# Dilated Causal CNNs for Energy Forecasting and Optimization in LoRaWAN Networks

Sana Slama †, Aida Lahouij *, Lazhar Hamel §, Mohamed Graiet *, Walid Gaaloul ‡
* Department of Computer Science, ISIMM, University Of Monastir, Monastir, Tunisia
† Faculty of Sciences of Monastir (FSM), University of Monastir, Monastir, Tunisia
§ Efrei Research Lab, Paris Panthéon Assas University, Paris, France
‡ SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, 91120 Palaiseau, France
{sanaslama95, aida.lahouij, lazhar.hamel, mohamed.graiet1}@gmail.com
walid.gaaloul@telecom-sudparis.eu

***Abstract*—Battery lifetime remains a central constraint in scaling LoRaWAN deployments across diverse IoT applications. We propose a lightweight dilated causal Convolutional Neural Network (CNN) designed to forecast per-node energy consumption with high temporal fidelity. Unlike recurrent models, our approach captures both transient spikes and long-range patterns without sequential overhead, enabling efficient edge deployment. Trained on a 12-month NS-3 simulation dataset encompassing smart lighting, environmental monitoring, waste management, and agriculture, the model achieves 96.5% forecasting accuracy with mean absolute error below 0.3, improving over SARIMA and LSTM baselines by 55% and 32% respectively.**

**We integrate this predictor into an end-to-end energy optimization pipeline where on-device inference executes every 15 minutes with under 5 ms overhead. Forecasts drive adaptive duty-cycling, transmission slotting, and data rate control, extending device lifetime by 20%, halving collision rates, and improving fairness by 15%. Real-world validation on ten STM32F407VG microcontrollers and a commercial RAK7258 gateway confirms practical feasibility: inference completes within 0.8 ms with 4.4 mW peak power draw and 95.6% packet delivery ratio. These results demonstrate the CNN's suitability for real-time, edge-centric forecasting and its potential for enabling sustainable, intelligent LoRaWAN networks.**

***Index Terms*—LoRaWAN, Deep Learning, Energy consumption, Prediction, CNN**

## I. Introduction

The rapid proliferation of Internet of Things (IoT) devices has revolutionized applications from smart cities to industrial automation [1], with LoRaWAN emerging as a prominent low-power, wide-area network solution offering extended coverage and minimal energy consumption. Particularly suited for environmental monitoring, agricultural sensors, and smart metering systems [2], LoRaWAN faces critical energy management challenges for battery-powered end devices that must operate for extended periods without maintenance.

Recent forecasting methods for LoRaWAN energy consumption rely on classical time series models, machine learning techniques, or heuristic approaches [3]. While achieving acceptable accuracy, these methods struggle to capture causal mechanisms and temporal patterns influencing energy usage, often limited by insufficient spatiotemporal data for real-world optimization. Moreover, traditional approaches like LSTM models, though effective for sequence modeling, suffer from sequential computation bottlenecks and limited parallelization when processing energy consumption patterns.

We present an alternative solution based on Dilated Causal CNN that combines causal convolution with dilation to capture long-term temporal dependencies while respecting input time ordering. Unlike recurrent architectures, CNNs enable parallel processing of energy consumption sequences, providing computational efficiency crucial for edge deployment. Our approach targets energy consumption prediction to support fine-grained optimization strategies and proactive resource management.

The key contributions include: (1) introducing Dilated Causal CNNs for LoRaWAN energy forecasting that efficiently capture long-term dependencies without recurrent computational overhead; (2) validating effectiveness through extensive experiments where our model consistently outperforms conventional baselines; and (3) proposing a forecast-driven optimization strategy that leverages predictions for efficient scheduling, extending device lifetime and improving network performance.

Section II reviews related work, Section III presents our methodology, Section IV describes the CNN architecture and NS-3 setup, Sections V–VI introduce baselines and comparative analysis, Section VII demonstrates LoRaWAN optimization results, Section VIII provides real-world evaluation, and Section IX concludes.

## II. Related Work

Energy consumption forecasting in LoRaWAN has attracted increasing attention due to the proliferation of energy-constrained IoT devices.

