Smart Scheduling of Home Appliances Using Happiness Aware Scalable Reinforcement Learning Agent

Debajyoti Dasgupta[1](✉), Arijit Mondal[2], and Partha P. Chakrabarti[1]

[1] Indian Institute of Technology Kharagpur, Kharagpur, India
debajyotidasgupta6@gmail.com, ppchak@cse.iitkgp.ac.in
[2] Indian Institute of Technology Patna, Patna, India
arijit@iitp.ac.in

Abstract. This study introduces IMPEARL (Incremental Model-based Predictor Network Agent with Reinforcement Learning), a novel deep reinforcement learning approach for optimizing household appliance scheduling. This method intelligently considers uncertainties, variable pricing models, and user satisfaction, efficiently scaling with increased tasks. The model uniquely represents appliance states and job statuses, employing a Deep Q-network with incremental training and feature-extractor networks for improved performance. By incorporating a recurrent connection, IMPEARL captures the time series aspect of the problem, leading to significant bill savings. Compared to traditional first-come, first-served methods and state-of-the-art models, IMPEARL achieves a 37.5% reduction in billing costs, a 66.5% increase in user satisfaction, a 29.08% improvement in minimizing billing costs, a 7.06% decrease in user dissatisfaction, and is 6.75% less sensitive to variability, demonstrating its effectiveness in dynamic, non-smart environments.

Keywords: Smart scheduling · Home appliances · Happiness aware · Reinforcement learning · Deep Q-network · Incremental training · Time-based fine-tuning · IMPEARL model · Predictor network · Sensitivity

1 Introduction

Increasing household power consumption demands advanced infrastructure despite the high costs. Peaks in energy usage occur during mornings and evenings, highlighting the need for efficiency improvements by shifting appliance use to off-peak hours, a key demand-side management strategy. Demand-side management involves direct load control with smart appliances and variable pricing. Smart grids, supported by municipal programs like PACE financing, are crucial, especially in renewable energy integration and electric vehicle charging. In developing countries with limited smart grids, our model uses internet-connected devices for flexible scheduling based on daily needs.

We employ a Deep Reinforcement Learning (DRL) approach to develop a job scheduling model that minimizes costs with time-of-day pricing and penalties for over-utilization. Building on previous work, we introduce an efficient image-based scheme to represent device states and energy usage, using feature extractors like Inception Net, ResNet50, and a modified ResNet. LSTM and LSTNet address time series aspects, while three novel training methods-incremental training, time-based fine-tuning, and predictor-guided training-along with a user happiness score metric, enhance performance. Incremental training increased efficiency by **22%** for less efficient models and **10%** for more efficient ones, while the predictor network setup boosted top architecture efficiency by **6.5%**. Our methods cover **60%** more potential schedules daily than existing methods. The resulting model, IMPEARL, achieves a **37%** cost reduction compared to manual scheduling and boosts user satisfaction by **66.5%**. The paper is organized as follows: related works (Sect. 2), system model (Sect. 3), IMPEARL model formulation (Sect. 4), experimental results (Sect. 5), and conclusion (Sect. 6).

2 Related Work

Demand Side Management (DSM) involves optimizing energy use without additional infrastructure, focusing on demand response, conservation, and load management [18,25]. Residential load control, a key DSM aspect, aims to reduce and shift consumption, with Direct Load Control (DLC) allowing remote management of home appliances [1,3,17]. However, user comfort remains a critical barrier [21]. Users independently shift heavy loads to off-peak hours to save on electricity bills, reducing the Peak-to-Average Ratio (PAR) and costs [14,16,19]. Deterministic algorithms face challenges modeling constraints for increased uncertainty and complexity [20], demanding AI-based solutions.

2.1 Deterministic Approaches

Methods in [4] model cost functions and appliance constraints using greedy algorithms for scheduling, influenced by electricity prices and incorporating penalties for significant changes. While simulations show benefits like lower costs and reduced peak loads, user satisfaction and manual intervention uncertainties are often overlooked [16].

2.2 Meta-heuristics and AI-ML Based Approaches

Genetic Algorithms (GA) have been applied for cost minimization, integrating battery storage and renewable energy sources (RESs) [16,19]. These systems charge batteries during low-cost periods for later use, prioritizing high-demand appliances from battery sources during peak prices. Evolutionary algorithms and Deep Reinforcement Learning (DRL) have also been employed for load optimization and energy storage management [10,13]. DeepRM [15] demonstrates RL's effectiveness for multi-resource scheduling, inspiring solutions in this paper.

