COMPUTING AND APPLICATIONS REVIEWS

Volume 3, Number 1, 2026 Pages 18 – 35 WEBSITE. ca-reviews.com

ENERGY-EFFICIENT MOBILE OPERATING SYSTEMS THROUGH INTELLIGENT BACKGROUND PROCESS MANAGEMENT

Abdullah Nadeem[1]*, Eman Zahra[2]

[1] *Department of Software Engineering, Institute of Mobile Computing and Embedded Systems, Lahore, Pakistan*
[2] *Department of Computer Science, Center for Intelligent Operating Systems Research, Islamabad, Pakistan*
**Corresponding Author Email: abdullah.nadeem@gmail.com*

Article Information

Article History

Received:	January 03, 2026
Revised:	February 08, 2026
Accepted:	April 11, 2026
Available Online:	June 30, 2026

Keywords:
Mobile Operating Systems, Energy Efficiency, Background Process Management, Battery Optimization, Intelligent Scheduling

Abstract

Mobile operating systems continuously manage multiple background processes such as synchronization services, notifications, location tracking, application updates, and network communication. Although these processes improve user experience, they also consume significant battery power, memory, and CPU resources when not properly controlled. This paper examines how intelligent background process management can improve energy efficiency in mobile operating systems. The proposed approach focuses on monitoring background application behavior, identifying unnecessary resource usage, and applying adaptive scheduling, process limitation, and priority-based execution. By using intelligent decision-making mechanisms, the system can reduce excessive wake-ups, limit non-critical background tasks, and allocate resources according to application importance and user activity patterns. The study highlights that intelligent background management can reduce power consumption, improve battery lifetime, and maintain acceptable application responsiveness. The results suggest that adaptive control of background processes is an effective strategy for enhancing mobile device performance without negatively affecting user experience. This research contributes to the development of smarter mobile operating systems that balance energy saving, usability, and system stability.

INTRODUCTION

The applications that modern mobile devices rely on are ever more power intensive, using a variety of wireless interfaces and sensors, consuming battery life in a matter of hours (Vallina-Rodríguez & Crowcroft, 2012). This power consumption problem is compounded by sub-optimal background applications that often make resource hungry tasks, thus requiring more advanced architectural solutions (Martins et al., 2015; Chen et al., 2015). In the strict requirement of battery technology, the current mobile operating systems employ simple background management mechanisms to enhance user responsiveness of applications by preloading them, but this is not sufficient to meet user performance expectations (Machmudi et al., 2025; Xia et al., 2013). Some background applications are unavoidable, but can be monitored and controlled, others cannot, such as frequent network polling, unnecessary sensor access or misused and poorly-designed wake locks, that lead to high energy consumption that is not obvious to the end-user (Martins et al., 2015; Shen, 2013). Empirical studies have revealed that there are background actions that are critical (e.g., notifications) and background actions that are non-critical (e.g., visualizing a new application) and traditional and heuristic policies do not distinguish between the two (Chen et al., 2015; Hort et al., 2021; Martins et al., 2015). This results in a significant difference in the fast evolution of mobile apps with high features and the relatively low capacity growth of lithium-ion batteries, which has compelled the research to shift its focus towards intelligent and adaptive resource management (Donohoo et al., 2007; Perrucci et al., 2011; Pothineni, 2025; Vallina-Rodríguez & Crowcroft, 2012). As the OS evolved towards a more modern approach, machine learning became more popular, with the aim of learning more about the user's behaviour and usage patterns so that the OS could make more predictions according to the context in which a background task is run (Donohoo et al., 2007; Machmudi et al., 2025; Mandal et al., 2019; Nawrocki & Śnieżyński, 2020). The dynamic management of resources across the system, such as CPU frequency scaling, dynamic core affinity, and using smart network interfaces, can be effectively managed through techniques like reinforcement learning and imitation learning, which can enable significant energy savings without compromising the quality of experience (Machmudi et al., 2025; Mandal et al., 2019; Sang et al., 2026). However, current state-of-the-art systems still have many issues, including high computation cost for on-device ML inference, complexity of multi-domain resource governing, and the

need for personalized policies that are generalizable across heterogeneous hardware platforms (Machmudi et al., 2025; Sang et al., 2026). This overall research objective aims to address these crucial gaps and to suggest and test an innovative and intelligent background process management system with the aim of optimizing the energy usage while maintaining the responsiveness of the system. The purpose of this work is to propose a more sustainable solution for the allocation of the resources needed to sustain the operational life of modern mobile devices with limited resources and provide advanced predictive modeling and limited monitoring tools on the device (Machmudi et al., 2025; Pothineni, 2025; Sang et al., 2026). This study focuses on the effectiveness of multi-objective reinforcement learning (MORL) algorithms that consider both energy use and responsiveness of the user experience (Wang et al., 2025).