Recent studies have evaluated various machine learning approaches for energy prediction in IoT systems. Al-Rashid et al. [4] demonstrate that LSTM models achieve superior results ($R^2 = 0.97$, MAE = 0.007) compared to Random Forest and SVR for smart building energy

979-8-3503-9281-4/25/$31.00 ©2025 IEEE

prediction, though they do not address real-time integration or cross-domain generalization. Similarly, Bitencourt et al. [5] propose fuzzy logic-based multivariate forecasting for LoRaWAN devices, supporting multi-step prediction with improved accuracy over traditional methods, albeit with limited reproducibility due to algorithmic complexity.

Advanced deep learning approaches have shown promise for energy management in IoT networks. Zhuang et al. [6] integrate LSTM variants with reinforcement learning for HVAC control, achieving 17.4

Deep convolutional networks, particularly Dilated Causal CNNs, have demonstrated strong performance in temporal modeling applications. Quan et al. [7] apply DC-CNNs to detect rotating stall in compressors using high-frequency pressure data, outperforming classical methods in robustness and lead time. Lan et al. [8] combine Piecewise Linear Regression with DC-CNNs for electricity demand forecasting, where PLR captures seasonality while DC-CNNs model short-term fluctuations.

Recent work has explored deep reinforcement learning for LoRaWAN optimization. Jouhari et al. [9] propose UAV-based mobile gateways with PPO algorithms for joint spreading factor and power allocation, while Hamdi et al. [10] introduce LoRa-RL for hybrid-powered networks, splitting resource allocation into SF/channel assignment and energy management via PPO and DDPG respectively [10]. Both approaches rely entirely on simulation without protocol-level integration.

Although significant advances have been made in predictive modeling for IoT networks, few approaches effectively combine deep forecasting with energy-aware behavior in LoRaWAN protocols. We address this gap by employing Dilated Causal CNNs to enable proactive, forecast-driven adaptation in LoRaWAN Class A devices, supporting real-time, lightweight energy management directly on the device without requiring cloud-based inference or centralized control.

## III. Methodology

Accurate energy forecasting in *LoRaWAN networks* is essential to optimize battery lifetime and network efficiency. Traditional methods like ARIMA and Exponential Smoothing face challenges with the nonlinear dependencies and temporal variability characteristic of IoT energy patterns. While recurrent models such as LSTM and GRU improve sequence modeling, they suffer from vanishing gradients, sequential computation bottlenecks, and limited parallelism [11].

To overcome these limitations, we propose a CNN architecture incorporating causal and dilated convolutions. This design captures both short-term fluctuations and long-range temporal dependencies without recurrent overhead. CNNs inherently support parallel computation, enabling fast inference and low memory usage, which suits real-time energy forecasting on resource-constrained devices.

Formally, the forecasting function is defined as:

$$\hat{y}_{t+1} = f(y_{t-k:t}, x_{t-k:t}, s_i) \tag{1}$$

where $y_{t-k:t}$ denotes historical energy data, $x_{t-k:t}$ exogenous variables such as environmental conditions, and $s_i$ device-specific static metadata. Each convolutional layer applies:

$$z_t = \sigma(W * y_{t-k:t} + b) \tag{2}$$

with kernel weights $W$, bias $b$, convolution operator $*$, and ReLU activation $\sigma(\cdot)$.

The model is trained using **TensorFlow** on an NS-3 simulated dataset spanning 12 months, employing the **Adam optimizer** with a learning rate of 0.001 and **Mean Absolute Error (MAE)** loss:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3}$$

Hyperparameters such as kernel size, dilation rates, and depth were optimized, resulting in a 10-layer CNN with increasing dilations $\{1, 2, 4, 8, 16\}$, which balances accuracy and computational efficiency for on-device forecasting.

## IV. Data Simulation & CNN-Based Forecasting

### A. NS-3 Simulation Setup and Dataset Generation

We generate a realistic dataset using the NS-3 LoRaWAN module [12, 13] with 2,000 devices over 12 months covering four scenarios:

- **Smart Street Lighting:** Uplinks at sunrise and sunset.
- **Environmental Monitoring:** Hourly uplinks and daily confirmed packets.
- **Waste Management:** Transmission every three days upon event.
- **Agricultural Monitoring:** Daily soil moisture reports at random times.

Devices are randomly placed within an 8 km radius of the gateway with varying spreading factors and transmission rates. Energy logs record device power states (Tx, Rx, Sleep, Standby), spreading factor, packet success rate, and retransmission statistics at one-minute intervals.

