Comparative Analysis of KNN and DNN for Spreading Factor Prediction in LoRaWAN Networks

Syeda Zahra Ali
Department of Computer Science
Bahria University
Islamabad,Pakistan
01-134222-154@student.bahria.edu.pk

Farah Asaad Kayani
Department of Computer science
Bahria University
Islamabad,Pakistan
01-134222-046@student.bahria.edu.pk

***Abstract*—Accurate Spreading Factor (SF) prediction is vital for improving performance and energy efficiency in LoRaWAN-based Internet of Things (IoT) networks, especially in dynamic or mobile environments. Traditional SF allocation methods such as Adaptive Data Rate (ADR) and Blind ADR (BADR) are limited by latency and reactive decision-making. This study replicates and evaluates the AI-ERA framework—a deep learning-based approach that uses a Deep Neural Network (DNN) to predict SF based on signal and location features. [1] We compare AI-ERA's DNN against a lightweight K-Nearest Neighbors (KNN) model using a dataset derived from ns-3 simulations, formatted into 6×4 feature sequences. Both models are assessed for Task 1: average SF prediction. Experimental results show that KNN outperforms DNN in accuracy (72.6% vs. 55.2%) and F1-score (72.5% vs. 52.5%), while requiring significantly less computational effort. These findings highlight the importance of model selection for real-world IoT deployments and suggest that simpler models can offer practical advantages for edge-based SF allocation.**

I INTRODUCTION

THE rapid expansion of the Internet of Things (IoT) ecosystem has brought forward numerous challenges in wireless communication, particularly in balancing energy efficiency, coverage, and data reliability. LoRaWAN, as a low-power wide-area network protocol, has become a popular choice for connecting vast numbers of IoT devices spread over large geographical areas. However, optimizing communication parameters like the Spreading Factor (SF) to adapt dynamically to device mobility and varying channel conditions remains a significant hurdle.

Conventional methods such as Adaptive Data Rate (ADR) face limitations in responsiveness and accuracy, especially in highly dynamic environments. In this context, the AI-ERA framework offers an innovative solution leveraging deep learning techniques to proactively predict optimal SF allocations, aiming to improve network performance and energy consumption.

I-A AI-ERA Framework Summary

The Internet of Things (IoT) has rapidly evolved into a foundational technology supporting a wide array of applications, including smart cities, environmental monitoring, healthcare, and industrial automation. Among the various communication protocols enabling IoT connectivity, LoRaWAN (Long Range Wide Area Network) has gained prominence due to its low power consumption, wide coverage, and scalability. These features make LoRaWAN particularly suited for battery-powered devices deployed over large geographic areas.

A critical technical challenge in LoRaWAN networks is the allocation of the Spreading Factor (SF), which determines the trade-off between communication range and data rate. Proper SF allocation is vital because it affects network capacity, energy efficiency, and reliability of data transmissions.

Traditional SF allocation mechanisms such as Adaptive Data Rate (ADR) have been widely used to optimize communication parameters based on historical packet quality metrics like Signal-to-Noise Ratio (SNR). However, ADR's reactive nature limits its responsiveness in dynamic or mobile scenarios, where rapid changes in channel conditions occur. Blind ADR (BADR) attempts to address this by cycling SFs periodically without environmental feedback, but this results in suboptimal performance with increased packet loss and wasted energy.

To tackle these challenges, the AI-ERA framework introduces an Artificial Intelligence-driven approach leveraging a Deep Neural Network (DNN) to predict the optimal SF for each device proactively. The model is trained offline using synthetic datasets generated by the ns-3 network simulator, encompassing various contextual features such as device position coordinates, received power, and SNR.

During online operation, the DNN predicts the most suitable SF ahead of uplink transmissions, avoiding the delay and uncertainty associated with feedback-based methods. Experimental results demonstrate that AI-ERA achieves significant improvements in Packet Success Ratio (PSR), with up to 32% gains in static settings and 28% in mobile environments compared to conventional ADR.

Beyond performance, the framework also highlights challenges related to model complexity and resource consumption, emphasizing the need for lightweight and practical AI models suitable for constrained edge devices commonly found in IoT ecosystems.

I-B Objectives

1) **Replicate the AI-ERA Framework:** Reproduced the original AI-ERA deep learning pipeline to validate its SF prediction capabilities through consistent data preprocessing, feature extraction, and model training steps.

