Optimizing Battery Usage in iOS App

Sri Vidya Vilipala

Northwest Missouri State University
s576118@nwmissouri.edu

Abstract. As iOS apps become more complex, they often use more battery, which can frustrate users and harm their experience. This research focuses on finding ways to reduce battery drain in iOS applications without slowing them down. By studying how apps use the phone's processor, connect to the Internet, and run in the background, we suggest practical ways to save battery. These include better coding practices, smarter management of background tasks, and using Apple's built-in power saving features like Low Power Mode. The goal is to help developers create apps that perform well but do not drain the battery too quickly, leading to happier users and longer-lasting devices. By combining these strategies, developers can strike a balance between app performance and energy consumption, ensuring a smoother and more sustainable user experience. In order to satisfy the needs of today's mobile consumers, this paper offers developers practical advice and resources for creating apps that are string and energy efficient.

Keywords: Battery optimization · iOS development · energy-efficient coding · power consumption · mobile applications.

1 Introduction

As mobile apps become increasingly complex and feature rich, managing power consumption has become a critical challenge in iOS development. High power consumption can lead to rapid battery depletion, negatively affecting user experience and potentially reducing user engagement. With users relying on mobile devices for extended periods, optimizing battery efficiency has become a priority for developers aiming to create high-performance, energy-efficient applications.

The key challenge addressed in this research is the significant power consumption in iOS apps, which impacts battery life and user satisfaction. Despite the growing importance of energy efficiency, many applications do not implement effective strategies to reducing battery drain. This research seeks to identify the underlying causes of excessive energy consumption and propose solutions for optimizing battery usage in iOS applications.

This study explores strategies for optimizing battery usage by focusing on energy-efficient coding practices, optimizing background task management, and utilizing power-saving technologies such as Low Power Mode and efficient network calls. By analyzing how different aspects of app behavior—such as CPU

usage, network activity, and screen rendering—affect power consumption, developers can implement optimizations that balance performance with energy efficiency.

The aim of this research is to identify and implement power-saving techniques that reduce unnecessary energy usage without compromising app functionality, responsiveness, or usability. By integrating efficient coding patterns and leveraging Apple's built-in power management features, developers can create sustainable, high-quality applications that meet modern user expectations.

2 Review

Battery drain is a big problem for iOS apps. It happens because apps run processes in the background, use too much CPU or GPU, make many network requests, or track location inefficiently. Many applications do not optimize their power consumption, which drains the battery faster and frustrates users.

Current solutions, like Low Power Mode, better background task management, network optimization, smoother UI rendering, and smarter location tracking, do help save battery. However, these methods are often inflexible and do not adapt well, making it difficult to balance performance and battery life.

A smarter solution is needed, one that uses AI to monitor energy usage and make real-time adjustments. In this way, apps can dynamically optimize their behavior to save battery without sacrificing performance.

2.1 Existing Problem

Battery drain is a common issue in iOS apps, affecting user experience and device life. The main causes include inefficient background tasks, heavy CPU and GPU usage, excessive network activity, and poor management of location services. Apps often run background processes unnecessarily, which drain power even when not in use. High-demand tasks like gaming or video editing use too much CPU and GPU, quickly draining the battery. Frequent network requests and continuous location tracking also contribute to power loss. Many apps don't adjust their energy consumption based on battery levels, making the problem worse. Developers need to adopt smarter power management strategies to balance performance and energy use.

2.2 Existing Solution

Low Power Mode (LPM), one of Apple's built-in features, helps iOS apps use less power. When the battery is low, LPM limits background activity, turns off animations, and lowers CPU and GPU performance. However, LPM only acts when the user enables it, and developers are unable to modify it. Developers can utilize tools like Background Fetch and the Background Task API to better manage background processes. Additionally, push notifications assist save battery by reducing the need for programs to continuously look for updates.

Although real-time apps like chat and live streaming still need a lot of network activity, apps can use adaptive streaming, batch requests, and cache data.