Current models often overlook electricity fluctuations' impact on losses and neglect end-user satisfaction. While studies like [12] introduce satisfaction, they fail to account for human uncertainty and struggle to optimize both cost and satisfaction. This paper addresses these gaps by focusing on user satisfaction and manual intervention delays, where deterministic algorithms typically fall short.

3 System Model

The following terminologies are used in the rest of the paper.

Jobs are atomic, non-preemptive tasks. A job is defined as $j_i = (\delta_i, w_i)$, where δ_i is the time required for the i^{th} job and w_i is the average power consumed per unit time.

A **process** is a task that the end user needs to schedule, consisting of one or multiple jobs. For instance, *washing clothes* includes jobs like spinning and rinsing. Processes can be preemptive or non-preemptive. Each process is characterized by a list of pending jobs, $h_i = \{j_{i_1}, j_{i_2}, ..., j_{i_k}\}$.

Machine refers to any household device used to execute a process, such as a washing machine or air conditioner. Machines are categorized into three types:

- **Type 1:** Appliances with definable stages, such as washing machines. These tasks can be preempted and resumed.
- **Type 2:** Periodic appliances like AC, maintaining a target condition and restarting as needed. For example, an AC scheduled at $25°C$ for t_1 to t_2 hours operates in intervals, cooling for x mins and wearing off for x mins.
- **Type 3:** Appliances are performing repeated tasks with variable settings, such as microwaves (short bursts of heating food).

Peak Power (at time t) is the total power consumed by all devices. Time-of-the-day pricing with penalties for crossing a base limit is used.

$$\text{Total Cost}_t = C_t \times \sum_{m=1}^{|M|} P_{t,m} + \lambda_t \times \max\left(0, \sum_{m=1}^{|M|} P_{t,m} - P_{thresh,t}\right) \quad (1)$$

where M is the total number of machines, the focus is minimizing the total bill amount and peak power consumption by spreading consumption throughout the day. User satisfaction is maximized by including in the reward function.

4 Overview of IMPEARL Methodology

4.1 Overall Architecture

Figure 1 shows the high-level flow of information for the RL Agent-based scheduler, which gathers system information while scheduling jobs, monitors for delays, and adjusts future schedules. This figure also illustrates the training and inferencing cycle in a real-world scenario.

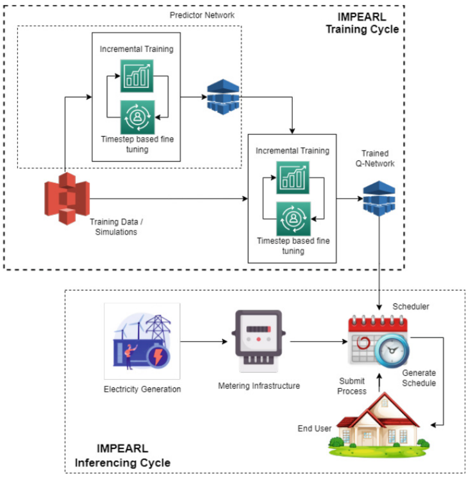

Fig. 1. Overview of the IMPEARL Scheduler

The **IMPEARL** architecture (Fig. 1) uses an untrained Q-network trained with incremental training and time-based fine-tuning methodologies. The best results were observed with $\alpha = 15$ and $\beta = 5$ mins. The training process guides another model with equal or better representational power. Feedback from the previously trained model improves representation. Training accounts for human intervention uncertainty and user satisfaction.

4.2 Deep Q Learning

Image Representation: The **observation state** is depicted as an image with dimensions **(number of machines** $\times$ **number of jobs** $\times$ **number of distinct types of machines)** (Fig. 2). Each image layer focuses on scheduling jobs for a specific type of machine, reducing redundant states. The intermediate state shows power utilization levels, with darker cells indicating lower power usage.

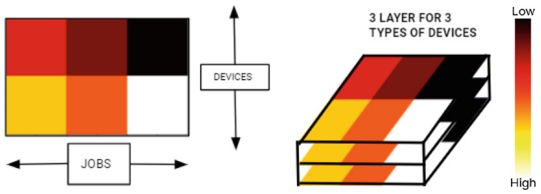

Fig. 2. (a) **Energy usage** for one device type, (b) 3D stacked 2D layers for three devices at a unit time, (c) Power usage scale (Dark [low] to White [high])

Action Space: The **action space** is represented as a bitmap, where each action schedules a job on the i^{th} device (1) or not (0). The action space range is defined by $Action_{min} = 0$ and $Action_{max} = 2^M \times 2^D$, depending on the number of machines (M) and types of machines (D, which is 3).

