Adaptive DVFS and Display Brightness Scaling for Maximising User Experience on Mobile Devices

SAMUEL ISUWA, University of Southampton, United Kingdom of Great Britain and Northern Ireland
SOMDIP DEY, University of Essex, United Kingdom of Great Britain and Northern Ireland
AMIT KUMAR SINGH, University of Essex, United Kingdom of Great Britain and Northern Ireland
BASHIR M. AL-HASHIMI, King's College London, United Kingdom of Great Britain and Northern Ireland
GEOFF V. MERRETT, University of Southampton, United Kingdom of Great Britain and Northern Ireland

As mobile devices become ubiquitous, the need for improved quality of experience (QoE) continues to place increasing demands on these devices. Delivering consistent and high QoE while managing various applications and user requirements on these energy-constrained devices remains a significant challenge. Dynamic voltage and frequency scaling (DVFS) and display brightness scaling are two commonly used power-saving techniques. We argue that existing approaches are not efficient in maximising user experience, especially in handling a variety of applications with varying requirements while meeting users' daily battery life expectations. In this article, we propose an adaptive approach that dynamically adjusts DVFS and display brightness scaling to maximise the user experience by considering the user's expected battery life and a wide variety of mobile workload characteristics. Our method employs K-means clustering to classify diverse workloads at runtime and then assigns the appropriate DVFS and brightness settings based on the clusters and contexts of the workload and energy budget to achieve the expected battery life. Experimental evaluations conducted on real-world mobile devices demonstrate the effectiveness of our approach, with average improvements in QoE and battery life of up to 9 % and 35 %, respectively, compared to the state-of-the-art.

CCS Concepts: • **Computer systems organization → Embedded systems**; • **Hardware → On-chip resource management**.

Additional Key Words and Phrases: Adaptive DVFS, adaptive brightness, resource management, DVFS and brightness scaling, battery life, quality of experience (QoE), user experience

ACM Reference Format:
Samuel Isuwa, Somdip Dey, Amit Kumar Singh, Bashir M. Al-Hashimi, and Geoff V. Merrett. 2026. Adaptive DVFS and Display Brightness Scaling for Maximising User Experience on Mobile Devices. *ACM Trans. Embedd. Comput. Syst.* 25, 6, Article 98 (November 2026), 27 pages. https://doi.org/10.1145/3842746

Authors' Contact Information: Samuel Isuwa, School of Electronics and Computer Science, University of Southampton, Southampton, Hampshire, United Kingdom of Great Britain and Northern Ireland; e-mail: si1g19@soton.ac.uk; Somdip Dey, School of Computer Science and Electronic Engineering, University of Essex, Colchester, Essex, United Kingdom of Great Britain and Northern Ireland; e-mail: somdip.dey@essex.ac.uk; Amit Kumar Singh, Computer Science and Electronic Engineering, University of Essex, Colchester, Essex, United Kingdom of Great Britain and Northern Ireland; e-mail: a.k.singh@essex.ac.uk; Bashir M. Al-Hashimi, King's College London, London, London, United Kingdom of Great Britain and Northern Ireland; e-mail: bashir.al-hashimi@kcl.ac.uk; Geoff V. Merrett (corresponding author), School of Electronics and Computer Science, University of Southampton, Southampton, Hampshire, United Kingdom of Great Britain and Northern Ireland; e-mail: gvm@ecs.soton.ac.uk.



This work is licensed under a Creative Commons Attribution 4.0 International License.
© 2026 Copyright held by the owner/author(s).
ACM 1539-9087/2026/11-ART98
https://doi.org/10.1145/3842746

1 Introduction

Mobile devices such as smartphones and tablets have become an essential part of our daily lives, with users relying on them for communication, entertainment, and productivity. The increasing functionalities coupled with growing user demands for higher performance and improved user experience (also referred to as quality of experience, QoE), have led to a significant increase in power consumption. This poses a major challenge to the battery technology of these devices, which historically advances at a slower relative pace [31, 37, 42]. Moreover, the varying diversity of user patterns combined with the proliferation of applications with diverse requirements further contribute to this problem [25, 37]. To mitigate these issues, various techniques have been employed. **Dynamic Voltage and Frequency Scaling** (**DVFS**) is one such technique [42], which effectively reduces the power consumption of processing elements by dynamically adjusting the operating voltage and frequency. Another approach is dynamic display brightness scaling [8, 22], which reduces energy consumption by dynamically adjusting screen brightness.

The process of identifying the appropriate power-to-QoE tradeoff in response to a user's expected battery life and energy demands is a critical element in addressing this challenge. This necessitates learning user usage patterns across various scenarios and applying suitable configurations to meet the expected battery life [25]. However, this task becomes increasingly complex as users continuously explore diverse sets of applications, each with distinct characteristics and requirements within a single battery discharge cycle, which may not necessarily require similar settings with regards to DVFS or brightness level. For example, in a typical usage scenario, users may move from checking their email to responding to social media messages or browsing the web for news and advertisements. Moreover, their environment (context) may vary from indoor normal light to outdoor bright light or even a dark environment, depending on the time of the day. Therefore, there is a need for a resource management approach that adapts continuously to changing application and context conditions.

Existing approaches [5, 9, 38] to address this involve classifying applications into clusters or canonical workload phases based on their unique characteristics, and then applying an appropriate management framework to address power and performance tradeoff concerns. Commonly employed approaches utilise parameters sourced from **Performance Monitoring Counters** (**PMCs**), including but not limited to instruction execution, branch-misses, cache misses, and cache references. Alternatively, some approaches rely on basic parameters, such as power samples obtained from respective processing elements (e.g., CPU and GPU) to effectively capture the unique characteristics of the application. Categorisation is commonly performed offline, online, or through a combination of both approaches. These classifications, combined with runtime utilisation of DVFS or display scaling using heuristic or machine learning methods, have been shown to improve the user experience on mobile devices.