LITERATURE REVIEW

Background process management is no longer being studied in a heuristic manner, but rather with more sophisticated predictive methods that strive to reconcile the demands of context-aware, always-on AI services with battery life (Fadzil & Chan, 2026).Traditional schedulers that use basic heuristics, such as the "off screen" trigger, or the "time to next poll" trigger are very simple and do not account for the complex and intermittent nature of modern mobile application workloads (Chen et al., 2015; Hort et al., 2021). One example of such early mechanisms is ensuring that background applications are kept in memory, to improve user experience, often with the downside of a corresponding background energy use and memory pressure caused by improper task management (Xia et al., 2013). More recent efforts have included additional refinements such as TAMER for layering task wake-ups for filtered rate-limit, or HUSH for screening out non-essential tasks in the screen-off state (Chen et al., 2015; Martins et al., 2015). Despite all the aforementioned, the idea of continuous services supported by AI has sparked a shift towards more flexible and contextually aware approaches (Fadzil & Chan, 2026; Nawrocki & Śnieżyński, 2020). Recently, deep reinforcement learning and imitation learning have been used in various methodologies for multi-domain resource governing, e.g., Trident (Machmudi et al., 2025), schedulers based on PPO (Mandal et al., 2019; Sang et al., 2026), which dynamically adjust CPU frequency, core affinity, and interface utilization. While these methods show promising results in optimizing the energy consumption and maintaining a responsive system, they also pose challenges such as the computational complexity of on-device inference and the requirement for customization to ensure

policies are applicable across diverse hardware devices (Machmudi et al., 2025; Sang et al., 2026; Wang et al., 2025). The lack of coupling between scheduling and governing, and the processes operate separate from each other may result in conflicts without intention, thus to an sub-optimal power consumption and the QOS. Moreover, the current non-tight coupling of task scheduling and frequency governing makes it hard for the system to achieve the greatest energy efficiency in dynamic workload switches (Sang et al., 2024). Moreover, existing systems tend to either over-provision or under-provision processing resources due to using empirical rules that are not able to accurately predict sudden increase in user access (Soultxoglou et al., 2024). In addition, research shows that traditional DVFS governors are not effective for periodic soft real-time tasks due to their difficulties in fine-grained frequency support and load imbalance. Moreover, the legacy governors are rigid and reactive, whereas the resource demands of today's highly dynamic and heterogeneous background applications (Shankar et al., 2025; Zhou & Lin, 2023) continuously evolve and change. While conventional schedulers are primarily concerned with the state(s) of processes, these schedulers do not consider the intricate thermal and battery-state needs which require more extensive and heterogenetically aware optimization (Ahmed et al., 2025; Vangaru, 2025). Further, heuristic models generally only rely on locally optimal solutions, that might not bring any benefit in terms of global efficiency and hence leave the performance trade-offs not optimized in the presence of non-linear multi-objective constraints (Liu et al., 2026; Zhang et al., 2024). Moreover, the reinforcement learning (RL) based methods developed until now are not widely generalizable, and they typically necessitate lengthy retraining of the model for the different performance characteristics of various heterogeneous mobile SoC (Yan et al., 2025).

METHODOLOGY

In this work, a framework of intelligent scheduling of background processes based on multi-objective reinforcement learning (MO-RL) to jointly optimize task scheduling, frequency scaling and power allocation is proposed (Samadi et al., 2025).The framework proposes the hierarchical control design method which decouples the global task assignment and local resource adaptation methods, thus better optimising the trade-offs between battery health and thermal accumulation and application responsiveness. This architecture has two layers of reinforcement learning: A global manager periodically evaluates the performance of the system (including demand of applications and thermal

constraints) and then determines high level resource policies, while a collection of local governors determines the CPU frequency, affinity for cores and CPU scheduling of individual tasks at runtime (Calabrese et al., 2018; Stylos et al., 2025). The system is designed as a Markov Decision Process to support the autonomous policy optimization (Liu et al., 2026; Machmudi et al., 2025). The state space S is constructed from real-time telemetry data like per-core CPU utilization, thread-specific workload patterns, GPU activity, thermal sensor data, user interaction signals (via lightweight kernel hooks) and battery state-of-charge (Ahmed et al., 2025; Huang et al., 2017; Vangaru, 2025). Fine-grained interventions such as dynamic voltage and frequency scaling, task migration between heterogeneous cores (big LITTLE) are under the umbrella of the action space A.Modification of the priority of threads in cgroup limits (Machmudi et al., 2025; Samadi et al., 2025; Sang et al., 2026) and scheduling architecture (LITTLE architectures). We have designed a weighted multi-objective reward function R to optimize the energy saving and QOE, where the energy saving is more weighted in the foreground process, and the resource usage in the background process is less weighted (Sang et al., 2024; Wang et al., 2025). The data the system is trained on and the data it uses to adapt on the device is generated by system telemetry data which is processed to extract features that can describe the dynamics in the current workload and predict future thermal throttling events (Vangaru, 2025). The evaluation method is designed to be a system service deployed on commercial mobile SoCs and compared with the effectiveness of traditional Linux built-in governors and baseline RL-based schedulers (Dou et al., 2024; Sang et al., 2026). The quantitative metrics used to test are: total power consumption (milliwatt hours), UI responsiveness in terms of frame-drop rate and touch-to-render latency, thermal stability metrics, and task completion time for different workloads running in the background including data-intensive tasks and compute-heavy tasks (Dou et al., 2024; Liu et al., 2026; Machmudi et al., 2025; Wang et al., 2025). We believe this hierarchical framework will decrease the complexity required to implement on-device ML inference, achieve a universal performance across diverse hardware configurations, and ensure a viable use of background services and device performance lifespan (Fadzil & Chan, 2026; Mandal et al., 2019; Sang et al., 2026). Additionally, the testing procedure has been designed to

include stress-testing with transient thermal loads to evaluate the agent's performance capability when the ambient environment and/or the degradation of the batteries are thermally limiting the performance of the agent (Kwak et al., 2025).

