2024 International Conference on Smart Applications, Communications and Networking (SmartNets) | 979-8-3503-8532-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/SMARTNETS61466.2024.10577722

Data-Driven End-to-End Lighting Automation Based on Human Residential Trajectory Analysis

Jack Zhu, Jingwen Tan, and Wencen Wu*
Computer Engineering Department
San José State University
San José, CA, USA
Email: {jiali.zhu, jingwen.tan, wencen.wu}@sjsu.edu

Abstract—**Smart home automation, particularly in lighting, holds the potential to significantly improve comfort, energy efficiency, and security by centralizing control over internet-of-things (IoT) devices. This paper introduces a smart lighting automation system that analyzes human movement trajectories using machine learning, deep learning, and reinforcement learning techniques integrated into the Home Assistant platform. In particular, we introduce a transformer-based deep neural network architecture with reward-based tuning as our backbone model. The system predicts the user's next location and adjusts the lighting accordingly based on the anticipated movement trajectories derived from data collected by IoT devices distributed throughout a residential area. This enhances both convenience and energy efficiency. We deployed the system in a residential setting and conducted experiments to validate its accuracy.**

Index Terms—**Smart Home, Automation, Machine Learning, Transformer**

I. Introduction

The world of smart home automation has witnessed significant innovation facilitated by the increasing computational power, the exponential growth of data generated by smart devices, and breakthroughs in machine learning (ML), deep learning (DL), and reinforcement learning (RL) methodologies [1]. These techniques are the driving force behind the evolution of home automation systems from simple programmable devices to complex, intelligent ecosystems capable of autonomous decision-making to streamline user's daily tasks.

Numerous studies have investigated the use of various sensors such as accelerometers, infrared motion, contact, light, temperature, and humidity sensors for capturing user activities in smart homes [2][3][4]. Those sensors provide comprehensive data regarding user location, activity levels, sequential movement, etc., enabling the development and training of learning models. Using the data collected by IoT devices, diverse methods have been employes to automate smart home applications. For example, an input structure called HAM, introduced in [5], filters conditions across three levels - day, time, and Markov-chain-based sensor level - to define triggers for smart events. In [6], sequence mining in time-related activity was performed by applying a generalized sequential pattern algorithm [6], which categorized users' different activities into subgroups and defined rules that ensure the order of activities. Additionally, researchers have been developing DL models for smart lighting systems. Authors in [7] utilized generative adversarial networks (GANs) to generate additional data and compared the performance of models on various dataset sizes, finding that a gated recurrent unit (GRU) model excels in multi-resident activity recognition. Furthermore, in [8], experiments using a recurrent extreme learning machine (RELM) model demonstrated its effectiveness in predicting users' activities over time, outperforming other models in terms of learning and training error.

Recognizing human trajectories is another critical aspect of smart homes. Authors in [9] introduced a method for recognizing human gesture trajectories in smart homes using DL, specifically employing the squeeze and excitation technique for classifying these trajectories, leading to high accuracy and reliability. This approach ensures that lighting automation is a classification problem, determining which areas of the lights should be activated. Another study [10] focused on predicting future human actions from current actions within non-residential environments. They proposed a recognition-then-prediction framework that divides the problem into two parts: forecasting the future trajectory and estimating its duration using the sigma-lognormal function.

Despite these advancements, current smart home applications such as Amazon Alexa [11] and Google Home [12] still fall short in several aspects. They often lack in-depth intelligence and require manual creation and maintenance of scenes and scripts through if-else brackets. Based on these limitations and inspired by recent research, we propose an enhanced system that predicts the user's next location and adjusts the lighting accordingly based on the anticipated movement trajectories through information collected by internet-of-things (IoT) devices distributed across the residential area.

In summary, the contributions of this work include (1) designing and configuring a data-driven smart lighting system that integrates Zigbee IoT devices with the Home Assistant platform for enhanced data collection and control; (2) developing a transformer-based deep neural network architecture with reward-based tuning, which serves as the backbone model for the smart lighting automation system; (3) deploying the model via a docker container within the local network for inference purposes and a custom Home Assistant platform add-on to integrate the model's prediction with the IoT devices, thereby creating a complete feedback loop; and (4) testing the system

The research work is supported by NSF grant RINGS-2148353.
979-8-3503-8532-8/24/$31.00 ©2024 IEEE

in a residential area to validate the accuracy and effectiveness of the smart lighting system.