However, these existing approaches have limitations. First, many are developed and evaluated on legacy research platforms (e.g., the Odroid-XU3, released in 2014) or rely on PMCs [5, 10, 14, 20, 38], which, while offering rich instrumentation capabilities, are architecture specific and non-standardised, often requiring bespoke implementations for each platform, and whose features are frequently obsolete or not readily available on most commercial off-the-shelf mobile devices. For example, fine-grained power metrics for individual processing components (such as CPU, GPU, etc.,) that are easily accessible on the Odroid-XU3 platform are often unavailable on commercial smartphones due to restricted access to **Hardware Abstraction Layers** (**HAL**) or the lack of standardised APIs for per-component power rails across different vendors [37]. Second, these strategies typically rely on individual subsystems, such as processing elements (e.g., CPU, GPU) [9–11, 15, 25, 27–29, 34, 38] or the display subsystem [3, 8, 22, 24, 33, 44], in isolation. As a result, they fail to holistically address system-wide power–performance tradeoffs that reflect real user

behaviour and expectations. Consequently, they often either fall short of meeting user's expected battery life or offer overly constrained QoE policies. Third, and most crucially, most prior works do not consider user-specific battery life expectations in their adaptation strategies. Existing models often aim for aggressive energy conservation or performance maximisation without explicitly considering when and how power-saving tradeoffs should be applied to meet user-defined battery life goals. As a result, these approaches often prove ineffective, particularly under dynamic runtime conditions or when user behaviour changes significantly, as we demonstrate in this article.

To address the aforementioned limitations, this article proposes an adaptive DVFS and display brightness scaling approach that is both context-aware and user-centric, aiming at maximising user QoE across expected battery life. In contrast to prior work, our approach does not assume universal optimisation objectives; instead, it adapts to individual user expectations regarding battery life, workload variations, and ambient (context) conditions. The proposed approach is designed for practical deployment on commercial off-the-shelf mobile devices, leveraging readily accessible system metrics (e.g., CPU/GPU utilisation, temperature, and memory frequency). It performs online classification of workloads and contextual conditions to dynamically select appropriate DVFS and display brightness configurations in real time, thereby maximising QoE while meeting the expected battery life. The article makes the following contributions:

(1) We propose an efficient K-means-based workload clustering approach that enables runtime classification of workloads and context identification on mobile devices.
(2) We show, using both qualitative and quantitative methods, the impact of frequency capping on user experience in a variety of real-world mobile applications, and based on these insights, derive a QoE parameter relationship.
(3) To meet the expected battery life, we propose an adaptive DVFS and brightness scaling manager that adaptively assigns appropriate configurations to applications based on workload clusters, contextual factors, and energy constraints.
(4) Experimental evaluation of the proposed approach on commercial off-the-shelf mobile devices with commonly used Android mobile applications demonstrates a QoE and battery life improvement of 9 % and 35 % over the state-of-the-art.

The remainder of this article is organised as follows: Section 2 presents the related work. Section 3 provides a motivational case study and the perceived QoE–resource relationship. Section 4 describes the proposed approach, including the clustering technique combined with the adaptive DVFS and display brightness scaling manager. Section 5 presents the experimental results, including a comparison with existing methods. Section 6 discusses the limitations and generalisability of the framework across modern mobile architectures and displays. Finally, Section 7 concludes the article.

2 Related Work

Existing research has explored workload-aware resource management [5, 9, 12, 14, 20, 38] to meet the varied demands of applications through diverse technical implementations. For instance, a multinomial logistic regression (MLR)-based classification algorithm to categorise workloads at runtime using CPU cycles was proposed in [14, 20]. In these works, workload characterisation and classification into pre-defined groups are performed offline to inform subsequent DVFS decision strategies. Similarly, Cochran *et al.* [12] utilised performance counters for offline characterisation, employing an MLR classifier to predict the workload associated with specific applications at runtime. Chetoui *et al.* [9] proposed a clustering-based technique to identify canonical workload phases using performance counters; these phases are identified offline via benchmark suites while monitoring CPU and GPU events. At runtime, the workload-aware governor identifies the active phase and applies optimal frequency settings to balance user experience with battery life. Further research has

introduced online learning-based task scheduling [5, 16] and analysed the bursty or throughput-oriented characteristics of mobile computations [38]. These studies aim at identifying lengthy computations susceptible to thermal throttling that significantly impact user experience. While these approaches effectively balance performance and power consumption, they focus primarily on the dynamic adjustment of voltage and frequency based on specific workload characteristics. A critical barrier to the deployment of such methods on off-the-shelf mobile devices is the restricted access to **Performance Monitoring Counters** (**PMCs**). Although PMCs, which are hardware registers that track events such as cache misses, branch mispredictions, and executed instructions, are vital for identifying performance bottlenecks, their availability on commercial mobile hardware is often limited or entirely inaccessible. This lack of visibility makes it challenging to accurately measure system performance or identify workloads using traditional hardware-centric metrics.

Shye *et al.* [41] proposed an **individualised DVFS** (**iDVFS**) approach that leverages user-specific hardware performance data to optimise power efficiency while maintaining user satisfaction. Similarly, Halpern *et al.* [21] analysed the relationship between system-level parameters and user satisfaction. However, unlike the offline analysis of user experience using pre-recorded videos in [21], our approach dynamically adapts to live user interactions; this enables real-time, simultaneous adjustments to both DVFS and screen brightness. While [41] correlated system parameters with user satisfaction, our framework goes further by proactively adjusting these parameters to achieve a balance between QoE and battery life.