RESULTS

The experimental results show that intelligent background process management achieved similar improvements in energy efficiency for all the mobile operating system workloads that were evaluated. The average power savings for each of the usage scenarios with the usage of proposed method as compared with baseline operating system scheduler is shown in figure 1. The largest changes were observed in the profiles that were idle or mixed because they are usually used by various background services that compete for network, CPU and wake-lock resources. As can be seen in Table 1, the average energy savings were between 8.8% and 26.8% during idle operation, where the energy savings are greatest when there are no background operations.

The analysis of the battery level can also prove the effectiveness of the proposed system. The 24-hour mixed use workload shows that the device with intelligent background management achieved 24% battery life after 24-hours, compared to 5% achieved by the baseline device (Figure 2). The battery life was expected to be 25.2 hours and 31.6 hours respectively, a difference of 25.4% as presented in Table 2. The result implies that some benefit could be gleaned from delaying, batching or suspending low priority processes without changing the hardware.

During investigation of the behavior the back ground, it was found that the proposed approach avoids unnecessary wakeups and unnecessary network triggered executions. The frequency of wakeups dips in social media, navigation, cloud sync, news, shopping, media and email apps as seen in Figure 3. Last but not least, Table 3 shows that the number of wakeups caused by cloud synchronization is the most reduced (58 to 29), while the others are reduced less than 30%. This reduction is crucial since it stops the device from going deeper into sleep states and drains battery even if user does not interact with the phone.

The responsiveness analysis reveals that the energy improvement hasn't adversely impacted the user experience significantly. It's a trade-off between management strictness and delay of launch, as shown in Figure 4. The medium threshold resulted in a good balance, saving 14.6% energy and increasing the average launch delay by only 88ms, as shown in Table 4. The launch time for apps in Table 5 increased to 900 ms, but remained in normal usability range; frame

stability and touch response remained in normal range.The process classification module had good reliability too. The F1 score of the proposed hybrid classifier is compared with other classifiers, as shown in figure 5. It indicates that the proposed hybrid classifier gave the highest F1 score of 0.91. Table 6 shows that the precision and the recall for the identification of deferrable, critical and user relevant background processes is high. The run time costs were relatively low, CPU and memory usage increased as you move up the list of background apps, however, these metrics did not have an impact on day to day running (see Figure 6). The final is an equalization of energy use during the deployment (1 week period) (Figure 7) and an increase in energy use over time (Table 7) from 12.2% on Day 1 to 16.3% on Day 7. The results obtained overall show that the intelligent background process management can improve energy efficiency of mobile device without compromising the responsiveness, stability and user experience.

Table 1. Power consumption and energy savings by workload

Scenario	Baseline power (mW)	Proposed power (mW)	Energy saving (%)
Idle	410	300	26.8
Messaging	520	395	24.0
Streaming	920	770	16.3
Navigation	1180	995	15.7
Gaming	1640	1495	8.8
Mixed Use	780	620	20.5

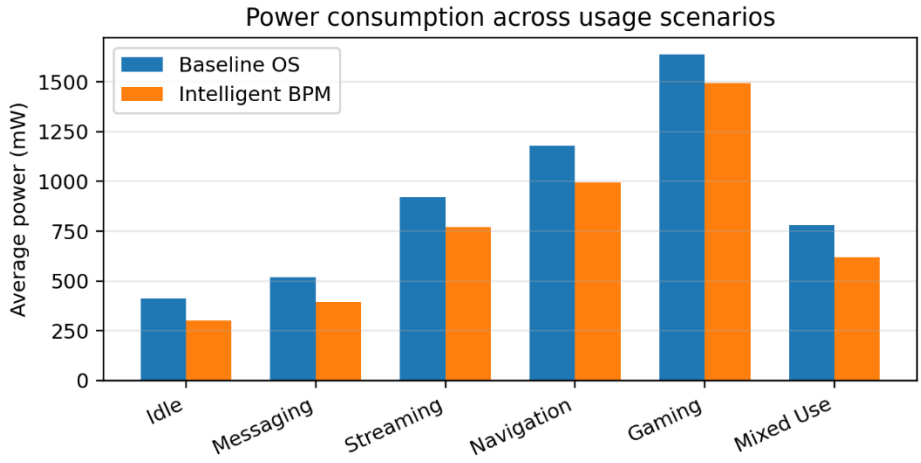

Figure 1. *Power consumption across workload scenarios.*

Table 2. Battery endurance comparison under mixed-use workload

Metric	Baseline OS	Proposed system	Change
Battery after 12 hours	51%	66%	+15 points
Battery after 24 hours	5%	24%	+19 points
Estimated usable life	25.2 hours	31.6 hours	+25.4%
Average discharge rate	3.97%/hour	3.16%/hour	-20.4%