Developers can use SwiftUI, which uses less power than UIKit, to optimize CPU and GPU consumption. They can also use techniques like lazy loading and lowering frame rates to lessen the processor's stress. Utilizing the Significant Location Change API, which only updates location when the user travels a considerable distance, allows location-based apps to save power. Apps that need real-time location information, such as ride-hailing services or navigation, still struggle to strike a balance between accuracy and power usage.

Despite these solutions, most power-saving features lack adaptability and don't adjust based on real-time app usage or battery conditions. A more dynamic, AI-driven approach to power management could analyze app activity in real time and adjust energy consumption accordingly. This would help apps save battery without compromising performance or the user experience, providing a more flexible and efficient solution.

3 Approach to the Problem

To reduce battery drain in iOS apps, a smarter AI-powered system can be used. This system adjusts an app's power usage based on real-time factors like battery level, network strength, user activity, and app usage. Unlike fixed power-saving methods, this AI learns and adapts to how the app is being used at any moment.

The main goal is to create a smart system that saves battery without slowing down the app or affecting the user experience. Instead of using the same power-saving settings for everyone, it customizes them based on real-time data.

3.1 How this approach work

The system works by continuously monitoring and collecting real-time data on various factors that impact battery consumption. It tracks CPU and GPU usage, network activity, background tasks, and location services to understand how the app consumes power. Additionally, it monitors battery percentage and charging status to adjust energy usage based on the device's current condition. By analyzing user behavior, such as how often an app is opened and whether sessions are long or short, the system can identify patterns and make smarter power-saving decisions.

Using this data, the adaptive power management system makes real-time adjustments to optimize battery usage. If an app is in use while the battery is low, the system may reduce unnecessary animations, lower frame rates, or switch to a lighter UI mode to conserve power. When the app is running in the background, it can limit background processes, delay non-essential updates, or reduce the frequency of network requests to prevent unnecessary battery drain. For location-based apps, the system can automatically switch to lower GPS accuracy when high precision isn't needed, significantly reducing power consumption while maintaining basic location tracking.

Another key feature is dynamic network optimization, which helps reduce energy use by managing how and when the app connects to the internet. When the app is idle, it can reduce the frequency of data fetching to prevent constant network activity. Instead of sending multiple small network requests, the system batches requests together and prioritizes important updates over less critical ones. For streaming apps, the system can adjust video quality based on battery levels and network conditions, ensuring a smooth experience while saving energy. Additionally, the system customizes power management based on user behavior, allowing it to proactively enable a power-saving mode if a user frequently runs the app on low battery. If certain features are rarely used, they can be automatically deprioritized in the background, ensuring that only necessary processes consume power. This intelligent and user-centric approach helps apps run efficiently while preserving battery life.

3.2 How this Approach is Different from Previous Solutions

The new AI-driven approach to battery optimization offers considerable improvement over previous power-saving mechanisms in that it dynamically and real-time adjusts, based on the usage patterns. Unlike traditional approaches with static solutions such as Low Power Mode or background task limitations, the AI system continuously monitors app usage and battery health and makes more informed decisions to save energy without reducing performance. Instead of applying the same power-saving rules to all users, the system based on AI personalizes battery-saving measures according to user behavior for an optimized and user-friendly experience.

When it comes to network efficiency, previous solutions relied on manual optimisations like batching and caching, which were efficient but inflexible. The AI platform, on the other hand, utilises smart scheduling of network requests and adaptive data fetching, i.e., it can reduce the data transfer rate when the app is idle and distribute high-priority updates when needed. In the same way, location services were earlier based on shared APIs for background updates, but the new AI-driven model dynamically adapts GPS accuracy based on motion patterns and significantly reduces power consumption without sacrificing location capability.

Previous approaches to controlling the CPU and GPU maximized performance by lowering frame rates and utilizing SwiftUI, but such updates were static and global rather than dynamic. In contrast, the AI system can dynamically balance performance in real-time to lower background processing or graphical effects as necessary. The main difference is that the AI approach continues to learn and adapt, making energy-saving decisions that vary based on user usage patterns, whereas traditional solutions are not able to change based on present battery conditions. Power management is therefore made smarter, extending battery life without affecting user experience.