Adaptive display brightness scaling has also been explored [32, 35, 39, 43, 45] to improve user experience and reduce energy consumption. For instance, Qian *et al.* [35] proposed a mobile video streaming method based on brightness scaling that optimises energy savings by considering user perception and context, such as device motion. In their work, user context is categorised into four distinct conditions: stationary, normal, in motion, and in a vehicle. Similarly, Mohomed *et al.* [32] considered device type, screen size, network bandwidth, and user location as contextual characteristics affecting web content. While Yan *et al.* [45] and Sun *et al.* [43] utilised ambient light as the primary context for mobile videos, Schuchhardt *et al.* [39] incorporated additional factors, including the sun's angle and on-screen time, to calculate optimal screen brightness. Despite these advancements, these approaches are often tailored for specific applications and rely solely on the display subsystem. Moreover, by neglecting user-defined temporal constraints (e.g., a specific battery life target), such systems risk over-allocating resources early in the discharge cycle. This lack of foresight necessitates aggressive throttling or premature shutdown later in the cycle, resulting in a significant and avoidable degradation of QoE.

Recent advancements have further expanded the display power-saving landscape through sophisticated colour modulation and pixel-level mapping techniques. These include sub-pixel rendering and chromaticity-aware scaling to reduce OLED power consumption without degrading visual fidelity [7, 17]. Other approaches have utilised content-aware refresh rate scaling and adaptive colour-space transformations to exploit human perceptual limitations [6, 30]. While these pixel-centric methods provide substantial display-side efficiency, they operate independently of the computational workload; consequently, they fail to adapt to shifting user behaviours or diverse application demands in dynamic runtime environments.

Our work builds upon these foundations by introducing a system-level mediation framework. Unlike isolated display or DVFS techniques, our approach establishes a joint QoE-energy objective function that treats display brightness and DVFS as a coupled vector. This enables the manager to proactively balance the experience budget across both subsystems; for example, the framework may permit a calculated reduction in display luminance to preserve the computational frequency required for smooth rendering in high-demand scenarios. By integrating these previously disparate constraints into a unified online lookup strategy, we achieve a coordinated resource allocation

that isolated mapping techniques cannot facilitate. While our prior works [23, 24] made progress by investigating the relationship between user perception and display-related factors in isolation, this article extends that foundation to a comprehensive system-wide perspective. Specifically, we introduce an integrated approach that jointly manages DVFS and display brightness, supported by a machine learning–based workload clustering technique for real-time classification and context identification. By combining workload-aware adaptation with cross-resource management, our framework enables maximising QoE across diverse application scenarios while adhering to the user's expected battery life.

3 QoE-Resource Relationship Analysis

Since user experience is subjective and arises from an individual's assessment of the fulfilment of their expectations with an application, service, or system used, in the specific context that they are currently in, it is paramount to collect such data and ratings from users for proper evaluation. To achieve this, workload characteristics, context, and user experience for various workloads and contexts need to be collected and evaluated. In this section, we investigate how user-perceived **Quality of Experience** (**QoE**) relates to underlying system resource behaviour. We first describe the design of our user perception study, then examine system resource responses under varying conditions, and finally identify the QoE–resource relationship relevant to DVFS and brightness scaling.

3.1 User Perception Study Setup

The objective of the user study is to evaluate the influence of varying processor frequency levels on user experience across diverse applications. Moreover, it aims at substantiating that changes in frequency within specific workloads influence the user's overall experience. To achieve this, it is necessary for participants to engage with various mobile applications while operating under varying DVFS configurations. Participants were informed that the experiment would assess seven distinct configurations (details in Section 3.1.4), specifically related to DVFS, without prior knowledge of which configuration was being tested. They were asked to provide ratings of their perceived experience with the applications under each configuration, presented in a randomised order. After each interaction with an application, users were given the opportunity to evaluate their experience, and these ratings were recorded alongside additional metrics, such as frame rate, power and frequency, collected from the device. The collected data is then analysed to derive the observations and insights presented in Section 3.2, and the QoE model detailed in Section 3.3.

3.1.1 Test Device. All our measurements and observations are based on real-life usage on mobile device. We used a Google Pixel 3 smartphone running Android 11. This mobile device leverages the ARM big.LITTLE architecture [18], a form of heterogeneous multi-processing (HMP) that combines two contrasting core types: powerful "big" cores for demanding tasks and energy-efficient "LITTLE" cores for everyday use. The architecture is particularly advantageous for battery-powered devices like smartphones and tablets, where balancing performance and power consumption is critical. To ensure that participants developed a reasonable and practical understanding of the battery discharge behaviour on the test device, we incorporated a structured familiarisation period. This period lasted approximately 20–30 minutes, during which participants engaged in everyday smartphone activities such as video playback and web browsing under typical usage conditions. These tasks were selected to reflect common usage patterns and to develop a sense of the phone's battery characteristics over the timeframe. To avoid introducing technical bias, instructions were intentionally framed using relatable and non-technical language.

3.1.2 Users. A total of 42 participants aged 18 and above were recruited for the subjective study. The sample comprised a mix of university staff, students, and acquaintances. To ensure a comprehensive evaluation of the proposed approach and model, we aimed for diversity in participant characteristics. This recruitment strategy enhances the generalisability of the findings across a broad range of user groups. The sample size of 42 aligns with established practices in relevant literature (e.g., [40]) and meets the subjective ITU-R standard for research in this field [26]. Moreover, precautions were taken to exclude participants directly involved in the system's development to mitigate potential biases stemming from detailed system knowledge. Ethical approval was granted by the University's ethics committee with reference number 78934.