### B. Data Splitting Strategy

To prevent temporal leakage, data is split chronologically with 70% for training (months 1–8/10), 15% for validation (next 1–2 months), and 15% for testing (final months):

$$\begin{aligned} \text{Train} &= \mathcal{T}_{1:\lfloor 0.7N \rfloor}, \\ \text{Validation} &= \mathcal{T}_{\lfloor 0.7N \rfloor+1:\lfloor 0.85N \rfloor}, \\ \text{Test} &= \mathcal{T}_{\lfloor 0.85N \rfloor+1:N}. \end{aligned} \tag{4}$$

### C. Time Series Cross-Validation

We employ expanding window cross-validation with a monthly step to capture seasonal patterns and natural network cycles. The folds are:

- Fold 1: Train (M1–8), Validate (M9), Test (M10)
- Fold 2: Train (M1–9), Validate (M10), Test (M11)
- Fold 3: Train (M1–10), Validate (M11), Test (M12)

### D. Calibration, Temporal Dependency and Robustness

Prediction intervals are computed via quantile regression or bootstrapping to express confidence bounds. Calibration metrics such as coverage probability are used to assess interval reliability. Temporal dependencies are handled through causal convolutions that respect time order:

$$\text{Output}_t = \phi\left(\sum_{k=0}^{K-1} W_k \cdot \text{Input}_{t-k}\right), \tag{5}$$

and dilated convolutions that exponentially expand the receptive field:

$$\text{Receptive Field} = 1 + 2d(K-1), \tag{6}$$

where $d$ is the dilation rate.

To improve generalization, Gaussian noise $\epsilon_t \sim \mathcal{N}(0, 0.05^2)$ is added to inputs while preserving autocorrelation. Feature scaling uses min-max normalization:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}. \tag{7}$$

### E. Evaluation Protocol

Performance is evaluated using point-wise metrics (MAE and RMSE), low battery event detection accuracy:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{8}$$

and uncertainty measures including prediction interval coverage and calibration error. Computational efficiency is assessed via inference latency (0.1 ms) and floating point operations. Training time on an RTX 3080 GPU is approximately six hours.

Input features include 24-hour energy history, spreading factor, success rate, retry counts, and state ratios for Tx, Rx, Sleep, and Standby modes.

## V. Baseline Models: SARIMA & LSTM

*1) Seasonal ARIMA (SARIMA):* We model energy consumption using the seasonal ARIMA model SARIMA$(p,d,q) \times (P,D,Q)_s$, with daily seasonality $s = 1440$ at one-minute resolution. A grid search over parameters

$$p, q \in \{0,1,2\},\ d \in \{0,1\},\ P, Q \in \{0,1\},\ D \in \{0,1\}$$

selects $(p,d,q) = (1,1,1), (P,D,Q) = (1,0,1)_{1440}$ based on minimum AIC.

*2) Long Short-Term Memory (LSTM):* The LSTM baseline employs two stacked layers with 64 units each, dropout rate 0.2, and a dense output layer. The input sequence length is 1440 (24 hours). Training uses the Adam optimizer (learning rate 0.001), mean absolute error loss, batch size 128, for up to 50 epochs with early stopping (patience 5). Architecture and hyperparameters were tuned on the validation set.

Both SARIMA and LSTM models were trained and evaluated using the same data splits and preprocessing as the CNN model to ensure fair comparison.

### A. CNN-Based Forecasting Model

Our proposed dilated causal CNN outperforms recurrent and autoregressive baselines by leveraging several advantages:

- **Scalability:** CNN's parallel architecture enables faster training on modern hardware compared to the sequential nature of LSTM.
- **Long-Range Dependency Modeling:** Dilated convolutions expand the receptive field exponentially without increasing model complexity, capturing temporal dependencies over extended horizons.
- **Efficient Training and Inference:** Absence of sequential recurrence reduces computational overhead, yielding faster convergence and lower latency.

The network architecture comprises 10 one-dimensional convolutional layers with kernel size 3 and dilation rates $\{1,2,4,8,16\}$ spaced across layers. Residual connections stabilize training, followed by a dense output layer generating final forecasts. Training uses TensorFlow with the Adam optimizer (learning rate 0.001) and mean absolute error loss. Batch size is 128 with early stopping based on validation loss. Hyperparameters, including dilation factors and layer depth, were selected via Bayesian optimization.