Rewards: **Rewards** shape the RL agent's decision-making. The custom reward function considers penalties for illegal scheduling or deadline breaches, rewards for completing jobs, and penalties inversely proportional to pending jobs. Rewards also include incentives for minimizing power consumption fluctuations and a happiness score for overall satisfaction.

Happiness: The happiness function quantifies user satisfaction based on job requirements, urgency, and completion time variation. It uses a mathematical formulation inspired by [5,6,22,26]:

$$\text{Happiness} = p \cdot (\frac{g^{1.3} - 75}{80} - 17) \cdot l_1 + (k_1 \cdot \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{g}{\sigma})^2}) \cdot l_2 \tag{2}$$

$$+ (\frac{(g-60)^{1.7}}{120} - k_2) \cdot l_3 \tag{3}$$

$$k_1 = 3.5 \times 10^4, k_2 = 10.5 \tag{4}$$

$$l_1 = \begin{cases} 1 & t \leq d - 125 \\ 0 & t \geq d - 125 \end{cases}; l_3 = \begin{cases} 0 & t \leq d + 125 \\ -1 & t \geq d + 125 \end{cases}; l_2 = \begin{cases} 0 & t \leq d - 125 \\ 1 & d - 125 \leq t \leq d + 125 \\ 0 & t \geq d + 125 \end{cases} \tag{5}$$

Task completion time is t, deadline is d, and the difference is $g = |t - d|$. The standard deviation σ is 35 min. Happiness penalizes late completion, promotes scheduling near the deadline, and varies rewards for early completion based on the device type. The function parameters are derived using a grid search. The **dis-satisfaction score** is defined as:

$$\text{D-Score} = 1 - Normalized\ Happiness$$

4.3 Neural Architectures to Realize IMPEARL

We use deep reinforcement learning with image-based representations and feature extractor networks like InceptionNet-V3 [24], ResNet-50(V2) [7], and smaller residual networks. We add a bidirectional LSTM layer and 1D Convolution for comparison. The predictor network uses LSTNet [11] for multi-variable time-series regression (Fig. 3).

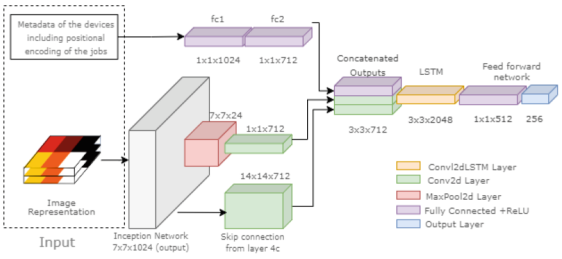

Fig. 3. Architecture of Neural Network for Deep Q-Network

4.4 Novel Training Methodologies

Incremental Training: Incremental training begins with a single hidden layer focusing on one machine. As more machines are included, subsequent layers are trained. This approach enhances learning efficiency. Initial runs avoid random initialization. Training allows previous layers to be trainable for a limited fraction of steps. For example, the 3rd layer is trained for 3000 steps, with 2500 focusing on the 3rd layer and 500 on the first two layers. The final phase involves fine-tuning the last two layers and the output layer over 2000-5000 steps.

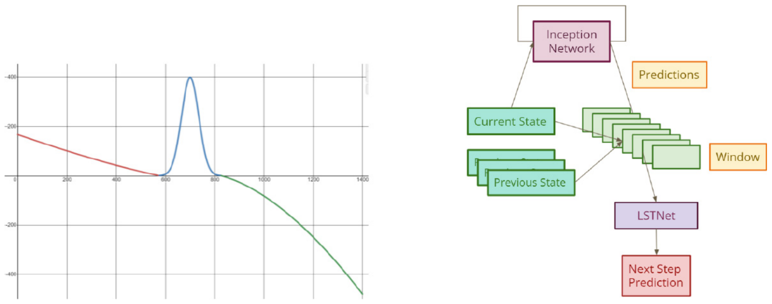

Fig. 4. (a) Happiness function depicted as a continuous graph (b) Predictor network process flow diagram (INet-V3 as predictor)

Time Based Fine Tuning: The model divides the day into α minute intervals for initial training and fine-tunes using shorter β minute intervals. This method improves performance, with $\alpha = 15$ and $\beta = 5$ mins yielding the best results.