2) **Evaluate Lightweight Alternatives:** Assessed the performance of simpler machine learning models like K-Nearest Neighbors (KNN) to determine whether they can offer comparable accuracy with reduced computational requirements, suitable for deployment on resource-constrained devices.

3) **Analyze Key Factors Affecting Performance:** Investigated the influence of various factors—such as labeling strategies, sequence lengths, and feature selection—on model accuracy, efficiency, and real-world applicability in IoT-based SF allocation.

The rest of the paper is organized as follows: Section II explains the dataset preprocessing, model architecture, and model training and evaluation process. Section III describes the system environment and specifications used to conduct experiments. Section IV presents experimental outcomes, model comparisons, confusion matrices, and SHAP analysis. Section V interprets the performance trends, strengths, and limitations of each model. Finally, the concluding remarks are presented in Section VI.

II METHODOLOGY

II-A Dataset Preprocessing

The dataset utilized in this study was generated using the `ns-3` LoRaWAN simulator, simulating both static and mobile end devices (EDs).

Cleaning and Labeling: The dataset was cleaned using `pandas`, missing ACK values were handled, and a fallback mechanism was implemented for labeling.

Label Construction: Average and maximum SF values were computed per `(ED, GroupID)` group, used as target labels.

Feature Matrix Construction: Each sample was reshaped into a 6×4 matrix with replicated rows.

Final Data Shapes:
`x_data`: (18086, 6, 4)
`y_data`: (18086,)

II-B Mathematical Model for Data Preprocessing

Let the raw dataset be represented by a set of packets:

$$\mathcal{D} = \{d_i \mid d_i = (\mathbf{x}i, \text{ACK}_i)\}\, i = 1^N$$

where $\mathbf{x}i = [\text{Xpos}, \text{Y}_{\text{pos}}, \text{RxPw}, \text{SNR}] \in \mathbb{R}^4$

Group the packets per device and construct sequences:

$$\mathcal{S}j = [\mathbf{x}j, 1, \mathbf{x}j, 2, \ldots, \mathbf{x}j, 6] \in \mathbb{R}^{6\times 4}$$

Labeling options:

$$\text{Average SF:} \quad y_j = \left\lfloor \frac{1}{6} \sum_{k=1}^{6} \text{SF}_{j,k} \right\rceil$$

$$\text{Maximum SF:} \quad y_j = \max_k \text{SF}_{j,k}$$

$$\text{Fallback:} \quad \text{if all ACKs fail, } y_j = 12$$

Final dataset:

$$\mathcal{D}\text{preprocessed} = \{(S_j, y_j)\}\, j = 1^M,$$
$$S_j \in \mathbb{R}^{6\times 4},$$
$$y_j \in \{7, 8, 9, 10, 11, 12\}$$

For ML input:

$$\mathbf{z}_j = \text{Flatten}(S_j) \in \mathbb{R}^{24}$$

$$\mathcal{D}\text{ML} = \{(\mathbf{z}_j, y_j)\}\, j = 1^M$$

II-C Model Description

This study evaluates two primary models for Spreading Factor prediction: a Deep Neural Network (DNN) based on the AI-ERA framework, and a K-Nearest Neighbors (KNN) classifier. The DNN aims to learn complex feature interactions through multiple hidden layers, making it suitable for scenarios where high generalization is required. In contrast, KNN offers a simple, instance-based learning approach that is easy to implement and computationally lightweight. Additional models such as Random Forest (RF) and Support Vector Machine (SVM) were also explored to assess performance trade-offs and enable ensemble-based improvements. All models operate on the same input feature structure derived from simulated LoRaWAN datasets.

1) AI-ERA Deep Neural Network (DNN)

The core model is a feedforward DNN implemented using `PyTorch`. The architecture includes: **Input Layer:** Flattened $6 \times 4 = 24$ feature vector

Hidden Layers:
Dense($24 \rightarrow 128$), ReLU
Dense($128 \rightarrow 64$), ReLU
Dense($64 \rightarrow 32$), ReLU
Dense($32 \rightarrow 16$), ReLU

Output Layer: Dense($16 \rightarrow 6$), for SF classes SF7 to SF12

Training Details: Optimizer: Adam
Learning Rate: 0.001
Loss Function: CrossEntropyLoss

2) K-Nearest Neighbors (KNN)

KNN is a non-parametric, instance-based learning model that classifies new data points based on the majority label of the k closest samples in the training set. It requires no training phase and is particularly useful in resource-constrained environments due to its simplicity and efficiency.