The rest of the paper is organized as follows. We initiate our discussion by starting with system design in Section II. Next, we introduce the data collection and preprocessing in Section III, followed by our proposed backbone approaches in Section IV. Evaluation methodology and results are presented in V and deployment and application of the system are discussed in Section VII. The paper is concluded by Section VIII.

II. System Design

In this section, we present the hardware design of the smart lighting system and discuss the close-loop communication among the devices.

We chose to install a smart lighting system in a two-bedroom apartment. The system consists of a closed-loop connection among three major hardware groups, a computer that runs the inference model, an edge hub with Home Assistant OS, and IoT devices that collect and execute environment signals. The Home Assistant platform is employed on an edge device, serving as the central hub for IoT management. These devices include one temperature sensor, four human presence sensors, five motion sensors, four light switches, and two brightness sensors from various brands, all utilizing Zigbee wireless technology. These sensors are located at the intersection between the living room and kitchen to optimize zoning construction. Equipped with millimeter-wave radar technology, the sensors can detect signals even from stationary individuals. The light switches have been strategically replaced with smart switches throughout the residential area. In each major area, human presence sensors are installed at strategic corners to maximize coverage. Brightness sensors are placed near the window, and motion sensors are distributed in the stairs and hallway. The floor plan of the apartment and the layout of device distribution are shown in Fig. 1. The configuration process involves linking the IoT devices with compatible drivers. However, in cases where certain IoT devices were not compatible with the Home Assistant platform, we undertook reverse engineering efforts and created convertible JavaScript files to ensure successful integration.

Fig. 2 illustrates the closed-loop communication cycle of the smart lighting system. As shown in the figure, the Zigbee IoT devices are integrated with Home Assistant through the Zigbee2mqtt add-on, enabling Home Assistant to read and control these devices. We developed a Docker container, named "model-inference add-on", for Home Assistant. This add-on sends HTTP GET requests to a separate Docker container running the deep learning model for prediction responses. Upon receiving these responses, the model-inference add-on adjusts the IoT devices' states within the Home Assistant interface. Subsequently, Home Assistant communicates these state changes to the IoT devices using Zigbee communication and closes the system cycle. The optimal system design would involve integrating the DL model directly within the edge hub, thereby eliminating the need for additional communication with a computer. However, as detailed in the discussion section VII, hardware limitations compel us to seek alternative solutions.

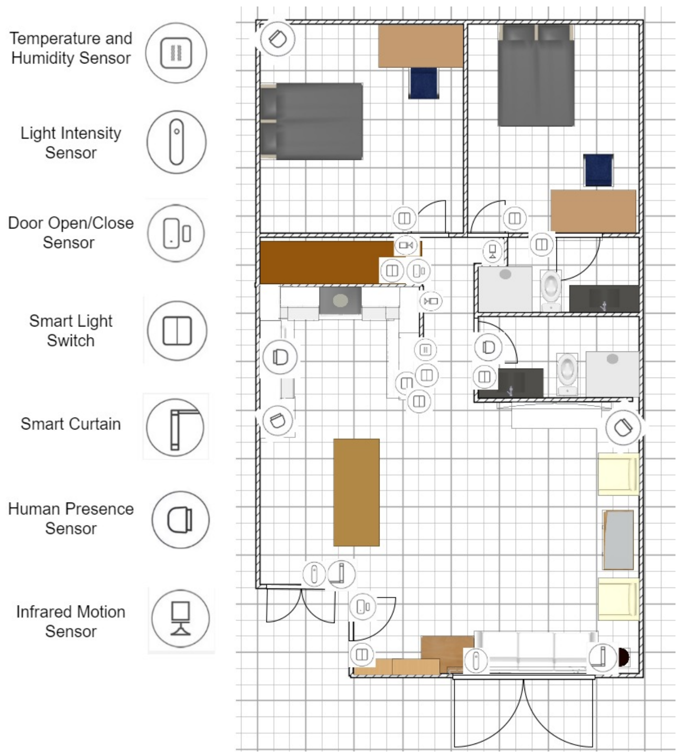

Fig. 1. IoT Device Floor Plan: A detailed layout of the experimental house and related IoT devices.