Predictor Network: The predictor network addresses limited exploration in DQN training by using LSTNet [11]. It combines 6 historical and 3 predicted future steps. The Inception Network-based RL agent serves as a feature extractor, guiding the LSTNet-based RL agent (Fig. 4(b)).

Sensitivity Analysis Methodology: Sensitivity analysis measures the impact of random perturbations on schedules. Let S be the schedule, and S_δ the perturbed schedule with a random shift $\delta \in \{-10, -5, 0, 5, 10\}$ mins. The sensitivity metric is defined as:

$$Sensitivity(S) = \sqrt{\frac{1}{|\Delta|} \sum_{\delta \in \Delta} (\text{Cost}(S_\delta) - \mu(S))^2} \tag{6}$$

where $\Delta = \{-10, -5, 0, 5, 10\}$, and S_δ is the perturbed schedule. Lower sensitivity indicates more robust schedules to random delays or early scheduling.

5 Results and Discussions

5.1 Experimental Setup

Environment: Our model consists of devices from various groups (3) that exhibit distinct states. For instance, washing jobs indicate their current process stage and remaining time, while air conditioners show the current temperature. The environment tracks job completion percentages and is implemented using two classes: Jobs and Machine. It maintains a list of pending jobs and descriptions, with each machine linked to a list representing process stages. To simulate real-world unpredictability, occasional delays of 5-10 min are introduced. The RL agent is informed of each job stage's time and power consumption, which are arranged sequentially. The input includes a time rate chart, with the output being a schedule detailing job cycles, timings, and assigned machines.

Table 1. Time of the Day Rating Chart for experiments (1 Rs (Indian Rupees) = \$0.012 (US\$) as per current rates)

Time upto (mins)	Cost (Rs/min)	limit (W)	penalty (Rs/min)
0-240	10	110	450
240-480	30	110	290
480-720	80	200	50
720-960	45	170	180
960-1200	20	140	380
1200-1440	5	230	550

Devices, Jobs, and Data: The number of device types is 3, categorized into 3 groups. We refer to [9] for a real-world time-of-the-day pricing scheme, used to build a small synthetic time-of-the-day pricing shown in Table 1.

5.2 Algorithmic (*Heuristic*) Bounds

Using N for machines, J for jobs, and T for total time units in a day, we compare performance against two primary heuristics: First Come First Served (FCFS) and Binary Search-based lower bound, as well as a random policy.

The **FCFS method** allocates jobs to the first available machine, mirroring typical human behavior, but often leading to inefficiencies due to job accumulation in the earliest available slots.

The **Binary Search** - based lower bound offers an idealistic minimum bill amount, assuming no delay or uncertainty. Minimizing it involves performing a binary search on the bill amount per time unit, iterating possible bill limits to find the lowest achievable under ideal conditions.

The **MILP baseline** models the scheduling problem as a mixed-integer linear program, leveraging optimization techniques for near-optimal solutions. It formulates the problem with linear constraints and an objective function to minimize, capturing aspects such as job deadlines, resource capacities, and cost considerations.

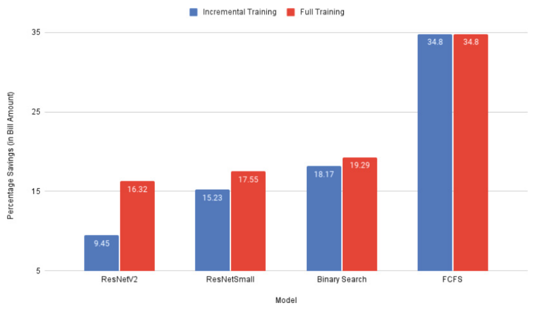

Fig. 5. Percentage Saving (in Bill Amount) of Incrementally trained InceptionNet w.r.t other models

5.3 Comparison Between Neural Network Architectures

Figure 5 shows that the Inception Network demonstrates a significant 37.5% reduction in total bill amount compared to FCFS, indicative of its efficiency over standard greedy human behavior. Figure 6 reveals that the Inception Network outperforms other models during incremental training, rapidly adapting to optimal scheduling. While simple residual networks initially show promise, ResNet-50 eventually shows better performance. The Inception Network consistently leads in performance as complexity escalates, outmatching simpler models.