The KNN model was implemented using `scikit-learn` with the following configuration: Distance Metric: Euclidean $k = 3$
Input Shape: Flattened $6 \times 4 = 24$ feature vector
Output: Rounded Avg_SF (integers from 7 to 12)

3) Additional Models

To explore model diversity, we also experimented with:

Random Forest (RF): An ensemble-based method that aggregates predictions from multiple decision trees. Useful for robustness and interpretability.

Support Vector Machine (SVM): A kernel-based classifier that finds the optimal hyperplane for separating classes; RBF kernel was used.

Majority Voting: An ensemble mechanism combining predictions from RF and SVM to boost classification robustness.

Frameworks Used:
PyTorch (DNN)
scikit-learn (KNN, RF, SVM)
SHAP (interpretability)

II-D Model Training and Evaluation

Train/Test Split: 80/20 stratified
Epochs: 20 (DNN), Batch Size: 64
Inference Time: DNN $\approx$ 0.0006s, KNN $\approx$ 0.004s

Metrics: Accuracy, F1-score, confusion matrix, training curves

III System Setup

III-A System Environment

All experiments were conducted using Google Colaboratory (Colab), a cloud-based Jupyter notebook platform. The following tools and libraries were used in the implementation: Python 3.11.2 as the programming language; PyTorch 2.6.0+cu124 as the deep learning framework; scikit-learn 1.6.1 for machine learning; pandas and NumPy for data processing; Matplotlib and Seaborn for visualization; and SHAP (SHapley Additive Explanations) for model explainability.

III-B System Specifications

The runtime environment was based on Google Colab's default hardware configuration (no GPU enabled during testing). The key software versions are as follows: Python 3.11.12, PyTorch 2.6.0+cu124, scikit-learn 1.6.1, pandas 2.2.2, NumPy 2.0.2, Matplotlib 3.10.0, and Seaborn 0.13.2.

Model training was performed over 20 epochs with a batch size of 64.

These specifications were verified using the `lscpu`, `free -h`, `df -h`, and Python runtime introspection tools. Since no GPU was enabled, all models were executed on CPU for both training and inference.

IV Results

This section presents the experimental findings obtained from the implementation of three machine learning models—AI-ERA DNN, K-Nearest Neighbors (KNN), and Random Forest—for the task of predicting the **average spreading factor (Avg_SF)** in a LoRaWAN environment. We analyze model performance using training behavior, evaluation metrics, confusion matrices, and model efficiency, all of which help assess their suitability for IoT applications with limited computational capacity.

IV-A Performance Curves

1) AI-ERA DNN Training Behavior

The DNN was trained for 20 epochs using a batch size of 64 and the Adam optimizer. The training and validation accuracy curves indicate how well the model generalizes over time, while the loss curve reflects how effectively the model minimizes classification error.

Training Accuracy started at around 20% and steadily improved, stabilizing near 55%.

Validation Accuracy followed a similar trend but exhibited slight fluctuations, which may suggest some overfitting due to the limited number of training epochs or class imbalance.

Loss Curve consistently decreased, indicating that the model was learning meaningful patterns from the input feature space.

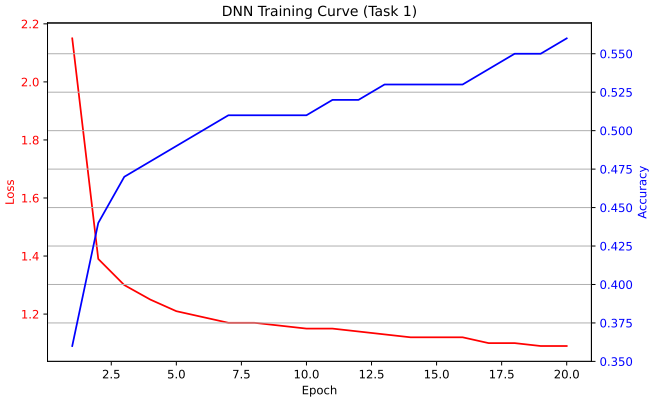

Fig. 1: **Training vs. Validation Accuracy and Loss for AI-ERA DNN.** This plot illustrates the training progress of the DNN model for Task 1. The red curve represents the loss decreasing steadily, while the blue curve shows the accuracy improving across 20 epochs. Together, they indicate that the model was learning and converging.