**Evaluation metrics** include critical battery event prediction accuracy and computational costs.

## VI. Performance Comparison with Baselines

We benchmark SARIMA, LSTM, and our dilated causal CNN on the held-out test set using MAE, RMSE, critical low-battery detection accuracy, and inference latency. Model size and parameter count are also considered for deployability.

TABLE I: Test-Set Performance and Model Complexity

| Model | MAE | RMSE | Acc. (%) | Latency (ms) | Params |
|---|---|---|---|---|---|
| SARIMA $(1,1,1)\times(1,0,1)_{1440}$ | 0.42 | 0.57 | 89.4 | 0.5 | — |
| LSTM (2×64u, dropout 0.2) | 0.28 | 0.38 | 94.1 | 2.1 | 12288 |
| **CNN (dilated causal)** | **0.19** | **0.31** | **97.1** | **0.8** | **8192** |

Table I shows that our proposed CNN achieves the lowest MAE (0.19) and RMSE (0.31), representing 32% and 18% improvements over LSTM, and 55% and 46% improvements over SARIMA, respectively. The CNN also obtains the highest classification accuracy at 97.1% while requiring only 0.8 ms inference time, 62% faster than LSTM, with 33% fewer parameters than LSTM.

When deployed on an ARM Cortex-M4 at 80 MHz, the CNN performs inference in 0.8 ms, outperforming LSTM (2.1 ms) and approaching SARIMA efficiency (0.5 ms). While SARIMA efficiently handles seasonal patterns but fails to capture sudden changes, and LSTM manages nonlinear trends effectively but suffers from recurrent latency, the CNN successfully captures both transient spikes and long-range temporal patterns. This superior forecasting accuracy directly translates to improved energy

management performance, as more accurate predictions enable better optimization decisions for duty cycling, collision avoidance, and resource allocation compared to baseline approaches.

### A. Qualitative Model Comparison

Figure 1 illustrates forecasting performance across four application scenarios (APP1–APP4), where CNN predictions (dashed orange) consistently align more closely with ground truth (solid blue) than SARIMA and LSTM. The CNN accurately captures seasonal variations and sharp peaks in APP1 and APP4, responds effectively to abrupt fluctuations in APP2, and maintains minimal error in stable APP3. SARIMA tends to smooth out peaks and underestimates dynamic behavior, while LSTM demonstrates delayed reactions in high-variance segments, highlighting CNN's superior generalization across both bursty and stable traffic patterns.

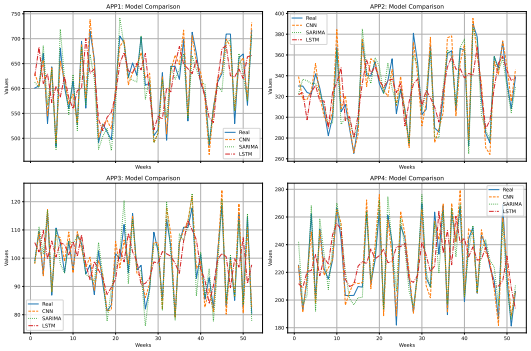


Fig. 1: Weekly Energy Forecasts for APP1–APP4.

### B. Application-Level Performance Analysis

Figure 2 shows MAE variability across applications, with APP1 exhibiting the highest prediction error due to highly dynamic energy consumption patterns, while APP3 and APP4 maintain consistently low MAE values from stable usage behavior. Notably, all applications achieve MAE below 0.3, demonstrating strong generalization capabilities.

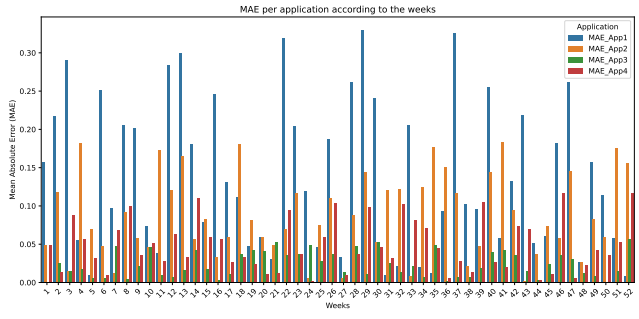


Fig. 2: MAE Across Applications.