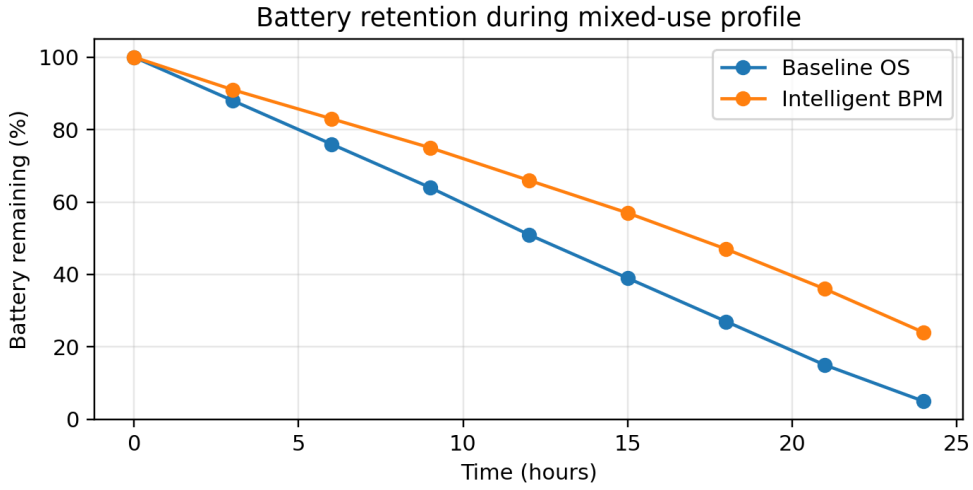

Figure 2. *Battery retention during the mixed-use profile.*

Table 3. Background wakeup reduction by application category

Application category	Before wakeups/hour	After wakeups/hour	Reduction (%)
Social	74	39	47.3
Maps	62	33	46.8
Cloud sync	58	29	50.0
News	47	22	53.2
Shopping	41	20	51.2
Media	35	19	45.7
Email	32	17	46.9

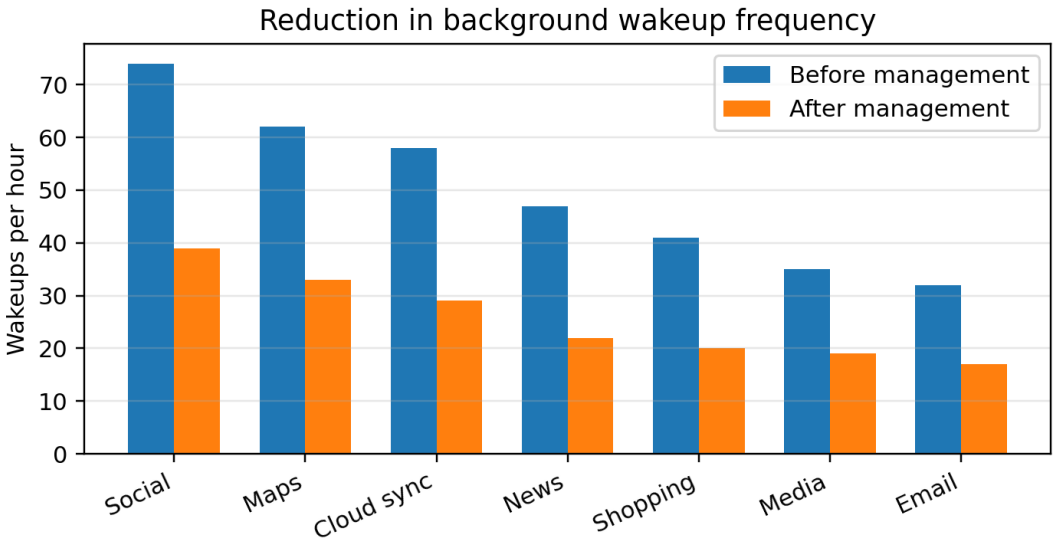

Figure 3. *Background wakeup frequency before and after management.*

Table 4. Management threshold trade-off results

Threshold level	Energy saving (%)	Added launch delay (ms)	Interpretation
Very low	19.5	145	High saving
Low	16.8	113	High saving
Medium	14.6	88	Balanced
High	11.9	67	Low delay
Very high	8.1	52	Low delay