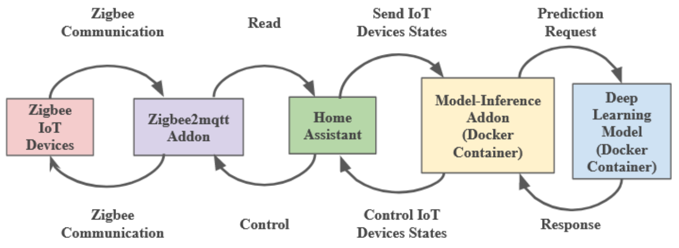

Fig. 2. Lightening System Diagram: A close cycle among model-integrated computer, Home Assistant edge hub, and home IoT devices.

III. Data Collection and Preparation

In this section, we outline the methods used to collect and preprocess data, which will be used by our proposed learning strategy to enable end-to-end lighting automation.

A. Data Collection

The IoT devices installed in the apartment continuously collect a variety of data points related to human presence, the states of smart light switches, brightness sensor readings, and environmental conditions such as time, date, and temperature. This comprehensive dataset includes two distinct types of data. The first type comprises data generated by automation scripts that regulate the behavior of the smart lights based on detection from the human presence sensor, while also considering factors like time of day and ambient brightness levels. The second type of data stems from manual light control actions performed by individual residents, wherein they manually turn lights on and off, thus providing valuable insights into the unique habits and preferences of the occupant. It's noteworthy that the data collection extends over one month within the apartment occupied by two residents. This timeframe is crucial as it allows for an accurate representation of the details embedded in the residents' daily routines and their engagement with the smart home features. Given that the dataset captures the living habits of two individuals, our learning model is designed to find a balanced response that accommodates the needs and preferences of both residents.

B. Data Preprocessing

In this phase, data from IoT devices was stored in the Home Assistant database. The data was then extracted and organized into a structured data frame format to facilitate efficient filtering of relevant features. To empower the model's understanding of time, we segmented the datetime information into month, day, hour, minute, and second. We extracted categorical features from the datetime information. The day of the week is denoted as d_w, where $d_w \in \{0, 1, 2, 3, 4, 5, 6\}$, with each number corresponding to a day within a week, starting from 0 for Sunday, 1 for Monday, and so on. We categorized the time of day into three classes: morning as 0, afternoon as 1, and evening as 2. For binary states, off/on and $noperson/personpresent$ labels are converted to numerical representations: "off" and "no person" are encoded as 0, while "on" and "person present" is represented as 1. Likewise, classes denoted as "low", "middle", and "high" are transformed into numerical equivalents: "low" as 0, "middle" as 1, and "high" as 2. This transformation streamlined the categorical data into numerical representations suitable for computational analysis.

We selected 35 features to make informed lighting decisions. These features encompass data from IoT devices, including human presence sensors, brightness sensors, the state of smart light switches, and various environmental variables. The selection of these features is thoughtfully made to capture crucial factors influencing lighting control. To enhance understanding of our training features, we have selectively included key representative features from each category in Table I. As for the output data that shown in Table II[1], it primarily consists of lighting states for different areas, specifically the bedroom, dining room, living room, and kitchen.

To prepare the data for model training, we batched the time series data into segments of 10 over 139066 rows, with the corresponding next event serving as the true label, thus formatting the training features and labels. The final training set input and label shapes are $(100000, 10, 35)$ and $(100000, 4)$, validation set input and label shapes are $(19990, 10, 35)$ and $(19990, 4)$, and testing set input and label shapes are respectively $(19066, 10, 35)$ and $(19066, 4)$. The model fine-tuning data set shapes are $(584, 35)$.

IV. Backbone Method and Design

In this section, we introduce a transformer-based deep neural network architecture with reward-based tuning as our backbone model as shown in Fig.3.

[1]The table only reflects a portion of the entire input feature set. Each category has one specific input feature being selected as a representative.

TABLE I
TABLE FOR THE SELECTIVE INPUT FOR THE MODEL TRAINING: THE INPUT FEATURES HAVE 7 MAJOR CATEGORIES.