3.1.3 Workloads. We evaluated six mobile workloads belonging to the top-10 % of the most popular mobile workload categories [2, 4]: YouTube, Chrome, Google Maps, Gmail, Genshin Impact (Genshin), and PDF reader. These workloads have a high user engagement rate, with the average user interacting with at least one of them within two days.

3.1.4 Task. The study comprises a series of interactions with the applications. The user engages with the applications throughout the entire duration of the activity. Meanwhile, DVFS configurations are periodically changed at random, cycling through the supported frequencies, ranging, for example, from 0.82 GHz to 2.64 GHz. When an application is suspended or closed at a designated DVFS configuration, a rating window appears requesting the user's evaluation of their experience with the application at the set configuration. Subsequently, the DVFS configuration is modified, and the process repeats under the updated DVFS configuration. This iterative procedure is executed until all expected DVFS configurations have been evaluated. The aforementioned procedure is iteratively executed for every assessed application. Similarly, the same experiments were conducted while changing the maximum frequencies of the big and LITTLE CPU cores. We utilised the OS built-in utility framework to report memory frequency information, including the current operating frequency. Real-time phone battery power consumption was also measured by accessing the **Battery Management System** (**BMS**) data via the **Android Debug Bridge** (**ADB**) interface and logging the metrics for analysis.

3.1.5 Quality Rating. To ensure a clear and consistent evaluation framework, we adopted a 3-point subjective quality rating scale, commonly employed in QoE studies (e.g., [24]), categorised as follows:

— Ideal (3): Denotes an impeccable and seamless user experience that either met or exceeded expectations.
— Acceptable (2): Represents a satisfactory or tolerable user experience, acknowledging potential limitations.
— Unacceptable (1): Indicates a highly unsatisfactory experience characterised by persistent glitches, unresponsiveness, or significant deviations from user expectations.

To enhance granularity, we introduced intermediate ratings of 2.5 (mostly ideal), reflecting an experience that is nearly ideal with only negligible imperfections, and 1.5 (barely acceptable), representing an experience that is marginally better than unacceptable but still falls short of being satisfactory. Recognising the inherent subjectivity in QoE and the varying expectations across different application categories, we instructed users to assess their experience relative to the specific context and use-case of each application. For example, in gaming, a score of 2.5 reflected expected responsiveness, while in web browsing, it corresponded to smoothness and page load latency. This approach prioritised capturing user-perceived quality within the specific context of each application, rather than enforcing a uniform, cross-application standard. By aligning the rating

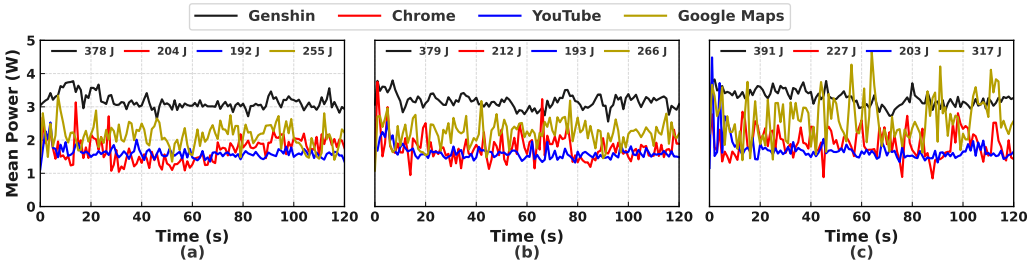

Fig. 1. Power consumption patterns exhibited by four mobile workloads executing under default scheduling and DVFS governor, with the *big* CPU core frequency capped at a) 0.82 GHz, (b) 1.68 GHz and (c) 2.64 GHz. Estimated energy consumption over a 120-second interval is annotated in each figure.

system with the subjective nature of user experience, we facilitate more meaningful, application- and context-specific insights.

3.2 Observed System Resource Behaviour

This section presents empirical observations of power consumption, memory, and GPU utilisation across our selected workloads, as gathered during the user study detailed in Section 3.1. Crucially, it then analyses these objective resource behaviours alongside subjective user-perceived QoE, revealing critical insights into the limitations of conventional resource management and motivating our work.

3.2.1 Power, Memory, and GPU Resource Utilisation. Mobile workloads exhibit diverse resource consumption patterns, influenced by application-specific demands and user interaction behaviours. Understanding these characteristics across varying system configurations is crucial for effective resource management. To analyse resource behaviour under different performance states, we capped the frequency of the *big* CPU cores at three representative levels: 0.82 GHz (low), 1.68 GHz (medium), and 2.64 GHz (high). These frequency points, selected from a broader sweep of available CPU operating states, enabled a consistent evaluation of power, memory, and GPU utilisation across distinct performance tiers, serving as the basis for the observations presented below.

Power Consumption Patterns. Figure 1 presents the power consumption profiles of the evaluated mobile workloads at the three specified CPU frequency caps.

- **Genshin:** This graphics-intensive workload consistently exhibits the highest power consumption across all evaluated CPU frequencies. As the CPU frequency cap increases from 0.82 GHz (Figure 1(a)) to 2.64 GHz (Figure 1(c)), Genshin's average power consumption increases from 3.15 W to 3.26 W, reflecting its sustained and high resource demands. The total energy consumed over a 120-second interval also increases with frequency: $E_{0.82\,\text{GHz}} = 377.84\,\text{J}$, $E_{1.68\,\text{GHz}} = 378.97\,\text{J}$, and $E_{2.64\,\text{GHz}} = 390.94\,\text{J}$.
- **Chrome and Google Maps:** These workloads demonstrate "bursty" power consumption patterns, characterised by periodic short bursts of elevated power interspersed with periods of lower consumption. While the magnitude of these bursts tends to increase with higher CPU frequencies (e.g., peak power is higher at 2.64 GHz than at 0.82 GHz), their average power consumption remains considerably lower than Genshin. For instance, at 1.68 GHz, Chrome averages 1.76 W and Google Maps averages 2.21 W. Their corresponding energy consumption over 120 seconds is also lower: $E_{\text{Chrome}} = 211.80\,\text{J}$, $E_{\text{Maps}} = 265.66\,\text{J}$. The burst frequency and duration also showed some modulation with CPU frequency, indicating the CPU's role in processing these intermittent demands.