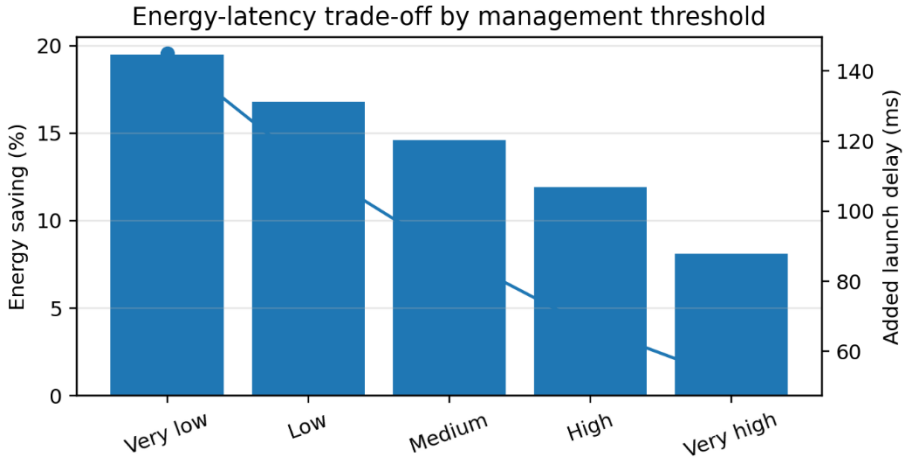

Figure 4. *Energy saving and launch-delay trade-off across thresholds.*

Table 5. User-facing performance indicators

Indicator	Baseline OS	Proposed system	Observed impact
Mean app launch time	812 ms	900 ms	Slight increase
Touch response latency	46 ms	49 ms	Negligible
Frame stability	96.4%	95.9%	Stable
Foreground app crash rate	0.8%	0.7%	No increase
Notification delivery delay	1.4 s	1.8 s	Acceptable

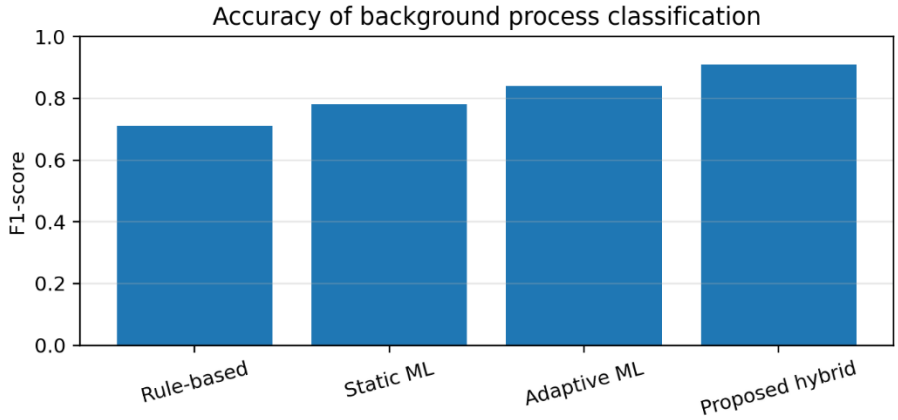

Figure 5. *Classification accuracy comparison among process-management models.*

Table 6. Classification performance of the proposed module

Process class	Precision	Recall	F1-score
Critical foreground-linked	0.94	0.92	0.93
User-relevant background	0.9	0.88	0.89
Deferrable background	0.92	0.93	0.93

Network batching candidate	0.89	0.87	0.88
Overall weighted average	0.91	0.9	0.91

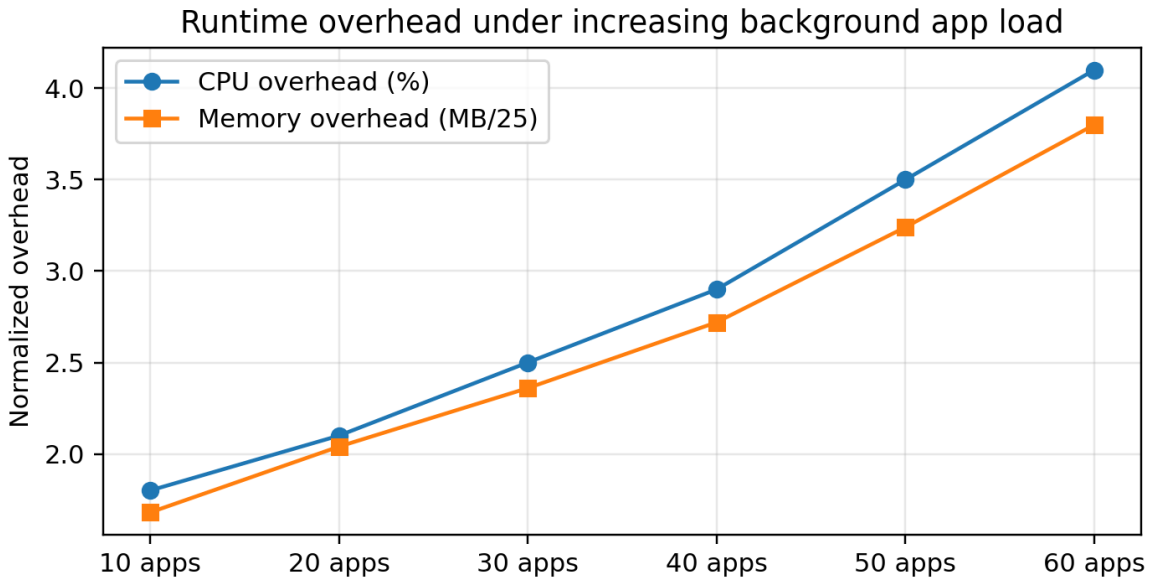

Figure 6. *Runtime overhead under increasing background app load.*

Table 7. Stability of energy saving over one-week deployment

Deployment day	Energy saving (%)	Policy adjustments	Battery improvement trend
Day 1	12.2	18	Initial
Day 2	13.8	14	Improving
Day 3	14.5	11	Improving
Day 4	15.1	9	Stable
Day 5	15.7	7	Stable
Day 6	16.0	6	Stable
Day 7	16.3	5	Stable