Input Feature Category	Features Name	Description	Value
Brightness sensor	sensor.dinning_room_ illumination_intensity_ brightness_state	Indicate the brightness State of the dinning room and kitchen areas	[0,2]
Time	day	Describe the day of the observation	[0,31]
Human Presence Sensor	binary_sensor.dinning_ room_sensor_presence	Detect if people remain in the dining room	0 = False 1 = True
Infrared Motion Sensor	binary_sensor.hallway_ motion_and_intensity_ sensor_occupancy	Detect if people pass by the hallway	0 = False 1 = True
Door Sensor	switch.open_close	Detected if the garage door is open or close	0 = Close 1 = Open
Light Switch	switch.front_door_ light_switch	Front door light	0 = off 1 = on
Dimmer Brightness	sensor.living_room _dimmer_brightness	Indicate the living room dimmer brightness .	0 = min 1 = max

TABLE II
TABLE FOR THE OUTPUT FROM THE MODEL: LIGHT OUTPUTS ARE BASED ON THE DIFFERENT LIVING AREAS IN THE HOUSEHOLD

Light Category	Feature Name	Description	Value
Light Dimmer	light.dining	Dining room light	0 = off, 1 = on
Light Dimmer	light.living	Living room light	0 = off, 1 = on
Light Switch	Bedroom_sensor	Bedroom room light	0 = off, 1 = on
Light Switch	switch.kitchen	Kitchen light	0 = off, 1 = on

A. Deep Neural Network Architecture

The deep learning neural network architecture serves as the cornerstone for processing raw input data and generating lighting decisions for the event of 5 seconds after. Our chosen architecture is based on a transformer structure, specifically the encoder portion. Given that our input data is a batch of time series data, we employed an LSTM layer to embed input tensor $\mathbf{X}$ where $batch\_size = 256$, $window\_size = 10$, and $feature\_size = 35$, is meticulously chosen to capture temporal dependencies inherent in human movement and lighting patterns. The batch size ensures efficient processing of data, the window size aligns with the granularity of movement within a home environment, and the feature size corresponds to the diverse data points collected, providing a comprehensive representation of the environment.

The LSTM's capacity to maintain a sequence of hidden states $H = (h_1, h_2, ..., h_{window\_size})$ is crucial for capturing the sequential nature of human activities and their impact on lighting decisions. The specific equations governing the LSTM layer:

$$C_t = tanh(W_c * [h_t - 1, x_t] + b_C), \tag{1}$$

$$C_t = f_t * C_{t-1} + i_t * C_t, \tag{2}$$

$$h_t = o_t * tanh(C_t), \tag{3}$$

are tailored to encapsulate the temporal dynamics and dependencies, where C_t is the cell state. f_t, i_t, o_t are forget, input and output gate at time t. W_c and b_C are weight matrices and bias vector.