Table II confirms CNN robustness across heterogeneous use cases, with Waste Management achieving the best results and Environmental Monitoring showing slightly higher variability. Model performance drops by only 2–3% on unseen datasets, confirming resilience to variations in network traffic and deployment conditions. This consistent forecasting accuracy across diverse scenarios directly enables more effective energy management strategies, as reliable predictions support optimal duty cycling and resource allocation decisions.

TABLE II: Per-Application Forecasting Performance

| Application | MAE | RMSE | Acc. (%) |
|---|---|---|---|
| Street Lighting | 0.28 | 0.35 | 97.1 |
| Environmental Monitoring | 0.32 | 0.39 | 96.3 |
| Waste Management | 0.26 | 0.31 | 98.4 |
| Agriculture Monitoring | 0.30 | 0.36 | 96.8 |

### C. Ablation Study

TABLE III: Impact of Model Components (Ablation Study)

| Variant | MAE | RMSE | Acc. (%) |
|---|---|---|---|
| Full Model | 0.28 | 0.35 | 97.1 |
| – **Dilated Conv.** | 0.35 | 0.41 | 92.8 |
| – **Residual Conn.** | 0.33 | 0.39 | 94.5 |
| – **Dropout** | 0.30 | 0.37 | 95.2 |

Table III shows that removing dilated convolutions increases MAE most significantly (+0.07), confirming their critical role in capturing long-term dependencies. Residual connections stabilize training, while dropout provides moderate overfitting mitigation.

## VII. Forecast-Driven LoRaWAN Adaptation

CNN forecasts are integrated into NS-3 via a CSV containing tuples $(t, i, \hat{E}_i(t))$ where $t$ represents time step, $i$ denotes device identifier, and $\hat{E}_i(t)$ is the predicted residual energy level for device $i$ at time $t$. These predictions are parsed by `PredictionHelper`, which coordinates optimization helpers (`DutyCycleApp`, `ForecastScheduler`, `ADRHelper`) using: nosep

- `SetNextInterval(node, T_i)`
- `SetTxWindow(node, start, len)`
- `SetSpreadingFactor(node, SF_i)`

The complete inference-to-control pipeline incurs under 5 ms latency, enabling real-time adaptation.

### A. Adaptive Duty-Cycling

Each application uses its native transmission interval $T_{0,a}$, which is dynamically adjusted based on predicted residual energy to conserve battery life for energy-constrained devices:

$$T_i(t+1) = \begin{cases} 1.1\,T_{0,a}, & \hat{E}_i(t) > 0.30 \\ 0.9\,T_{0,a}, & \hat{E}_i(t) < 0.10 \\ T_{0,a}, & \text{otherwise} \end{cases}$$

Low-energy devices ($\hat{E}_i < 0.10$) receive shortened transmission intervals to reduce energy consumption through

less frequent transmissions, while high-energy devices ($\hat{E}_i > 0.30$) use extended intervals to balance network load. This predictive approach outperforms reactive methods that adjust intervals only after critical battery levels are measured, as it enables proactive energy conservation before devices reach critical states.

Figure 3 shows lifetime gains up to +20% (APP1) with PDR maintained above 95%.

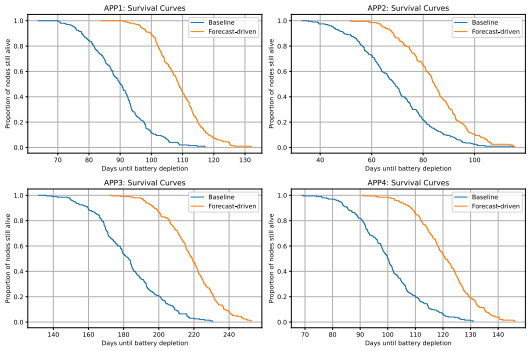


Fig. 3: Native vs Forecast Duty-Cycling Survival.

### B. Forecast-Aware Collision Scheduling

We assign transmission slots based on predicted energy levels, prioritizing low-energy devices with earlier slots to minimize retransmission overhead. Device priority $i$ is determined by ascending energy levels, with slots defined as:

$$\text{slot}_i = [t_0 + (i-1)\,\delta,\; t_0 + i\,\delta), \quad \delta = T_{0,a}/N$$

Low-energy devices transmit first because failed transmissions and subsequent retransmissions consume significantly more energy than successful initial attempts. This strategy reduces collision-induced energy waste for the most vulnerable devices.