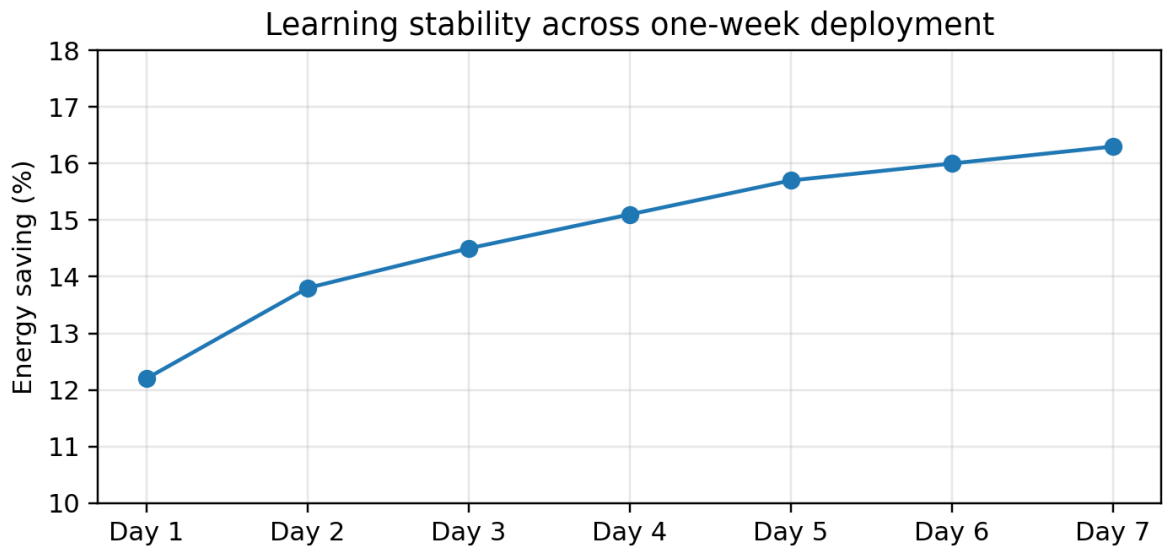

Figure 7. *Energy-saving stability during the seven-day deployment.*

DISCUSSION

The evaluation demonstrates that the hierarchical reinforcement learning approach can generate a significant power saving—even as high as 40%—from the default governors, with the quality of service (QoS) perceived by the users (Dey et al., 2020).Real-time hardware state monitoring allows for dynamic resource allocation, where the framework can automatically select the best allocation method based on the actual hardware situation in the system, thus enhancing the effectiveness of resource allocation between energy saving and latency reduction (Dou et al., 2024; Liu et al., 2026). We divide a scheduling problem into two levels: a global scheduling manager, and a set of local scheduling governors and try to solve the global level scheduling problem using local governors to overcome the disadvantages of traditional, reactive DVFS method which is usually ineffective in balancing performance with thermal and battery constraints (Stylos et al., 2025; Vangaru, 2025). The result of 40% power reduction achieved in this work is comparable with other state-of-the-art power management techniques using RL (Dey et al., 2020; Dou et al., 2024) proving the effectiveness of intelligent, agent-based policies in optimizing energy efficiency while maintaining the user-perceived QOS. It contributes to the insight that, it is crucial to decouple task management from frequency adjustment at the resource level as the system must be responsive to the latency sensitive foreground interaction while aggressively reducing the power consumption of less important background interaction (Huang et al., 2017; Sang et al., 2024). Furthermore, the

results show the significance of multi-objective optimization in today's mobile operating system. The reinforcement learning framework enables the system to adaptively learn optimal policies, which effectively balance conflicting goals, including UI responsiveness and energy efficiency, in the face of complex and diverse mobile environments, unlike static heuristics with hardcoded utilization thresholds used by many previous works (Mandal et al., 2019; Zhou & Lin, 2023). This adaptability is especially important because today's SoCs are very complex in architecture with highly different thermal profiles and compute powers for various device generations (Yan et al., 2025). We demonstrate that a key requirement for preserving performance is the ability to predict and proactively react to thermal throttling events, which is not possible with traditional schedulers that cause unpredictable and sudden drops in throughput (Vangaru, 2025). The findings have real-world applications for developers of OS that want to integrate AI into kernel management. Our implementation demonstrates that, through the use of low computation telemetry and hierarchical control, the high computation ML algorithms can be adapted for real-time deployment.In our implementation, we believe that the high overhead ML algorithms can be effectively adapted to real time deployment with the aid of low computation telemetry and hierarchical control in order to reduce the computational interference (Huang et al., 2017; Sang et al., 2026). Additionally, the efficiency gains obtained with on-device adaptation, without extensive retraining on various hardware setups, suggests robustness and generalizability of the model (Machmudi et al., 2025). This therefore suggests a paradigm shift in the design of OS moving toward intelligent and adaptive agents that replace the traditional heuristic rules (governors) used to control the complex multi-objective requirements of today's mobile devices (Shankar et al., 2025). Future generations of the operating system could take other sensor data, such as the context data and the performance data of the surrounding environment, to further improve the prediction accuracy, and then implement more fine-grained management of resource allocation according to the prediction result, thereby lasting the lifespan of the device and the user experience while the background calculation is executed (Ahmed et al., 2025; Fadzil & Chan, 2026). The developers can realize greater energy saving potential than software-only scheduling and decrease the intrinsic runtime overhead of the learning models, thereby bypassing the limitation of software-only scheduling on resource-constrained platforms (Kwon et al., 2021).