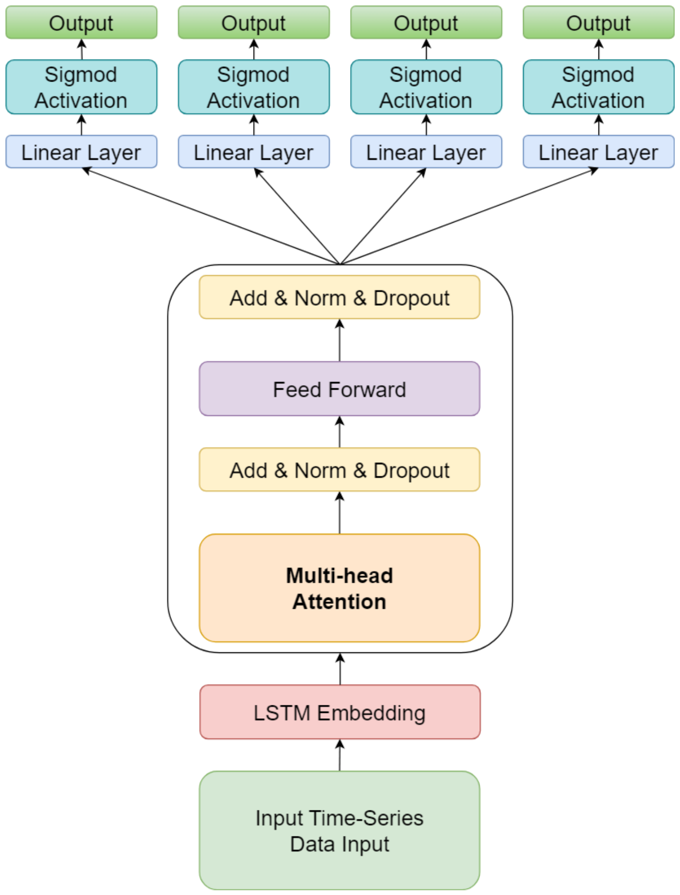

Fig. 3. The proposed deep learning model diagram: The model architecture consists of an LSTM-based embedding layer, followed by a series of transformer encoder layers, each comprising a multi-head attention mechanism, linear layers, layer normalization, and dropout. The model also includes multiple linear output heads and a final sigmoid activation function, allowing for complex data transformations and classification tasks.

Subsequently, The LSTM's output is further processed by a multi-head action module with $h = 4$. The attention mechanism, as calculated in Equation (4), was introduced by Vaswani et al. [13]:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{4}$$

where Q, K, V represent the query, key, and value vectors fundamental to the attention mechanism, facilitating the model's focus on relevant input data for context interpretation. Then, a module list that uses a linear layer and sigmoid activation was employed to analyze each output of the four classes from the multi-head module's pooled information to yield intermediate outputs I_i:

$$I_i = \sigma(W_{\text{ff},i} \cdot A + b_{\text{ff},i}), \tag{5}$$

where $W_{\text{ff},i}$, A, and $b_{\text{ff},i}$ are the weight matrix, activation outputs, and bias terms, which processes the attention-weighted inputs to produce subsequent layers' activations. Lastly, a transformer encoder-based classifier is used to produce an array of four variable outputs. Each of these variables undergoes normalization via a sigmoid function to predict the lighting state:

$$P_i = 1/(1 + e^{-}O_t), \tag{6}$$

where P_i is normalized output probability for each class i, ready for the lighting decision.

Selecting appropriate hyperparameters is vital for optimizing the model's performance. Therefore, we calibrated the following hyperparameters to train our model:

- Epochs: 100
- Batch size: 256
- LSTM layers: 2
- Number of heads (nhead): 4
- Learning rate: 0.0005
- Number of transformer layers: 1
- Embedding dimension (embedd_d): 8
- Dimension of feedforward network: 16
- Dropout rate: 0.3

These hyperparameters were chosen to balance the model's learning capacity with computational efficiency. We have experimented with different parameters, and this combination is giving the optimal training performance.

B. Human Feedback Reward-Based Mechanism Fine-Tuning

The initial phase of model fine-tuning involves utilizing human feedback to assign labeled rewards based on the comparison between the model's predicted actions A_p and the actual actions A_t, given the state S. The reward function $R(A_p, A_t, S)$ is defined on the set $\{0, 0.25, 0.75\}$, where a reward of 0 is assigned if A_p does not match A_t under any state S. A reward of 0.25 is given if A_p equals A_t during the day, and a reward of 0.75 is allocated for matches during the night. This weighting aims to focus on nighttime light control, intentionally biasing the model to learn more effective lighting policies.

To enhance the model's performance, we employed the Proximal Policy Optimization (PPO) algorithm, which incorporates actor and critic mechanisms as presented in Fig. 4 [14]. These mechanisms are instrumental in optimizing lighting decisions using human-labeled feedback and offline data buffers. Observations for this process are drawn from the offline dataset. The actor model, originating from the initialized deep learning model, defines an action space that includes actions like turning lights on and off. It is then utilized to compute action probabilities based on the processed observation. These probabilities are transformed into Bernoulli distributions, with each distribution representing the likelihood of a specific action in the environment. The ensemble critic network is a stack of 4 critic networks with the shared network module across the critic to evaluate the expected value associated with the observation and the policy. Ultimately, the model learns to select the best actions to minimize the total loss.

V. Evaluation Methodology & Results

In the evaluation, we tested the model's performance based on established classification metrics. For each of the four binary classes corresponding to different lighting areas, we calculate accuracy, precision, recall, and F1 score.