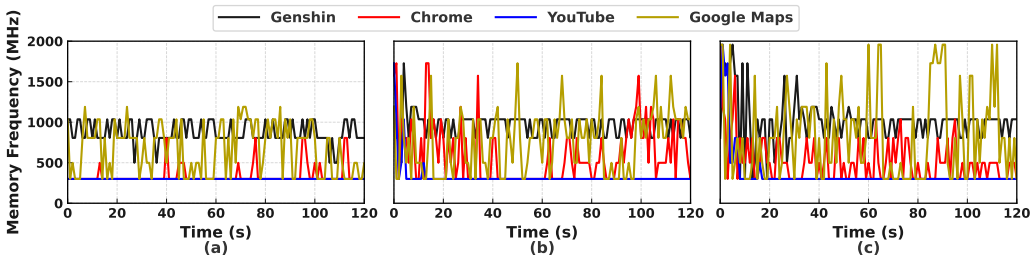

Fig. 2. Memory frequency patterns exhibited by four mobile workloads under default scheduling and DVFS governor settings, with the frequency of *big* CPU cores capped at (a) 0.82 GHz (b) 1.68 GHz and (c) 2.64 GHz.

— **YouTube:** This workload consistently displayed the lowest power consumption among the evaluated applications across all CPU frequency caps. Specifically, during the streaming of *Big Buck Bunny*, its power profile remained relatively stable and low, regardless of CPU frequency, averaging 1.61 W and consuming minimal energy over 120 seconds ($E_{\text{YouTube}} = 193.45$ J at 1.68 GHz). This reflects its inherently lower resource demands during continuous video streaming.

Memory Utilisation. Complementary to the power profiles, Figure 2 illustrates the observed median memory frequency for each workload across varying CPU frequency caps. *Genshin* consistently exhibits high memory bandwidth requirements, with median memory frequency increasing from 806 MHz (at 0.82 GHz) to 1036 MHz (at both 1.68 GHz and 2.64 GHz). This indicates the system's adaptive allocation of higher memory frequencies to support increased data processing at elevated CPU clock speeds, reflecting Genshin's sustained memory demands. Chrome and Google Maps reflect their bursty power consumption patterns in memory usage, exhibiting intermittent, high-intensity memory access. While peak activity generally increases with CPU frequency, their median memory frequency profiles diverge: Chrome's median memory frequency rose from 300 MHz (at 0.82 GHz) to 499 MHz (at higher frequencies), whereas Google Maps consistently maintained a median memory frequency of 806 MHz across all evaluated CPU frequency caps. *YouTube*, due to its comparatively lower processing demands, consistently exhibited the lowest memory activity. Its median memory frequency remained stable at 300 MHz across all CPU frequency caps, reflecting its inherently minimal and consistent demands on the memory subsystem.

GPU Utilisation. Figure 3 illustrates the GPU utilisation percentages for the four workloads under the three specified CPU frequency caps.

— **Genshin:** Consistent with its graphics-intensive nature, *Genshin* demonstrated a substantial and sustained high GPU utilisation, averaging 73.30 % (at 0.82 GHz) and 78.39 % (at 2.64 GHz). The quantifiable decrease in average GPU utilisation as the CPU frequency cap decreased suggests that at lower CPU frequencies, the CPU may become a bottleneck, thereby limiting data throughput to the GPU and restricting its peak utilisation.
— **Chrome and Google Maps:** These applications show lower overall GPU utilisation compared to Genshin, reflecting their bursty nature. Google Maps, in particular, exhibits highly intermittent GPU activity due to dynamic map rendering. For both applications, their GPU usage patterns showed varied and inconsistent CPU frequency dependency.
— **YouTube:** As expected, YouTube consistently registered the lowest GPU utilisation, with minimal variation observed across all CPU frequency caps, further confirming its low graphics demands.

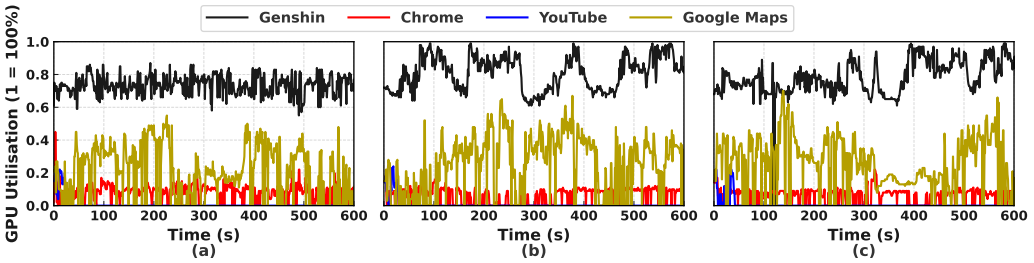

Fig. 3. GPU utilisation percentages for four mobile workloads under default scheduling and DVFS governor, with the frequency of the *big* CPU cores capped at (a) 0.82 GHz, (b) 1.68 GHz and (c) 2.64 GHz.