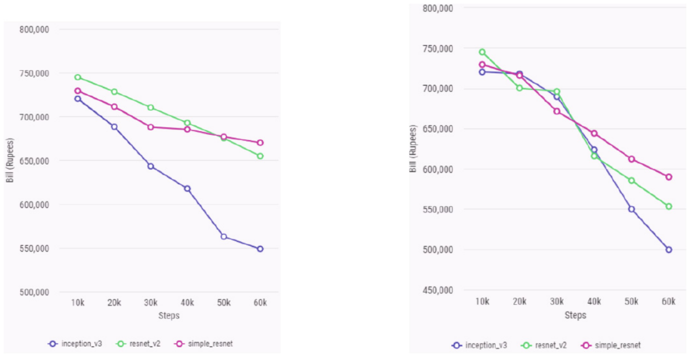

Fig. 6. Bill amount achieved by InceptionNet per training step (in evaluation phase) (a) Full Training (b) Incremental Training

5.4 Ablation Study of Jobs and Devices Using InceptionNet

Scalability tests on the InceptionNet Agent (Fig. 7, top-left) reveal that maintaining a fixed ratio of devices to jobs results in stable pricing. Figure 7 (top-right) shows that pricing escalates with constant devices and increasing jobs due to higher electricity consumption and scheduling challenges. Conversely, Fig. 7 (bottom) shows that increasing devices while keeping jobs constant reduces costs due to improved scheduling and balanced peak power consumption.

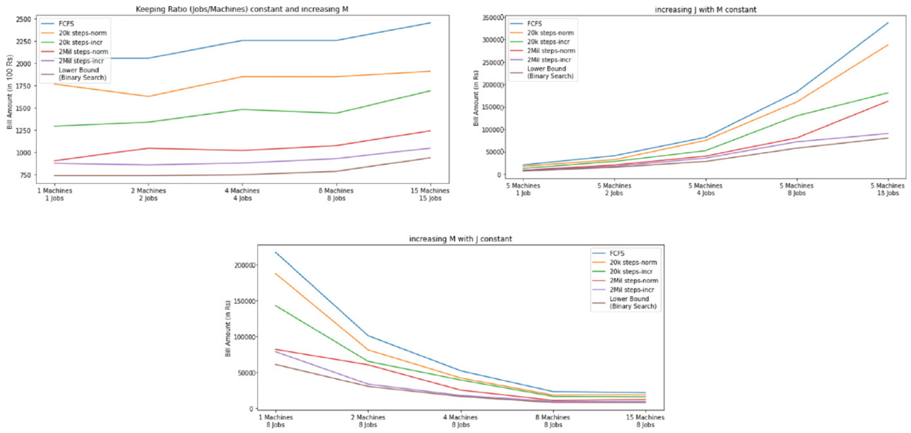

Fig. 7. Minimum bill achieved by the InceptionNet-based agent keeping (top-left) the ratio of number of devices to jobs fixed (top-right) number of devices fixed and increasing jobs (bottom) number of jobs fixed and increasing devices

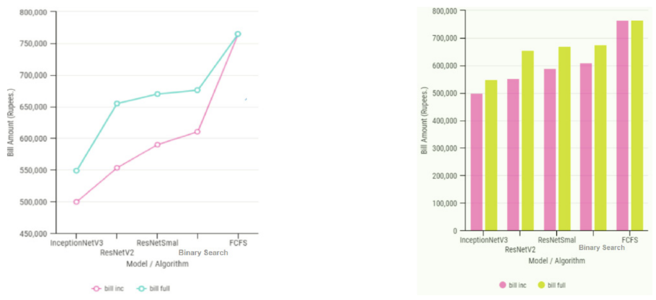

Fig. 8. Comparison of the minimum bill of full versus incremental training (full vs inc) using Inception Network based Agent - (a) Line plot (b) Bar plot

5.5 Effect of Incremental Training

Figure 8 shows that models trained using fine-tuning perform better than those trained using the full training approach. The Inception network shows less difference due to its efficiency, while Resnet shows significant improvement with incremental training, demonstrating its efficiency in training less efficient networks. Incremental binary search algorithms perform rearrangements in some schedule subsegments.