For the evaluation of the deep learning neural network architecture, we raised questions regarding the necessity of

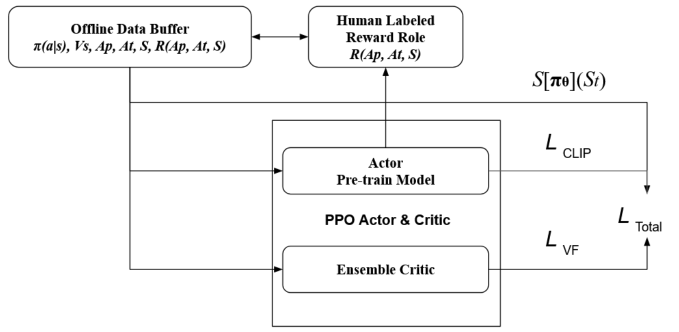

Fig. 4. Reward-based fine-tuning mechanism: The reinforcement model utilizes offline data, actor-critic structure, and human feedback reward.

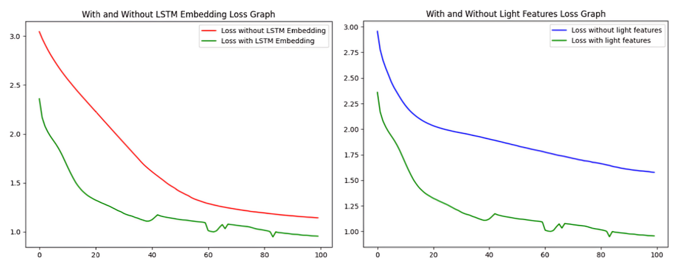

Fig. 5. Evaluation results on transformer-based model: The provided plots depict the training process of a model using the transformer algorithm, focusing on experimenting with LSTM and lighting features that should be integrated into the model.

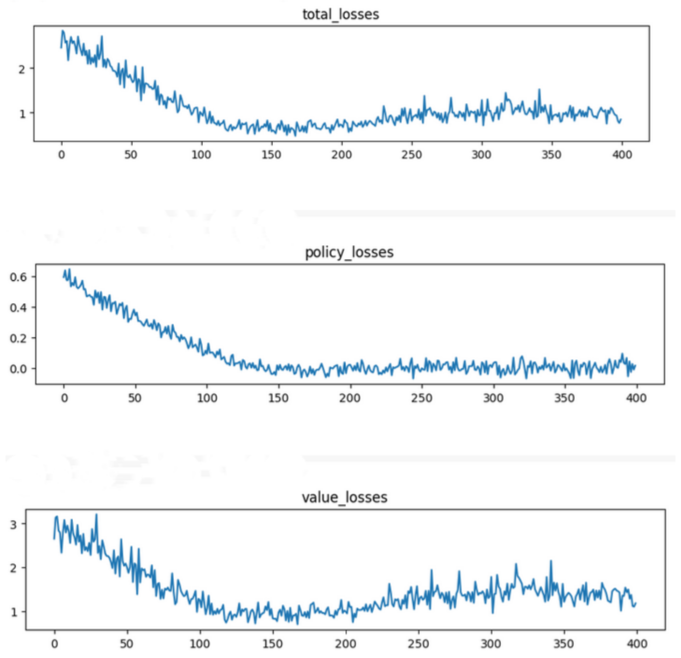

Fig. 6. Evaluation results on PPO fine-tuning: The provided plots depict the training process of a model using the PPO algorithm, focusing on total losses, policy losses, and value losses over a series of training episodes.

including LSTM and lighting features, which are indicative of the current state of switches as model input. As illustrated in Fig. 5, applying LSTM embedding significantly reduces loss, demonstrating its efficacy in handling time-series data for predicting lighting conditions over a 5-second horizon. Additionally, incorporating light features further enhances the model's predictive accuracy. These improvements are crucial for a system tasked with real-time responsiveness to resident behavior and environmental changes. The overall descending trend in loss across the training and evaluation phases signals a reasonable learning process.