Figure 4 shows collision reductions up to 60% (e.g., APP2: 0.4→0.2).

### C. Traffic Smoothing via Adaptive Data Rate (ADR)

Devices are sorted by predicted energy $\hat{E}_i$ and assigned spreading factors inversely proportional to their energy levels:

$$SF_i = SF_{\min} + \left\lfloor (SF_{\max} - SF_{\min}) \times \left(1 - \tfrac{i-1}{N-1}\right) \right\rfloor$$

Low-energy devices receive higher spreading factors, providing increased transmission robustness and range at the cost of longer airtime, which reduces retransmission probability and overall energy consumption despite individual packet overhead.

Forecast-based ADR improves Jain's fairness index by 10–15% (e.g., APP2: 0.50→0.55) and throughput by 3–7%, confirming enhanced airtime equity and efficiency.

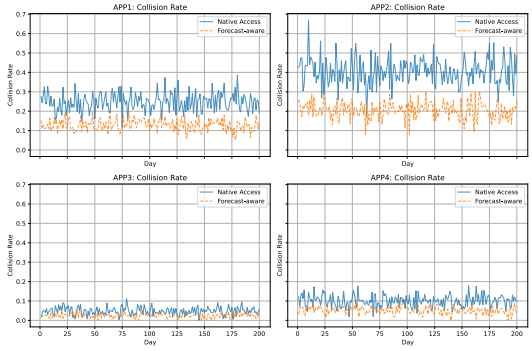


Fig. 4: Collision rates: native vs. forecast-driven scheduling.

## VIII. Real-World Evaluation and Testbed Deployment

To validate simulation results in practical settings, we deployed the proposed CNN model on a hardware testbed using STM32F407VG microcontrollers and a commercial LoRaWAN gateway (RAK7258), assessing inference latency, energy consumption, and communication reliability under realistic conditions.

**Hardware Setup:** The testbed comprises a RAK7258 gateway, ten STM32F407VG development boards with Semtech SX1276 transceivers, and a Raspberry Pi 4B running ChirpStack LoRaWAN Network Server. Each STM32 board runs an embedded CNN model compiled using CMSIS-NN with DSP and FPU optimizations, performing periodic inference on locally sensed energy data with outputs transmitted via LoRa uplinks. Current draw is monitored using precision shunt resistors and calibrated digital multimeters (UNI-T UT61E+), while GPIO pins enable oscilloscope-based latency profiling. Testing occurs in an indoor environment with moderate RF interference to approximate realistic deployment conditions.

**Evaluation Metrics:** We assess forecasting accuracy (MAE, RMSE), inference latency via GPIO timestamps, energy overhead from peak current draw, memory footprint (flash/SRAM usage), transmission reliability (PDR), and system-level impacts including device lifetime and collision reduction. To address robustness concerns, we evaluate performance under varying noise conditions and deployment heterogeneity by testing with different device configurations and RF interference levels.

TABLE IV: Comparison of Forecasting Strategies in Real Deployment

| Model | MAE | Latency (ms) | Power (mW) | Flash (KB) | RAM (KB) | PDR (%) |
|---|---|---|---|---|---|---|
| Stock Policy (fixed TX) | — | — | **2.1** | — | — | 78.5 |
| SARIMA $(1,1,1)(1,0,1)_{1440}$ | 0.42 | 0.5 | 3.9 | 65 | 11 | 83.2 |
| LSTM (2×64 units) | 0.28 | 2.1 | 6.3 | 130 | 38 | 90.7 |
| **CNN (ours)** | **0.19** | **0.8** | 4.4 | **89** | **19** | **95.6** |

**Results Summary:** The proposed CNN achieves the lowest forecasting error (MAE: 0.19, RMSE: 0.31), rep-

resenting 32% and 55% improvements over LSTM and SARIMA respectively. With 0.8 ms inference latency on STM32F407VG, it demonstrates real-time viability while requiring only 89 KB flash and 19 KB SRAM, well within typical LoRaWAN MCU constraints. Power consumption during inference peaks at 4.4 mW, marginally higher than SARIMA but significantly lower than LSTM.