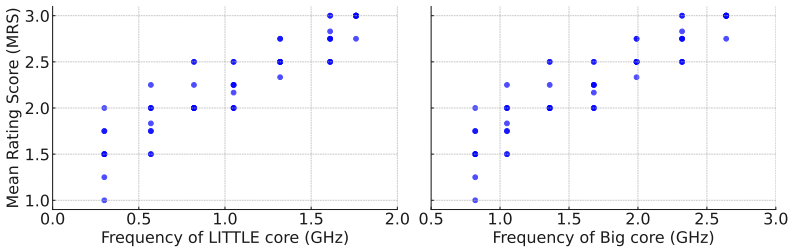

Fig. 4. Correlation between the MRS and cores frequencies (GHz) of (a) LITTLE and (b) big cluster

By analysing the correlation between **Mean Rating Score** (**MRS**) and CPU core frequencies (Figure 4), alongside observed system-level resource behaviour, we identify a critical QoE–resource challenge that highlights the complexity of dynamic resource management. While reducing CPU frequency lowers power consumption, it often results in a disproportionate decline in perceived QoE. For instance, with *Genshin*, operating at the lowest frequency (0.82 GHz) led to a degradation in MRS from approximately 3.0 (at 2.64 GHz) to below 1.5, a barely acceptable QoE level, due to aggressive throttling and CPU-induced GPU bottlenecks. Conversely, maximising CPU frequency does not always yield proportional improvements in QoE and can result in inefficient energy usage. For compute-intensive workloads such as *Genshin*, we observed diminishing returns in QoE beyond intermediate frequency settings. In contrast, for less demanding applications such as *YouTube*, DVFS had negligible impact on QoE, enabling energy savings. For bursty workloads, including *Chrome* and *Google Maps*, rapid and context-aware frequency adaptation is crucial to prevent QoE degradation. In essence, neither static resource provisioning nor battery-life–agnostic throttling consistently maximises QoE. This intricate interplay between objective resource metrics and subjective QoE highlights the need for, and motivates, the proposed adaptive resource management approach presented in Section 4.

3.3 Identifying the QoE–Resource Relationship

Existing models that establish a relationship between DVFS and performance have predominantly relied on objective performance metrics, such as application load time, obtained from the system. However, while these objective measures are correlated with DVFS settings, they do not adequately incorporate subjective user experiences when running various applications on real-world mobile devices. Moreover, the experimental setups, which frequently employ development platforms and mobile benchmarks, are not a true representation of real-world mobile operational scenarios [37], especially with regards to the critical factor of QoE. Given these inherent limitations, it becomes imperative to

Table 1. Predictive Performance and Overhead Comparison of Alternative QoE Models

Model Structure	Pearson r	MAE	RMSE	Execution Latency
Linear (Selected)	**0.918**	**0.151**	**0.201**	0.05 µs
Second-Order Polynomial	0.922	0.144	0.196	0.19 µs
Third-Order Polynomial	0.923	0.148	0.194	0.35 µs

involve users in order to gain insight into how changes in frequency affect their actual experiences, paving the way for improved user-centred DVFS strategies on real-world mobile devices. Consequently, bridging the gap between theoretical models and practical, user-focused management.

To determine the relationship between QoE and DVFS, we utilised the dataset from the user study (detailed in Section 3.1). This dataset consists of user ratings gathered from the execution of diverse workloads at varying **Processing Element** (**PE**) frequencies. The QoE model was developed using a systematic approach, starting with an exploratory analysis of the data in order to identify correlations, interdependencies, trends, and patterns between observed resource behaviours and the perceived QoE. The consistency of the subjective feedback was validated using the Inter-User **Coefficient of Variation** (*CV*). The mean Inter-User *CV* observed across the discrete frequency operational tiers was 0.096, indicating a high, statistically significant level of user agreement regarding frequency-induced processing performance.

The dataset was then partitioned into training and test sets for model development and evaluation. We applied five-fold cross-validation to ensure robust model performance. Multiple linear regression was used to fit the model, accounting for various independent variables and their collective impact on user ratings. While Figure 4 indicates a non-linear plateau in user perception at high frequencies, we adopt a linear approximation for the primary operational range (0.8 GHz to 2.64 GHz). While higher-order polynomial or non-linear modelling alternatives could theoretically capture localised perceptual variations, they introduce additional computational complexity and mathematical overhead. Because this estimation must execute within a tight, real-time control loop on resource-constrained mobile hardware, minimising runtime inference latency is paramount. Our first-order model maintains high predictive accuracy ($r = 0.918$, MAE $= 0.151$, RMSE $= 0.201$) across the primary operating frequencies without introducing non-trivial execution footprints. To provide a clear empirical justification for this architectural choice, Table 1 contrasts the predictive accuracy and computational latency of our selected linear model against higher-order polynomial alternatives evaluated directly on the user study dataset.

As demonstrated by the empirical metrics in Table 1, transitioning to non-linear spaces yields marginal improvements in predictive accuracy (under a 0.6 % variance gain in the Pearson correlation coefficient) while increasing the runtime microarchitectural inference latency by up to 7×. Because this objective function must execute within a high-frequency, real-time resource management loop, the minimal arithmetic footprint of the linear model is selected to guarantee a negligible system-level performance overhead. A thorough investigation into the exact cost-to-benefit boundaries of alternative non-linear modelling structures remains an open area of research, which we defer to future work. The established relationship between QoE and DVFS can be expressed as follows:

$$QoE_{DVFS} = \beta_0 + \sum_{k=1}^{m} \beta_k F_k. \tag{1}$$

where QoE_{DVFS} represents the user experience, which is equivalent to the users' **Mean Rating Score** (**MRS**) based on the DVFS settings. $\beta_0, \beta_1, \ldots, \beta_m$ are the coefficients or regression parameters associated with the frequency settings of the number of clusters for devices that support cluster-wise

DVFS. Since our device supports cluster-wise DVFS for the big and LITTLE clusters of the CPU, $m = 2$. In our regression analysis, the empirical global coefficients were calculated as $\beta_0 = 1.2997$, $\beta_1 = 0.9661$, and $\beta_2 = -0.0264$, calculated to minimise the sum of squared residuals. F_1 and F_2 denote the dynamic frequency caps for the LITTLE and big CPU clusters, respectively, which are used to predict *QoE*. The specific discrete frequency operational steps available on the evaluation device range from 0.3 GHz to 1.76 GHz for the LITTLE cluster, and 0.82 GHz to 2.64 GHz for the big cluster, as provided in Sections 3.1.4 and 5.1.