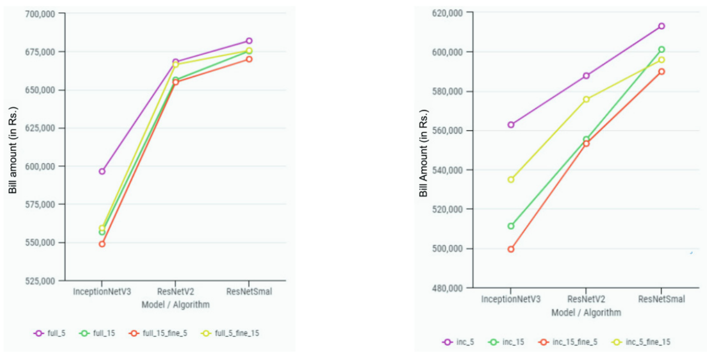

Fig. 9. Comparison of performance of models under (a) Full (b) Incremental Training for different combinations of tuning

5.6 Study of Fine-Tuning

Figure 9 shows that the combination of 15-minute coarse and 5-minute fine training consistently yields the best results. Adding a recurrent layer (LSTM) in conjunction with the feature extractor layer shows mixed results. The InceptionNet benefits marginally from the LSTM layer, achieving similar results to its non-LSTM counterpart, while Resnet v2 shows significant improvement with the recurrent layer, capturing time-related aspects effectively (Fig. 10).

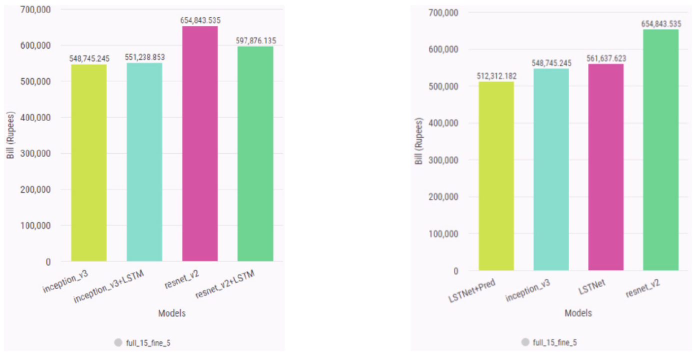

Fig. 10. Billing Cost of different NN (a) with and without recurrent layer (b) compared with predictor network (IMPEARL)

5.7 Predictor Network

The predictor network uses the optimized inception network as the predictor model with the LSTNet model. LSTNet is used for multivariate time series forecasting. The predictor network guides the LSTNet during training, significantly improving its performance compared to the stand-alone InceptionNet (Table 2).

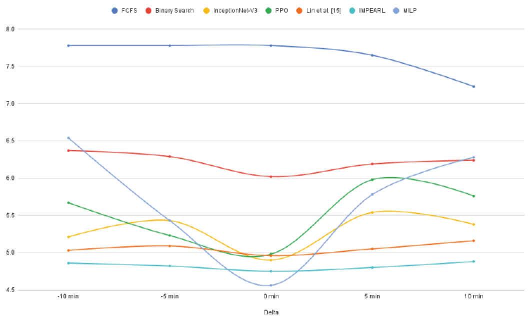

Fig. 11. Average value of the bill amount achieved by 5 rounds of random perturbance to the base schedule for each value of δ (Rs. 100k)

5.8 Sensitivity Analysis

Figures 11 and 12 show that the IMPEARL model outperforms both MILP and [12] in mean bill amount and sensitivity. MILP achieves the lowest mean bill amount of Rs. 456k but exhibits the highest sensitivity of 0.78, indicating high susceptibility to task timing perturbations. IMPEARL balances cost optimization and robustness, with a mean bill amount of Rs. 482k and a sensitivity

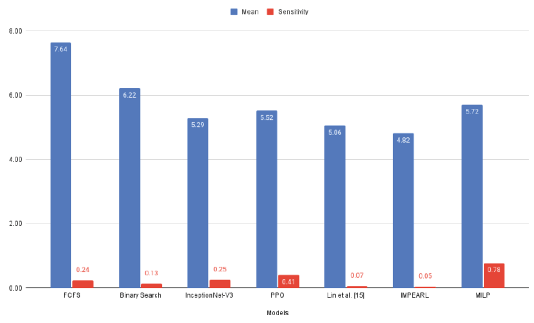

Fig. 12. Mean and sensitivity of the schedules generated by the models after 5 rounds of perturbance for each of the 5 values of δ (Rs. 100k)

of 0.05, outperforming [12] (mean bill amount Rs. 506k, sensitivity 0.07). This suggests that IMPEARL generates more robust schedules.

Table 2 reveals that LSTNet consistently outperforms InceptionNet and ResNet. The optimal training methodology involves coarse training with 15-minute intervals and subsequent fine-tuning with 5-minute intervals. Incremental training methods produce better outcomes than full training. The most effective model combines an LSTNet with a Predictor Network (optimized InceptionNet), using the $Inc_{15}\ fine_{5}$ training.

