun, "Robust composite quantile regression with large-scale streaming data sets," *Scandinavian Journal of Statistics*, vol. 52, no. 2, pp. 736–755, 2025.

[19] M. Xie, J. Hu, S. Guo, and A. Y. Zomaya, "Distributed segment-based anomaly detection with kullback–leibler divergence in wireless sensor networks," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 1, pp. 101–110, 2017.

[20] J. F. Kurian and M. Allali, "Detecting drifts in data streams using Kullback-Leibler (KL) divergence measure for data engineering applications," *Journal of Data, Information and Management*, vol. 6, no. 3, pp. 207–216, Sep. 2024.

[21] P. Kaswan, M. F. Marzban, W. Nam, S. Akkarakaran, and T. Luo, "Statistical ai/ml model monitoring for 5g/6g: Interference prediction case study," in *2024 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2024, pp. 638–643.

[22] M. Al-Jarrah, E. Alsusa, and C. Masouros, "A unified performance framework for integrated sensing-communications based on kl-divergence," *IEEE Transactions on Wireless Communications*, vol. 22, no. 12, pp. 9390–9411, 2023.

[23] P. T. Yamak, L. Yujian, and P. K. Gadosey, "A comparison between arima, lstm, and gru for time series forecasting," in *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, ser. ACAI '19. ACM, 2020, p. 49–55. [Online]. Available: https://doi.org/10.1145/3377713.3377722

[24] A. J. Cannon, "Non-crossing nonlinear regression quantiles by monotone composite quantile regression neural network, with application to rainfall extremes," *Stochastic Environmental Research and Risk Assessment*, vol. 32, no. 11, pp. 3207–3225, Nov. 2018.

[25] A. Abdalla, W. O. El-Khafifi, A. M. Mami, and N. Elmesmari, "Eliminate the heterogeneous variances effect using quantile regression," *International Journal for Multidisciplinary Research (IJFMR)*, vol. 5, no. 5, pp. 1–13, 2023, published October 27, 2023, Volume 5, Issue 5 (September–October 2023). [Online]. Available: https://doi.org/gszvfj

[26] R. R. Wilcox, "Chapter 3 - estimating measures of location and scale," in *Introduction to Robust Estimation and Hypothesis Testing (Fifth Edition)*, fifth edition ed., R. R. Wilcox, Ed. Academic Press, 2022, pp. 45–106.

[27] B. W. Silverman, *Density estimation for statistics and data analysis*. Routledge, Feb. 2018.

[28] T. Gneiting and A. E. and Raftery, "Strictly Proper Scoring Rules, Prediction, and Estimation," *Journal of the American Statistical Association*, vol. 102, no. 477, pp. 359–378, Mar. 2007.

[29] R. Morabito, B. Pandey, P. Daubaris, Y. R. Wanigarathna, and S. Tarkoma, "A generative ai-enhanced digital twin framework for heterogeneous networks and smart environments," in *IEEE INFOCOM 2025 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2025, pp. 1–7.

[30] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond throughput: a 4g lte dataset with channel and context metrics," in *Proceedings of the 9th ACM Multimedia Systems Conference*, ser. MMSys '18. ACM, 2018, p. 460–465. [Online]. Available: https://doi.org/10.1145/3204949.3208123

[31] D. Raca, D. Leahy, C. J. Sreenan, and J. J. Quinlan, "Beyond throughput, the next generation: a 5g dataset with channel and context metrics," in *Proceedings of the 11th ACM Multimedia Systems Conference*, ser. MMSys '20. ACM, 2020, p. 303–308. [Online]. Available: https://doi.org/10.1145/3339825.3394938

[32] AWS IoT TwinMaker Documentation. https://docs.aws.amazon.com/iot-twinmaker/. Accessed: 2025-08-20.

[33] Azure Digital Twins documentation. https://learn.microsoft.com/en-us/azure/digital-twins/. Accessed: 2025-08-20.

[34] ThingsBoard architecture. https://thingsboard.io/docs/reference/. Accessed: 2025-08-20.

[35] M. A. Razzaque, C. Bleakley, and S. Dobson, "Compression in wireless sensor networks: A survey and comparative evaluation," *ACM Trans. Sen. Netw.*, vol. 10, no. 1, Dec. 2013. [Online]. Available: https://doi.org/10.1145/2528948

[36] Global IoT eSIM Connectivity | Telnyx IoT & M2M SIMs. https://telnyx.com/products/iot-sim-card. Accessed: 2025-07-02.

[37] IoT data plans & pricing. https://www.hologram.io/pricing/. Accessed: 2025-07-02.

[38] IoT data plan and M2M pricing for everyone | Things Mobile. https://www.thingsmobile.com/business/plans/overview. Accessed: 2025-07-02.