While the empirical model coefficients are calibrated to the evaluation platform, the underlying feature set (Table 2) is derived from standardised Android HAL APIs. This ensures that the framework's methodology remains portable across heterogeneous architectures, as the workload identification logic relies on universal system metrics rather than hardware-specific performance counters. Although QoE is influenced by a multi-dimensional array of internal variables (e.g., GPU and DRAM frequencies) and external contextual factors, this study prioritises CPU frequency as the primary determinant for the QoE-DVFS relationship. This focus is justified by the CPU's dominant role in driving application logic and resource orchestration in mobile workloads. Statistical evaluation confirms the model's global efficacy, achieving a Pearson Correlation Coefficient (r) of 0.918, an MAE of 0.151, and an RMSE of 0.201. While workload-specific regression coefficients could theoretically offer marginal gains in granularity, a universal coefficient approach was adopted to ensure negligible inference latency and prevent control-loop instability during concurrent application execution. Consequently, the model is deployed to determine the quantitative impact of each frequency setting on user experience in real-time.

Similarly, for the display subsystem, we incorporate a symmetrical visual counterpart, QoE_{DB}, synthesised from the established perception model proposed in [24]. This model implements a self-adaptive brightness scaling mechanism informed by ambient light sensor readings and real-time display content analysis. This integrated ambient- and content-aware approach enables fine-grained brightness control, providing potential for power savings without perceptual compromise, especially in dynamic lighting conditions. The primary novelty of this work lies in the joint optimisation and online mediation of these display constraints alongside the workload-aware DVFS policy.

To model the overall QoE, we represent it as a weighted dot product of the perceived contributions from the DVFS and DB subsystems. This formulation captures the distinct impact of each subsystem, as changes in execution frequency and display configurations influence user perception through separate modalities. Far from implying operational independence, this joint formulation is necessary precisely because these distinct mechanisms compete for the same finite energy resource. By evaluating both subsystems simultaneously within a unified space, the framework resolves their coupled tradeoffs and scales each component according to its relative importance. The resulting joint QoE objective function is expressed in Equation (2):

$$QoE_{(DVFS,DB)} = \alpha^\top \cdot \mathbf{QoE}, \tag{2}$$

where $\mathbf{QoE} = \begin{bmatrix} QoE_{(DVFS)} & QoE_{(DB)} \end{bmatrix}^\top$ is the feature vector comprising the perceived QoE for DVFS and DB. Specifically, $QoE_{(DB)}$ represents the perceived display *QoE*, modelled as a piecewise function of brightness φ, ambient luminance L_{amb}, and content luminance L_C adapted from the methodology established in [24]. To account for the saturation of human visual perception in high-glare environments, $QoE_{(DB)}$ is defined as $(\alpha+\beta L_C+\gamma L_C^2)+(\delta L_C+\epsilon)\varphi+\rho\varphi^2$ for $L_{amb} < 1000$ lux, and follows a logistic form $(1 + \exp^{-(\sigma+\lambda L_C+\eta\varphi)})^{-1}$ for $L_{amb} \geq 1000$ lux [24]. The ambient luminance is obtained from the device's ambient light sensor, while $QoE_{(DVFS)}$ follows the derivation in Equation (1). The weight vector $\alpha = [\alpha, (1-\alpha)]^\top$ defines the relative importance of DVFS and DB with respect to the overall QoE (Equation (2)), which serves as the Joint QoE Objective Function used

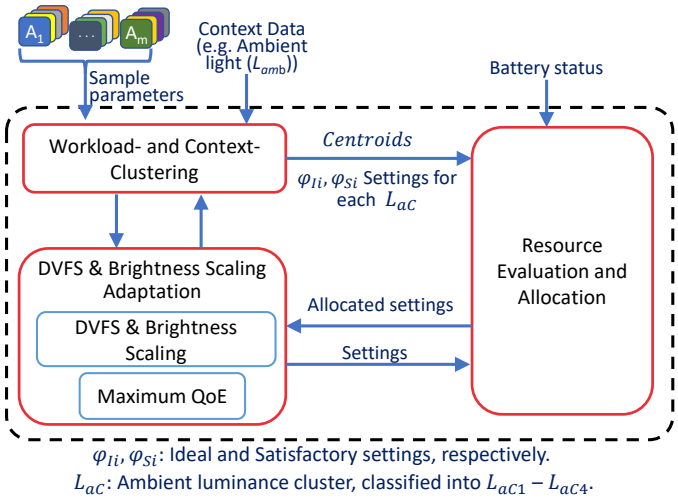

Fig. 5. System overview of the proposed adaptive DVFS and display brightness scaling approach

to link these disparate subsystems. By utilising the weighting parameter α, the framework ensures that computational responsiveness (QoE_{DVFS}) and display visibility (QoE_{DB}) compete for the same finite energy resource, thereby coupling the two quality metrics within a unified optimisation goal.