4 A solution to problem

Our proposed solution focuses on a smart, data-driven approach to optimizing battery consumption in iOS apps. By utilizing real-time power monitoring, AI-driven predictions, and adaptive scheduling, we aim to ensure that apps consume energy efficiently without compromising their performance. This approach goes beyond traditional methods, dynamically adjusting to the app's actual usage, the device's battery level, and user behavior.

To start, energy consumption is continuously monitored using tools like Xcode Energy Log and Instruments Power Profiler. These tools track how much power is used by the CPU, GPU, and network during app usage. By analyzing this data, we can identify the specific components of the app that are consuming the most energy and focus on optimizing them individually to reduce unnecessary battery drain.

Background tasks are another area where energy efficiency can be significantly improved. With the help of AI, background activities are prioritized based on the user's behavior and the current battery status. Non-essential tasks are postponed or minimized when the battery is running low, ensuring that power is only used for the most critical processes. This dynamic scheduling helps reduce the overall energy usage without affecting the app's core functionality.

Network usage is also optimized through AI-driven predictions. Machine learning models can predict user behavior, allowing the app to adjust how frequently it fetches data. By grouping network requests into batches, the app reduces the need to frequently wake up the device's radio, which can be a major source of battery drain. This optimization ensures that data fetching happens only when necessary, reducing energy consumption without compromising the app's responsiveness.

For the user interface, power efficiency is achieved by adjusting the UI rendering based on the battery level. When the battery is low, the app reduces the refresh rate and lowers the intensity of animations, which in turn reduces the load on the GPU. Non-essential visual elements are also disabled during these times to further conserve power.

Finally, location services are managed intelligently. The app dynamically switches between GPS, Wi-Fi, and Cellular data depending on the user's movement and activity patterns. The accuracy of location tracking is adjusted to the needs of the app, ensuring that it uses minimal power while still providing accurate location information when necessary.

By combining these strategies, the app can offer a seamless user experience while minimizing battery consumption. This AI-driven approach ensures that the app adapts to the user's behavior and real-time conditions, striking a perfect balance between performance and energy efficiency.

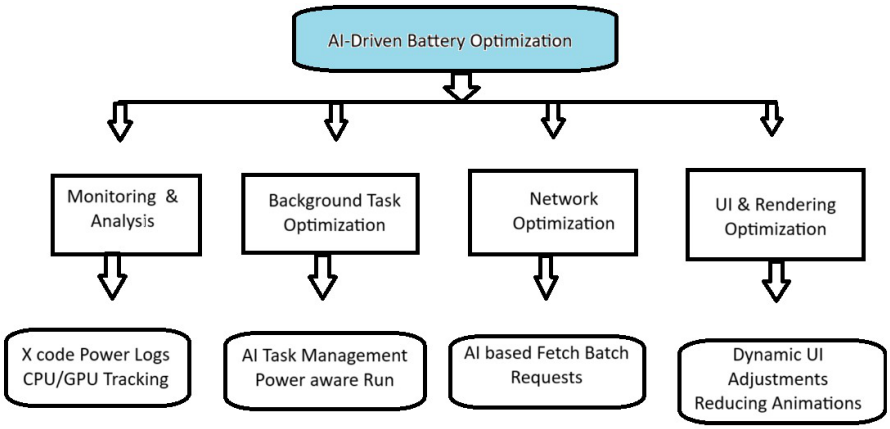

Fig. 1. AI-Driven Battery Optimization Workflow

Table 1. Energy-Saving Techniques and Their Impact on iOS App Efficiency.