The final metrics for the DL models, as delineated in Table III, reveal a stark contrast in performance across different lighting areas within the home automation system. Notably, the model exhibits outstanding accuracy and precision in managing the dining room's lighting conditions, as evidenced by high scores across all evaluated metrics: an Accuracy of 0.979, Precision of 0.977, Recall of 0.938, and an F1 Score of 0.958. These results underscore the model's proficiency in predicting and adjusting the dining room lighting in alignment with occupants' needs and behaviors. Conversely, the model encounters pronounced difficulties in accurately predicting the lighting requirements for the living room, bedroom, and kitchen. This challenge is particularly evident in the drastically low Recall and F1 Scores in these areas, indicating a significant shortfall in identifying when the lights need to be activated. For instance, the Recall scores for the living room, bedroom, and kitchen plummet to 0.024, 0.005, and 0.002, respectively, suggesting a pronounced data bias issue. Such bias, characterized by the uneven distribution of "on" and "off" states within the training dataset, often reflects real-world scenarios where lighting remains predominantly off. Consequently, this imbalance skews the model's learning process towards the "off" state, significantly impairing its ability to accurately predict true positive instances when lighting is required.

TABLE III
PERFORMANCE METRIC FOR DEEP LEARNING MODEL

DL Pre-train	Dining Room Light	Living Room Light	Bedroom light	Kitchen Light
Accuracy	0.979	0.789	0.840	0.809
Precision	0.977	0.853	0.82	0.75
Recall	0.938	0.024	0.005	0.002
F1 Scores	0.958	0.047	0.009	0.003

As for the training progression plots for the PPO-based tuning, it reveal an overarching decline in total losses, indicating successful model learning and optimization in line with enhancing user convenience in lighting conditions. The consistent descent in policy losses, as depicted in Fig. 6, suggests that the actor component is increasingly adept at choosing actions that correlate with higher cumulative rewards, reflecting a growing proficiency in predicting user preferences. However, the value losses exhibit greater fluctuation, suggesting that the critic component faces challenges in grappling with complex data patterns or variability while improving its value estimations. This observation can also be proved by comparing metrics from the two tables. As shown in Table IV, show improvements in accuracy and precision across the dining room, living room, and bedroom experiencing notable enhancements. Despite these advancements, the recall metric remains significantly low for the bedroom and kitchen, continuing to highlight the model's limitations in these spaces.

These findings underscore the persistent challenge of data bias, even after fine-tuning, affecting the model's performance in predicting accurate lighting needs across different home areas.

TABLE IV
PERFORMANCE METRIC FOR FINE-TUNED DEEP LEARNING MODEL

RL Fine-tune	**Dining Room Light**	**Living Room Light**	**Person Room light**	**Kitchen Light**
Accuracy	0.981	0.819	0.840	0.808
Precision	0.977	0.950	1.0	0.08
Recall	0.948	0.169	0.0007	0.0002
F1 Scores	0.962	0.287	0.001	0.0005

Overall, fine-tuning through reinforcement learning has enhanced the model's performance in multi-room scenarios. Nevertheless, the bias performance caused by the imbalanced dataset is still an essential problem to be solved.

VI. Deployment & Application

The deployment centers on the Home Assistant Supervisor, which serves as the primary hub for our IoT ecosystem. This home assistant tool not only acts as a dashboard but also facilitates data storage. To ensure seamless integration and accessibility, we opted for a Docker container to host the model within the local network. This approach enhances security and reduces network latency. The containerization provided an industry-grade model deployment and package library scalability and management. To establish connectivity between our IoT devices and the model, we developed a custom add-on, which is a specialized type of Docker container that is specifically used specifically in the Home Assistant OS. This add-on serves as the bridge, utilizing the states of the IoT devices as input parameters while sending HTTP GET requests to the model's endpoint. This streamlined approach ensures a cohesive interaction between our model and the diverse array of IoT devices within the ecosystem.

VII. Discussions

Our ultimate objective for the home automation system is to ensure precise predictions, efficient responses, and straightforward integration. While our proposed model meets the expected accuracy standards, the integrated system still faces challenges in simplification and real-time responsiveness.