Table 2. Results summary table (Bill Amount in Rs.)

Training	Inception Net-V3	ResNet50 +LSTM	IMPEARL
$Full_5$	589,613	636,928	543,807
$Full_{15}$	557,280	618,263	519,582
$Full_5\ fine_{15}$	565,193	625,112	526,764
$Full_{15}\ fine_5$	548,745	597,876	512,312
Inc_5	564,128	590,532	517,883
Inc_{15}	511,321	550,624	485,762
$Inc_5\ fine_{15}$	547,299	566,789	502,875
$Inc_{15}\ fine_5$	499,464	537,725	**479,225**

5.9 Summary Table and Recommended Method

Table 3 consolidates these findings, comparing the best-performing models against established baselines. This comparison, focused on the 15-minute coarse and 5-minute fine training settings, demonstrates significant cost reductions and improved end-user satisfaction.

Table 3. Perfomance comparison

Model	Bill (Rs. 10^5)	Sensitivity (Rs. 10^3)	D-Score
FCFS	7.68	1.00	1.000
Binary Search	6.16	0.80	0.101
InceptionNet-V3	4.99	3.84	0.297
TD3 [23]	5.12	7.36	0.453
PPO [2]	5.19	5.67	0.372
DDPG [8]	5.35	10.12	0.591
[12]	5.03	7.98	0.085
MILP	4.82	6.54	0.335
IMPEARL	**4.79**	**1.25**	**0.079**
Empirical Bound	4.01	0.53	0.054

The **Empirical Bound** represents the statistical lower bound for 95% of the observations, calculated as the mean (μ) minus two times the standard deviation (σ). This bound provides a reference point for expected optimal performance.

5.10 Ablation on Happiness Function

The happiness function is a crucial component of the IMPEARL model. It quantifies user satisfaction based on job completion time relative to its deadline, early or late completion impact, and specific device requirements.

The device-specific parameter p allows for customization based on device characteristics. The happiness function captures preferences for timely completion, tolerance for slight deviations, and device-specific requirements. A grid search approach determines the optimal parameter values. The ablation study assesses the model's sensitivity to these parameters, revealing the robustness and effectiveness of the proposed happiness function (Fig. 13).

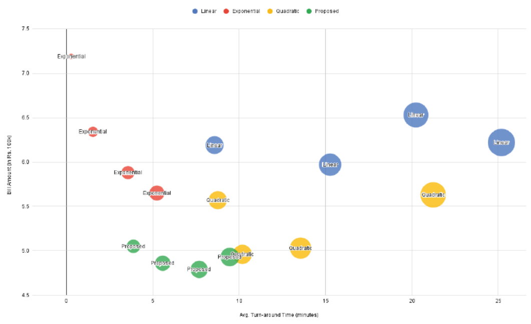

Fig. 13. Comparison of different happiness functions centered at the job's deadline. The proposed happiness model captures the complex dynamics of user satisfaction across various scenarios

6 Conclusions and Future Work

This paper introduces an innovative approach to representing power consumption profiles as images, which are then effectively transformed into multivariate time series representations. We demonstrate that models integrating mathematical concepts, like ResNet and LSTNet, exhibit superior performance. The efficacy of incremental training, especially in scenarios where learning from simpler versions of a problem aids in solving more complex ones, is highlighted. This approach, which involves faster initial learning steps followed by finer, detailed fine-tuning, enhances the learning process. Additionally, leveraging knowledge from well-trained models (like Inception Network) proves beneficial in guiding newer models (like LSTNet) toward more efficient solutions by avoiding unproductive learning states. Future work could explore integrating renewable energy sources to improve happiness scores, implementing similar scheduling techniques on the generation side to minimize energy losses, and extending the model to a multi-agent framework for broader scheduling applications.