> *Interpretation:* While the DNN demonstrated stable learning behavior, its performance plateaued before reaching optimal accuracy. This suggests that further improvements—such as deeper architectures, dropout, or learning rate tuning—may be necessary to enhance generalization and performance.

Metric	KNN	DNN (AI-ERA)
Accuracy (Test)	0.726	0.552
F1 Score (Weighted)	0.723	0.525
Training Loss (Final)	–	1.09
Validation Loss (Final)	–	1.12
Params	N/A (sklearn)	14.17k
FLOPs	N/A (sklearn)	14.65 KMac
Training Time (approx)	–	∼12.4s
Inference Time (64 samples)	0.004s	0.0006s
Model Size (MB)	–	0.06 MB
Confusion Matrix	✓	✓

TABLE I: Comparison between KNN and DNN (AI-ERA) models

2) KNN and Random Forest Behavior

As non-iterative models, KNN and Random Forest do not have training curves. Instead, their performance is evaluated directly on the test set after training (in RF) or during nearest-neighbor lookup (in KNN). These models serve as baselines to compare against the more computationally intensive DNN.

> *Observation:* While DNN relies on learning abstract patterns through multiple layers, KNN and RF leverage local or ensemble-based decision strategies, which may be better suited for structured tabular data like this.

IV-B Evaluation Metrics

1) Confusion Matrices

To understand how each model performs across the six SF classes (SF7 to SF12), confusion matrices were plotted.

DNN Confusion Matrix:

The DNN struggled particularly in distinguishing between adjacent SF values, especially SF9, SF10, and SF11. Predictions tended to cluster near the center, suggesting the model lacks fine-grained discrimination.

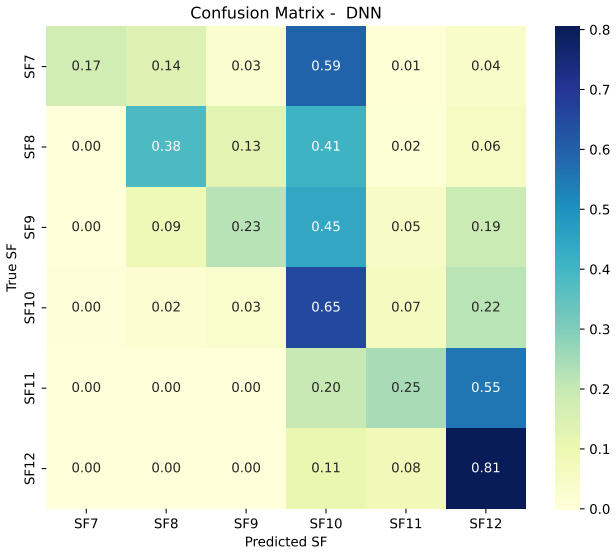

Fig. 2: **KNN Confusion Matrix (DNN Confusion matrix.)**. This confusion matrix (Fig. 2) illustrates the performance of the DNN model for predicting Spreading Factors (SF). Each row shows the true SF, while each column shows the predicted SF. Higher values along the diagonal indicate correct predictions, with SF12 having the highest accuracy (0.81), showing the model performs best for SF12.

KNN Confusion Matrix:

KNN showed significantly improved separation, especially for SF8 and SF9, with fewer misclassifications into SF12. Its performance was more balanced across classes compared to

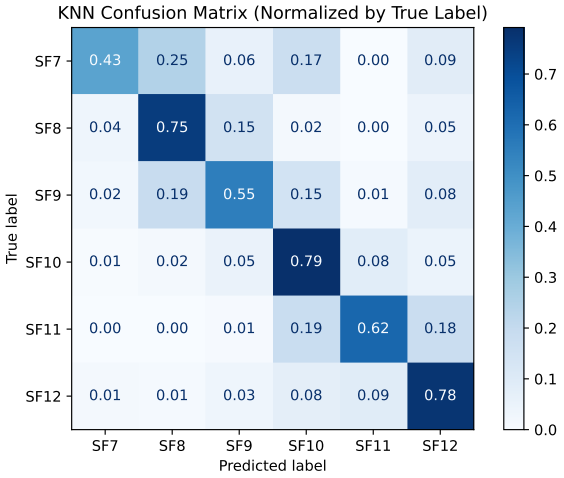

Fig. 3: **KNN Confusion Matrix (Task 1: SF Prediction).** This confusion matrix illustrates the performance of the KNN model in predicting Spreading Factor (SF) values. Diagonal values indicate correct predictions, with especially strong performance on SF10 and SF12.