After constructing the system between the learning models and edge devices, a noticeable delay of 3-4 seconds persists in the control process. This delay predominantly arises from the interactions among IoT devices, the Home Assistant hub, and the computer. Reflecting on the architecture illustrated in Fig. 2, the computer processes the inference model using real-time sensor data collected from the hub and sends the predicted outcomes back to the hub for control signals dispatching. This lag in the communication process adversely affects the system's responsiveness, causing a noticeable delay in the final inference. In our initial design, we utilized the edge device as both the host and the model executor. However, we faced challenges in establishing a TensorFlow environment on this Home Assistant edge device. Additionally, using the device solely as a host in repeated inference tests led to overheating issues with the hardware. This discrepancy between the dashboard's representation and the actual execution on the IoT devices highlighted a significant operational challenge within the system.

VIII. Conclusions and Future Work

In conclusion, our project aimed to transform lighting automation by leveraging IoT, deep learning, and reinforcement learning to create energy-efficient and comfortable systems aligned with user needs. While successful in integrating various devices and training our model, real-world challenges like edge device delays and overheating emerged. This underscores the opportunity for enhancing edge device communication, crucial for a smoother and more responsive smart lighting solution in homes. Our project encompassed both hardware and software elements, providing invaluable experience in developing a complete AI-driven edge product cycle, offering hands-on insights from inception to deployment in a real-world scenario. Future work includes refining the learning algorithms to enhance the system's responsiveness and accuracy, and expanding the system to include more diverse environments and lighting conditions.

References

[1] A. G. Putrada, M. Abdurohman, D. Perdana, and H. H. Nuha, "Machine learning methods in smart lighting toward achieving user comfort: A survey," *IEEE Access*, vol. 10, pp. 45 137–45 178, 2022.

[2] D. Cook, "Learning setting-generalized activity models for smart spaces," *IEEE Intelligent Systems*, vol. 27, no. 1, pp. 32–38, 2012.

[3] L. Wang, T. Gu, X. Tao, H. Chen, and J. Lu, "Recognizing multi-user activities using wearable sensors in a smart home," *Pervasive and Mobile Computing*, vol. 7, pp. 287–298, 06 2011.

[4] R. Fritz and G. Dermody, "A nurse-driven method for developing artificial intelligence in "smart" homes for aging-in-place," *Nursing Outlook*, vol. 67, 11 2018.

[5] P. Rashidi and D. J. Cook, "Keeping the resident in the loop: Adapting the smart home to the user," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 39, no. 5, pp. 949–959, 2009.

[6] J. Rosales-Salas, S. Maldonado, and A. Seret, "Mining sequences in activities for time use analysis," *Intelligent Data Analysis*, vol. 24, pp. 339–362, 03 2020.

[7] A. Natani, A. Sharma, T. Peruma, and S. Sukhavasi, "Deep learning for multi-resident activity recognition in ambient sensing smart homes," in *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*, 2019, pp. 340–341.

[8] Z. Liouane, T. Lemlouma, P. Roose, F. Weis, and H. Messaoud, "An improved extreme learning machine model for the prediction of human scenarios in smart homes," *Applied Intelligence*, vol. 48, no. 8, p. 2017–2030, aug 2018. [Online]. Available: https://doi.org/10.1007/s10489-017-1062-5

[9] A. Li, E. Bodanese, S. Poslad, T. Hou, K. Wu, and F. Luo, "A trajectory-based gesture recognition in smart homes based on the ultrawideband communication system," *IEEE Internet of Things Journal*, vol. 9, no. 22, pp. 22 861–22 873, 2022.

[10] Y. Cheng and M. Tomizuka, "Long-term trajectory prediction of the human hand and duration estimation of the human action," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 247–254, 2022.

[11] C. Jimenez, E. Saavedra, G. del Campo, and A. Santamaria, "Alexa-based voice assistant for smart home applications," *IEEE Potentials*, vol. 40, no. 4, pp. 31–38, 2021.

[12] S. A. Kumer, P. Kanakaraja, A. P. Teja, T. H. Sree, and T. Tejaswni, "Smart home automation using ifttt and google assistant," *Materials Today: Proceedings*, vol. 46, pp. 4070–4076, 2021, international Conference on Materials, Manufacturing and Mechanical Engineering for Sustainable Developments-2020 (ICMSD 2020). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214785321017570

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.