References

1. Ahmad, A., et al.: An optimized home energy management system with integrated renewable energy and storage resources. Energies **10**(4) (2017)
2. Alec, R.: Proximal policy optimization algorithms (2017)
3. Azar, A.G., Jacobsen, R.H.: Appliance scheduling optimization for demand response. Int. J. Adv. Intell. Syst. **9**, 50–64 (2016)
4. Chavali, P., Yang, P., Nehorai, A.: A distributed algorithm of appliance scheduling for home energy management system. IEEE Trans. Smart Grid **5**(1), 282–290 (2014). https://doi.org/10.1109/TSG.2013.2291003
5. Dhillon, J.S., Kothari, D.P.: Power System Optimization. Prentice-Hall of India (2000)
6. Guo, Y., Li, K., Laverty, D.M., Xue, Y., Fei, M.: A multi-objective optimization approach to balancing cost and customer satisfaction in demand response. In: IEEE Power & Energy Society General Meeting, pp. 1–5. IEEE (2015)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90
8. Hunt, J.J.: Continuous control with deep reinforcement learning (2019)
9. K Raheja Group, P.: Know your bills (2018). https://www.krahejacorppower.com/know-your-bill
10. Kumar, H., Mammen, P.M., Ramamritham, K.: Explainable AI: deep reinforcement learning agents for residential demand side cost savings in smart grids. CoRR arXiv:1910.08719 (2019)
11. Lai, G., Chang, W., Yang, Y., Liu, H.: Modeling long- and short-term temporal patterns with deep neural networks. CoRR (2017)
12. Li, S., Cao, D., Huang, Q., Zhang, Z., Chen, Z., Blaabjerg, F., Hu, W.: A deep reinforcement learning-based approach for the residential appliances scheduling. Energy Rep. **8**, 1034–1042 (2022). https://doi.org/10.1016/j.egyr.2022.02.181, https://www.sciencedirect.com/science/article/pii/S2352484722004279, iCPE 2021 - The 2nd International Conference on Power Engineering iCPE 2021 - The 2nd International Conference on Power Engineering

13. Logenthiran, T., Srinivasan, D., Shun, T.Z.: Demand side management in smart grid using heuristic optimization. IEEE Trans. Smart Grid **3**(3), 1244–1252 (2012). https://doi.org/10.1109/TSG.2012.2195686
14. Ma, K., Yao, T., Yang, J., Guan, X.: Residential power scheduling for demand response in smart grid. Int. J. Electric. Power Energy Syst. **78**, 320–325 (2016). https://doi.org/10.1016/j.ijepes.2015.11.099
15. Mao, H., Alizadeh, M., Menache, I., Kandula, S.: Resource management with deep reinforcement learning. In: HotNets '16, Proceedings of the 15th ACM Workshop on Hot Topics in Networks, pp. 50–56. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/3005745.3005750
16. Marzband, M., Yousefnejad, E., Sumper, A., Domínguez-García, J.L.: Real time experimental implementation of optimum energy management system in standalone microgrid by using multi-layer ant colony optimization. Int. J. Electric. Power Energy Syst. **75**, 265–274 (2016). https://doi.org/10.1016/j.ijepes.2015.09.010
17. Mesarić, P., Krajcar, S.: Home demand side management integrated with electric vehicles and renewable energy sources. Energy Build. **108**, 1–9 (2015)
18. Mohsenian-Rad, A.H., Wong, V.W.S., Jatskevich, J., Schober, R., Leon-Garcia, A.: Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid. IEEE Trans. Smart Grid **1**(3), 320–331 (2010)
19. Nadeem, Z., Javaid, N., Malik, A.W., Iqbal, S.: Scheduling appliances with GA, TLBO, FA, OSR and their hybrids using chance constrained optimization for smart homes. Energies **11**(4) (2018)
20. Paliwal, N.K.: A day-ahead optimal scheduling operation of battery energy storage with constraints in hybrid power system. Procedia Comput. Sci. **167**, 2140–2152 (2020). https://doi.org/10.1016/j.procs.2020.03.263, international Conference on Computational Intelligence and Data Science
21. Rasheed, M.B., et al.: Priority and delay constrained demand side management in real-time price environment with renewable energy source. Int. J. Energy Res. **40**(14), 2002–2021 (2016). https://doi.org/10.1002/er.3588
22. Schwartz, B., Ward, A., Monterosso, J., Lyubomirsky, S., White, K., Lehman, D.R.: Maximizing versus satisficing: happiness is a matter of choice. J. Pers. Soc. Psychol. **83**(5), 1178–1197 (2002)
23. Sutton, R.S.: Learning to predict by the methods of temporal differences. Mach. Learn. **3**(1), 9–44 (1988). https://doi.org/10.1007/BF00115009
24. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016). https://doi.org/10.1109/CVPR.2016.308
25. Wen, Z., O'Neill, D., Maei, H.R.: Optimal demand response using device based reinforcement learning. CoRR arXiv:1401.1549 (2014)
26. Zhang, Y., Yao, F., Iu, H., Fernando, T., Trinh, H.M.: Optimal operation of building energy systems incorporating customer satisfaction. IEEE Trans. Smart Grid **5**(4), 1802–1813 (2014)