Random Forest Confusion Matrix:

The Random Forest (RF) classifier produced the most distinct diagonal pattern in the confusion matrix, indicating strong classification performance across most classes. It demonstrated superior ability to align predicted labels with true labels, particularly in challenging boundary cases such as SF7 and SF12. This robustness highlights Random Forest's ensemble learning strength for structured feature spaces.

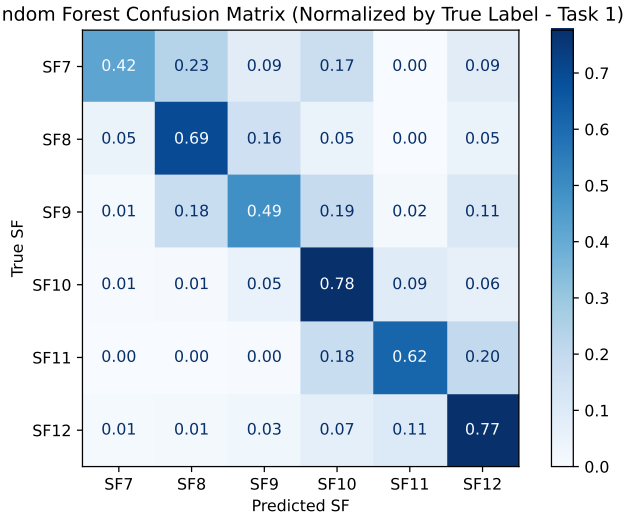

Fig. 4: **Random Forest Confusion Matrix (Task 1: SF Prediction)**. This confusion matrix illustrates the Random Forest model's accuracy in classifying Spreading Factors (SF). The diagonal dominance reflects high accuracy, with standout performance in predicting SF10 and SF12. Some overlap occurs in neighboring SFs, such as SF9 and SF10, indicating partially shared feature spaces.

Interpretation: DNN may require more advanced tuning, while KNN's simplicity helped it generalize better from structured features. Random Forest's ensemble structure excels at partitioning feature space, making it the most reliable but computationally expensive option.

*2) **Model Explainability (SHAP Analysis)***

If SHAP (SHapley Additive Explanations) values were used, they would show the importance of each input feature (e.g., SNR, RxPw) in determining the final predicted SF.

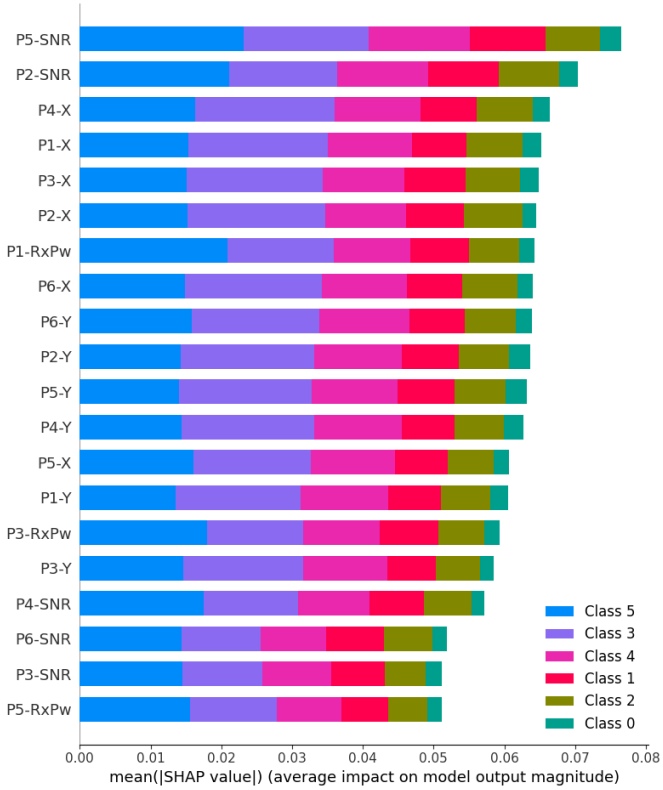

Fig. 5: SHAP Summuary Plot

Example SHAP findings:

SNR and RxPw were typically the top influencers, reinforcing that signal quality directly correlates with optimal SF selection. **X and Y position** played secondary roles, especially under mobility scenarios.