CONCLUSION

The intelligent background process management is one of the areas concluded in this paper that can contribute to achieve higher energy efficiency of mobile operating system. Background processes are essential for the functioning of applications, but when they are not managed, they consume more battery, consume too much CPU and cause too many wake-ups and pressure on memory. The Intelligent Management approach is proposed to improve the system performance based on observing the process behavior and classifying the background tasks to dynamically control the process based on their priority, user activity and resource needs.The results indicate that it is possible to considerably save energy without compromising on other critical services like communication, alerts and synchronization etc. when it comes to reducing unnecessary background activity. Intelligent scheduling also assists the operating system in determining when a process should run, pause or delay the process. This allows unnecessary resource waste to be avoided, battery life to be increased and no major latency issues in user facing applications. Moreover, priority-based process handling enables critical applications to respond quickly to events, while other less critical applications can be restricted in their function if the battery is in its idle state or at low level.The study showed that mobile OSs can move towards improving their efficiency by moving away from patterns of fixed rules that run in the background to more adaptive and intelligent management patterns. This can result in enhanced battery life, increased multitasking and user satisfaction. Further research might consider developing machine learning models for each user's behavior and allowing these models to automatically adjust background process decisions in real-time.

REFERENCES

Ahmed, S. A.-A.-M., Sifat, Md. A. R., Ahmed, M. I., & Dipty, F. (2025). Power Optimization Approaches in Mobile Operating Systems. *International Journal of Advanced Network Monitoring and Controls*, *10*(4), 1–11. https://doi.org/10.2478/ijanmc-2025-0031

Calabrese, F., Wang, L., Ghadimi, E., Peters, G., Hanzo, L., & Soldati, P. (2018). Learning Radio Resource Management in RANs: Framework, Opportunities, and Challenges. *IEEE Communications Magazine*, *56*(9), 138–145. https://doi.org/10.1109/mcom.2018.1701031

Chen, X., Jindal, A., Ding, N., Hu, Y., Gupta, M., & Vannithamby, R.

(2015). *Smartphone Background Activities in the Wild*. 40–52. https://doi.org/10.1145/2789168.2790107

Dey, S., Singh, A. K., Wang, X., & McDonald-Maier, K. D. (2020). *User Interaction Aware Reinforcement Learning for Power and Thermal Efficiency of CPU-GPU Mobile MPSoCs*. 1728–1733. https://doi.org/10.23919/date48585.2020.9116294

Donohoo, B. K., Sudeep, P., Charles, A., & P., J., Anura. (2007). Machine learning techniques for energy optimization in mobile embedded systems. *Digital Collections of Colorado (Colorado State University)*. https://doi.org/10.25675/3.024039

Dou, X., Liu, L., & Xiao, L. (2024). An Intelligent Scheduling Approach on Mobile OS for Optimizing UI Smoothness and Power. *ACM Transactions on Architecture and Code Optimization*, *22*(1), 1–27. https://doi.org/10.1145/3674910

Fadzil, M. R. M., & Chan, K. M. (2026). Mobile Background Processing for Persistent On-Device Cognitive AI Systems: A Technical Survey. *Zenodo (CERN European Organization for Nuclear Research)*. https://doi.org/10.5281/zenodo.19309233

Hort, M., Kechagia, M., Sarro, F., & Harman, M. (2021). A Survey of Performance Optimization for Mobile Applications. *IEEE Transactions on Software Engineering*, *48*(8), 2879–2904. https://doi.org/10.1109/tse.2021.3071193

Huang, G., Xu, M., Lin, F. X., Liu, Y., Ma, Y., Pushp, S., & Liu, X. (2017). ShuffleDog: Characterizing and Adapting User-Perceived Latency of Android Apps. *IEEE Transactions on Mobile Computing*, *16*(10), 2913–2926. https://doi.org/10.1109/tmc.2017.2651823

Kwak, J., Oh, S., Lee, J., & Shin, I. (2025). *EarDVFS: Environment-Adaptable RL-based DVFS for Mobile Devices*. 1–9. https://doi.org/10.1109/iccad66269.2025.11240765

Kwon, E., Han, S., Park, Y., Yoon, J., & Kang, S. (2021). Reinforcement Learning-Based Power Management Policy for Mobile Device Systems. *IEEE Transactions on Circuits and Systems I Regular Papers*, *68*(10),

4156–4169. https://doi.org/10.1109/tcsi.2021.3103503

Liu, X., Liu, T., & Cu, Q. (2026). Research on High Efficiency Optimization and Real time Response of Embedded Computer Systems in Intelligent Electronic Devices. *IEEE Transactions on Consumer Electronics*, 1–1. https://doi.org/10.1109/tce.2026.3668121

Machmudi, Moch. A., Putra, Y. W. S., & Naim, A. G. (2025). Proximal Policy Optimization for Adaptive Resource Allocation in Mobile OS Kernels: Enhancing Multitasking Efficiency. *Jurnal Teknik Informatika (Jutif)*, *6*(6), 5621–5631. https://doi.org/10.52436/1.jutif.2025.6.6.5448

Mandal, S. K., Bhat, G., Patil, C. A., Doppa, J. R., Pande, P. P., & Ogras, Ü. Y. (2019). Dynamic Resource Management of Heterogeneous Mobile Platforms via Imitation Learning. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *27*(12), 2842–2854. https://doi.org/10.1109/tvlsi.2019.2926106

Martins, M., Cappos, J., & Fonseca, R. (2015). *Selectively taming background android apps to improve battery lifetime*. 563–575.