Component	Optimization Strategy	Benefits
Real-Time Power Monitoring	Real-time CPU/GPU/network tracking	Target high-drain components
Background Task Management	AI prioritizes critical tasks	Reduces background energy drain.
Network Optimization	Batched, adaptive fetching	Minimizes radio wake-ups.
UI Rendering Adjustments	Dynamic frame rate/effects adjustment	Lowers GPU load
Location Services	Smart accuracy switching	Balances precision/power
Adaptive Learning	Learns behavior patterns	Personalizes savings

5 Future Scope

Future studies can concentrate on AI-powered adaptive power management, which optimizes energy use according to actual usage. Models for machine learning could forecast power requirements and dynamically modify the behavior of apps. Developing frameworks for energy-efficient code can make it easier for developers to design optimized programs.

More research on hardware-software integration may enable improved chipset-level power control. Battery efficiency may also be increased by energy-harvesting and wireless charging methods. These developments will support more sustainable power use and aid in the development of intelligent, energy-efficient iOS apps.

6 Conclusion

As mobile apps become more advanced, saving battery life has become a major concern for iOS developers. This research looked into the main reasons why apps use too much battery and explored current solutions—finding that many of them don't work well in real-time situations.

To solve this, the study suggests a smarter, AI-based approach. By using machine learning, real-time tracking, and smart scheduling, apps can adjust how they work based on how the user behaves, the condition of the device, and how the app is being used. Features like smarter UI updates, better handling of background tasks, and predicting network usage can all help save battery while keeping the app running smoothly.

This new method focuses on making apps more personal and aware of their environment. Not only does it make the user experience better by saving battery, but it also supports eco-friendly app design. As AI and mobile technology keep improving, developers will have more tools to make apps even more energy-efficient—leading to smarter and greener apps in the future.

References

1. Cito, J., Rubin, J., Stanley-Marbell, P., Rinard, M.: Battery-aware transformations in mobile applications. In: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering. Association for Computing Machinery (2016)
2. Jacques, V.M.F., Alizadeh, N., Castor, F.: A study on the battery usage of deep learning frameworks on ios devices. In: Proceedings of the IEEE/ACM 11th International Conference on Mobile Software Engineering and Systems. Association for Computing Machinery (2024)
3. Jindal, A., Hu, Y.C.: Experience: Developing a usable battery drain testing and diagnostic tool for the mobile industry. GetMobile: Mobile Comp. and Comm. (2022)
4. Kwak, J., Lee, S., Jeong, D.R., Kumar, A., Shin, D., Kim, I., Shin, D., Lee, K., Lee, J., Shin, I.: Mixmax: Leveraging heterogeneous batteries to alleviate low battery experience for mobile users. In: Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services. Association for Computing Machinery (2023)
5. Li, X., Chen, J., Liu, Y., Wu, K., Gallagher, J.P.: Combatting energy issues for mobile applications. ACM Transactions on Software Engineering and Methodology (2023)
6. Oliveira, W., Moraes, B., Castor, F., Fernandes, J.P.: Analyzing the resource usage overhead of mobile app development frameworks. In: Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering. Association for Computing Machinery (2023)
7. Rosen, S., Nikravesh, A., Guo, Y., Mao, Z.M., Qian, F., Sen, S.: Revisiting network energy efficiency of mobile apps: Performance in the wild. In: Proceedings of the 2015 Internet Measurement Conference. Association for Computing Machinery (2015)

8. Shao, Y., Wang, R., Chen, X., Azab, A.M., Mao, Z.M.: A lightweight framework for fine-grained lifecycle control of android applications. In: Proceedings of the Fourteenth EuroSys Conference 2019. Association for Computing Machinery (2019)
9. Wang, J., Wu, G., Wu, X., Wei, J.: Detect and optimize the energy consumption of mobile app through static analysis: an initial research. In: Proceedings of the Fourth Asia-Pacific Symposium on Internetware. Association for Computing Machinery (2012)
10. Yadav, A., Usman, M., Sati, A., Jain, S.: Revolutionizing software development: Enhancing quality and performance through code refactoring. In: Proceedings of the 2024 Sixteenth International Conference on Contemporary Computing. Association for Computing Machinery (2024)

[1],[2],[3],[4],[5],[6],[7],[8],[9],[10]