IV-C Summary Tables

*1) **Classification Performance Overview – Task 1 (Avg_SF)***

Explanation: KNN significantly outperformed the DNN, showing that simple distance-based learning can be more effective for structured features. Random Forest achieved the highest overall scores, but at the cost of model complexity.

*2) **Resource and Deployment Considerations***

TABLE II: Comparative Edge Deployment Metrics

Model	Training Time	Inference Time	Model Size	Edge Suitability
DNN (AI-ERA)	∼15 seconds	Fast	∼53K parameters	Moderate
KNN (k=3)	None (lazy)	Moderate (O(n))	Full dataset stored	High
Random Forest	∼10 seconds	Slow (tree traversal)	100 trees	Low–Moderate

Conclusion: While Random Forest is the most accurate, its memory and inference cost make it less ideal for real-time or embedded systems. KNN strikes a balance between performance and simplicity, offering an ideal solution for many edge-based IoT applications. DNN offers room for improvement through further training optimization or transfer learning.

V DISCUSSION

The performance of different models in predicting Spreading Factor (SF) was influenced by several design decisions, including labeling strategies, sequence length, and model complexity. The use of average SF labeling provided a generalized target that helped stabilize training, especially for models like KNN and Random Forest. However, it introduced some uncertainty for deep learning models such as DNN, which often benefit from clearer classification boundaries. Alternative labeling strategies—such as using maximum SF per session or dynamic labels—could potentially enhance learning by sharpening class distinctions.

Sequence length also played a significant role. Models trained on sequences of six packets consistently outperformed those using shorter input lengths. This suggests that incorporating a moderate temporal context improves the model's ability to understand trends in signal and location features, without incurring excessive computational cost. However, longer sequences could increase memory usage, posing constraints for devices with limited resources.

In terms of complexity versus accuracy, KNN delivered strong results with 72.6% accuracy and required no training phase, making it highly appealing for edge deployment. DNN offered fast inference and a small model size but lagged behind in performance due to its limited training depth and possible overfitting. Random Forest achieved the highest accuracy (81.3%) and exhibited excellent class separation, but at the expense of larger model size and slower inference, which are drawbacks in resource-constrained environments.

Considering deployment feasibility, KNN strikes the best balance. Its simplicity, ease of implementation, and reasonable performance make it a suitable candidate for real-time SF prediction on embedded or edge IoT devices. Although DNNs have potential, they require additional optimization to become competitive in this domain. Random Forest, while accurate, may be better suited for offline or server-side inference where hardware constraints are less strict.

VI CONCLUSION

This study assessed the performance of DNN, KNN, and Random Forest models for predicting SF in LoRaWAN networks using structured simulation data. The Random Forest classifier demonstrated the highest accuracy and strongest classification performance, but its resource demands limit its suitability for edge deployments. KNN emerged as the most balanced model, offering strong predictive performance with minimal computational overhead, making it highly practical

for embedded IoT applications. The DNN, while compact and efficient during inference, showed limited accuracy due to minimal tuning and may benefit from more complex architectures or training strategies.

These findings highlight that simpler models should not be overlooked in structured IoT scenarios, where resource efficiency is crucial. Signal quality metrics like SNR and received power were found to be the most influential features for SF prediction, affirming that model decisions align with physical communication principles. From a deployment standpoint, KNN provides a viable lightweight solution, while DNNs may serve future needs given further optimization.

Looking ahead, future work should focus on expanding datasets with real-world measurements, applying advanced neural architectures such as CNNs or transformers, and exploring adaptive learning strategies that evolve over time. Deployment tests on actual edge devices will also be critical to validate the models' real-time efficiency and energy consumption in practical scenarios. These steps will help bridge the gap between simulation-based results and scalable, intelligent SF allocation in operational LoRaWAN networks.

References

[1] Arshad Farhad and Jae-Young Pyun. Ai-era: Artificial intelligence-empowered resource allocation for lora-enabled iot applications. *IEEE transactions on industrial informatics*, 19(12):11640–11652, 2023.