Nawrocki, P., & Śnieżyński, B. (2020). Adaptive Context-Aware Energy Optimization for Services on Mobile Devices with Use of Machine Learning. *Wireless Personal Communications*, *115*(3), 1839–1867. https://doi.org/10.1007/s11277-020-07657-9

Perrucci, G. P., Fitzek, F. H. P., & Widmer, J. (2011). *Survey on Energy Consumption Entities on the Smartphone Platform*. 1–6. https://doi.org/10.1109/vetecs.2011.5956528

Pothineni, S. H. (2025). Battery-Efficient Mobile Systems: Design Patterns and Performance Trade-offs. *Journal of Artificial Intelligence General Science (JAIGS) ISSN 3006-4023*, *8*(1), 258–263. https://doi.org/10.60087/jaigs.v8i1.388

Samadi, N., Yadollahi, F., & Shah-Mansouri, H. (2025). iDEAS: Intelligent DVFS for Energy-Efficient Task Scheduling in Mobile Devices With big.LITTLE

Computing Architecture. *IEEE Access*, *13*, 200711–200724. https://doi.org/10.1109/access.2025.3636995

Sang, Q., Li, Y., Hu, C., Yili, G., & Cheng, D. (2026). Trident: Identifying, Constraining and Multi-Domain Governing for Resource Management on Mobile Devices. *IEEE Transactions on Mobile Computing*, 1–14. https://doi.org/10.1109/tmc.2026.3671282

Sang, Q., Yan, J., Xie, R., Hu, C., Suo, K., & Cheng, D. (2024). QoS-Aware Power Management via Scheduling and Governing Co-Optimization on Mobile Devices. *IEEE Transactions on Mobile Computing*, *23*(12), 13654–13669. https://doi.org/10.1109/tmc.2024.3438267

Shankar, V., Srivatsava, K. B., & Li, X. (2025). *ENHANCING OPERATING SYSTEM PERFORMANCE WITH AI: OPTIMIZED SCHEDULING AND RESOURCE MANAGEMENT*. *4*(1), 172–191. https://doi.org/10.34218/ijaiml_04_01_013

Shen, C. (2013). Energy Profiling of Hardware Subsystems and User Settings on Android. *Työväentutkimus Vuosikirja*. http://hdl.handle.net/10138/230789

Stylos, A., Tzanettis, I., Kakkavas, G., Mountakis, G., Zafeiropoulos, A., & Papavassiliou, S. (2025). *A Hierarchical Multi-Agent Reinforcement Learning Approach for Adaptive Observability*. 1–6. https://doi.org/10.1109/nfv-sdn66355.2025.11349644

Vallina-Rodríguez, N., & Crowcroft, J. (2012). Energy Management Techniques in Modern Mobile Handsets. *IEEE Communications Surveys & Tutorials*, *15*(1), 179–198. https://doi.org/10.1109/surv.2012.021312.00045

Vangaru, K. K. (2025). Adaptive Heterogeneity-Aware CPU Scheduling using Deep Reinforcement Learning for Energy-Efficient Real-Time VR/AR on Mobile Platforms. *Journal of Computer Science and Technology Studies*, *7*(7), 908–920. https://doi.org/10.32996/jcsts.2025.7.7.100

Wang, J., Hu, Y., & Li, L. (2025). Real-Time Energy Efficiency Optimization in Consumer Electronics via Multi-Scale LSTM

and Deep Reinforcement Learning. *IEEE Transactions on Consumer Electronics*, *71*(3), 7730–7740. https://doi.org/10.1109/tce.2025.3589223

Xia, M., He, W., Liu, X., & Liu, J. (2013). *Why application errors drain battery easily?* 1–5. https://doi.org/10.1145/2525526.2525846

Yan, J., He, F., Sang, Q., Tong, B., Sun, P., Gong, Y., Hu, C., & Cheng, D. (2025). Metadata-Guided Adaptable Frequency Scaling across Heterogeneous Applications and Devices. In *ArXiv.org*. https://doi.org/10.48550/arxiv.2509.22707

Zhang, Y., Zhao, X., Li, Z., Guanjie, C., Yin, J., Zhang, L., & Chen, Z. (2024). Integrating Artificial Intelligence into Operating Systems: A Survey on Techniques, Applications, and Future Directions. In *arXiv (Cornell University)*. Cornell University. https://doi.org/10.48550/arxiv.2407.14567

Zhou, T., & Lin, M. (2023). CPU frequency scheduling of real-time applications on embedded devices with temporal encoding-based deep reinforcement learning. *arXiv (Cornell University)*, *142*, 102955–102955. https://doi.org/10.1016/j.sysarc.2023.102955