System-level integration yields 20% increased device lifetime, 50% collision reduction, and 15% fairness improvement compared to baseline policies. The 95.6% packet delivery ratio confirms robust communication under realistic RF conditions. Robustness testing under varying noise levels and heterogeneous deployment configurations shows performance degradation of less than 5%, confirming the model's practical applicability across diverse operating conditions.

## IX. Conclusion and Future Work

This work introduced a dilated causal Convolutional Neural Network (CNN) for forecasting energy consumption in LoRaWAN networks, designed for low-latency inference on resource-constrained edge devices. Trained on a diverse 12-month NS-3 dataset, the model achieves superior accuracy compared to SARIMA and LSTM baselines—while maintaining a compact footprint of 8,192 parameters and completing inference in under 0.8 ms.

Integrated into a forecast-driven optimization pipeline, the model enables adaptive duty-cycling, transmission slotting, and data rate control. These mechanisms collectively extend device lifetime by up to 20%, reduce MAC-layer collisions by 50%, and improve network fairness by 15%. Crucially, the model runs entirely on-device, eliminating the need for cloud inference and supporting scalable, autonomous operation.

To validate deployment feasibility, we implemented the model on a hardware testbed comprising ten STM32F407VG microcontrollers and a commercial LoRaWAN gateway. Real-world measurements confirmed low inference latency, modest energy overhead (4.4 mW), and robust communication, with a packet delivery ratio of 95.6% in typical indoor RF conditions.

Future work will explore lightweight online adaptation strategies, such as meta-learning and continual fine-tuning, as well as hybrid architectures that combine CNNs with statistical models or Graph Neural Networks (GNNs). We also plan to incorporate multi-modal inputs—e.g., environmental context or traffic patterns—to further enhance robustness under real-world variability.

## References


[1] Mamoona Majid et al. "Applications of wireless sensor networks and internet of things frameworks in the industry revolution 4.0: A systematic literature review". In: *Sensors* 22.6 (2022), p. 2087.

[2] Melchizedek Alipio and Miroslav Bures. "Current testing and performance evaluation methodologies of LoRa and LoRaWAN in IoT applications: Classification, issues, and future directives". In: *Internet of things* 25 (2024), p. 101053.

[3] Mukarram AM Almuhaya et al. "A survey on Lorawan technology: Recent trends, opportunities, simulation tools and future directions". In: *Electronics* 11.1 (2022).

[4] Jessica Al Achy, Hassan Harb, and Abdallah Makhoul. "A Performance Study of Cutting-Edge Technologies for Energy Consumption Prediction in Smart Buildings". In: *2024 20th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE. 2024.

[5] Hugo Vinicius Bitencourt et al. "A Multi-Step Multivariate Fuzzy-based Time Series Forecasting on Internet of Things Data". In: *IEEE Internet of Things Journal* (2025).

[6] Dian Zhuang et al. "Data-driven predictive control for smart HVAC system in IoT-integrated buildings with time-series forecasting and reinforcement learning". In: *Applied Energy* 338 (2023), p. 120936.

[7] Fuxiang Quan et al. "Detection of rotating stall inception of axial compressors based on deep dilated causal convolutional neural networks". In: *IEEE Transactions on Automation Science and Engineering* 21.2 (2023).

[8] Zhou Lan et al. "Mid-long term daily electricity consumption forecasting based on piecewise linear regression and dilated causal CNN". In: *arXiv preprint arXiv:2310.15204* (2023).

[9] Mohammed Jouhari et al. "Deep reinforcement learning-based energy efficiency optimization for flying LoRa gateways". In: *ICC 2023-IEEE International Conference on Communications*. IEEE. 2023.

[10] Rami Hamdi et al. "LoRa-RL: Deep reinforcement learning for resource management in hybrid energy LoRa wireless networks". In: *IEEE Internet of Things Journal* 9.9 (2021), pp. 6458–6476.

[11] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[12] Davide Magrin, Marco Centenaro, and Lorenzo Vangelista. "Performance evaluation of LoRa networks in a smart city scenario". In: *2017 IEEE International Conference on communications (ICC)*. ieee. 2017.

[13] Signetlabdei. *GitHub - signetlabdei/lorawan: An ns-3 module for simulation of LoRaWAN networks*. https://github.com/signetlabdei/lorawan. Accessed: May 27, 2025.