4 Proposed Adaptive DVFS and Display Brightness Scaling Approach

This section introduces the proposed adaptive DVFS and **Display Brightness** (**DB**) scaling approach, designed to maximise QoE under varying workload and environmental conditions. Building on the insights gained from the user perception study and the QoE–resource relationship established in Section 3, the approach enables real-time system adaptation to both workload characteristics and contextual dynamics. Figure 5 presents the system architecture, which comprises three main components: workload- and context-aware clustering, resource evaluation and allocation, and an online DVFS and brightness scaling manager. The clustering module (Section 4.1) periodically samples runtime context parameters to classify workloads and determine appropriate DB levels (specifically, ideal (φ_{I_i}) and satisfactory (φ_{S_i}) settings). Based on these clusters and a target battery life constraint, the resource evaluation module (Section 4.2) continuously assesses current system demands and allocates configurations that maximise QoE while meeting the expected battery life. The online manager (Section 4.3) then applies the allocated settings and continuously monitors both system and environmental states to adapt DVFS and DB levels in real time, responding dynamically to workload fluctuations and ambient conditions. The following subsections detail each component of the proposed approach.

4.1 Workload- and Context-aware Clustering

The objective of workload- and context-aware clustering is to identify a set of clusters that can be employed at runtime for both workloads and context (ambient light). Effectively identifying the appropriate cluster for a given workload and context enables the implementation of proactive measures that maximise the QoE. These are the steps involved.

4.1.1 Data Collection. The proposed approach requires a number of parameters to be collected during the execution of mobile applications in order to uniquely characterise workload behaviour for clustering. As shown in Figure 5, we begin by executing various applications (A_1 to A_m) and

Table 2. Sample Runtime Parameters for the Proposed Workload- and Context-Aware Clustering

S/N	Parameter	S/N	Parameter
1	Temperature of the LITTLE cluster cores	8	Frequency of the big cluster
2	Temperature of the big cluster cores	9	Frequency of the GPU cores
3	Temperature of the GPU cores	10	Percentage of GPU utilisation
4	Current	11	Frame Rate (Frames Per Second)
5	Voltage	12	Memory frequency
6	Power	13	Ambient light
7	Frequency of the LITTLE cluster		

collecting a variety of parameters. Table 2 outlines the various parameters used in this investigation, including CPU and GPU temperatures, current, voltage, power, CPU and GPU frequencies, GPU utilisation percentage, and memory frequency. Periodically, these parameters are gathered and used to infer the workload cluster based on their intrinsically distinct patterns, specifically every 200 ms. These parameters are utilised because PMCs (easily accessible from the performance monitoring unit of many development platforms) are not typically accessible on commercial devices. In addition, the selected parameters have demonstrated relevance in prior work for capturing workload dynamics. To ensure both the sufficiency and relevance of this feature set, we conducted a feature importance analysis using the Gini criterion alongside cluster validation (e.g., silhouette score). A mean Silhouette score of 0.72 confirms high cluster cohesion and separation. This analysis identified CPU utilisation and memory frequency as the dominant drivers of cluster separation, collectively accounting for over 60% of the model's predictive power. Thermal and voltage metrics, whilst secondary, provided the necessary contextual variance to distinguish between different states, such as thermal-throttling and high-performance execution modes.

4.1.2 Cluster Determination. Our system employs an efficient classification mechanism to dynamically group workloads based on resource requirements. This approach identifies workload and context clusters at runtime, enabling tailored DVFS and display brightness settings to maximise QoE while meeting the expected battery life.

(i) Workload Clustering and Labelling: We employ K-Means clustering to classify workloads based on their runtime characteristics (e.g., CPU, memory, and GPU usage). K-Means is selected for its computational efficiency and ability to form coherent clusters based on feature similarity, making it well-suited for real-time classification. To ensure the clusters are semantically meaningful for system adaptation, we validate them through post hoc analysis of resource usage patterns within each cluster. Each cluster is then labelled according to its dominant behavioural traits. For example, workloads with sustained high CPU and memory demands are labelled as *High-Demand Applications*. These labels serve to abstract workload types and guide the system in allocating appropriate device settings. To maintain system reliability during edge-case scenarios, we integrate a stability and fallback framework. For new or unseen workloads that do not align with established training centroids, the system utilises a Euclidean distance threshold; if a new application's feature vector falls beyond a defined radius of existing centroids, the manager defaults to a *Conservative-Performance* profile. Furthermore, a 500 ms temporal hysteresis filter is applied to prevent rapid frequency switching during intra-application phase changes, ensuring that a cluster transition is only triggered once the workload shift demonstrates temporal stability.

(ii) Context Clustering and Integration: In addition to workload clustering, our system integrates ambient context clustering to inform adaptive display control, following the approach in [23, 24]. Specifically, we cluster ambient luminance readings (in lux) from the device's light sensor into four

predefined ranges: L_{aC1} [0, 100), L_{aC2} [100, 400), L_{aC3} [400, 1000), and L_{aC4} [1000, ∞). These ranges were selected to capture distinct lighting conditions, ranging from dark indoor environments to bright outdoor settings, based on thresholds commonly used in mobile display adaptation literature.

These ambient clusters, combined with workload centroids, form a unified input for the online DVFS and DB scaling mechanism. K-means clustering is executed only when a significant workload shift is detected (specifically when the feature vector deviates beyond the defined Euclidean threshold), with results cached and reused within the same session to minimise recomputation overhead.

4.2 Resource Evaluation and Allocation

To allocate appropriate settings that deliver acceptable QoE (i.e., acceptability (A), defined by $A \in [A_S, A_I]$, where A_S and A_I represent satisfactory and ideal thresholds, respectively) and meet the expected battery life, the resource evaluation and allocation process balances the power-to-QoE tradeoff. This balance is achieved by comparing the device's remaining battery energy (E_{rem}) with the expected energy demand and selecting configurations for DVFS and display brightness (DB) that align with current workload and context clusters to meet user expectations and battery life requirements. The expected energy demand (E_{req}), which represents the total energy required to meet the user-specified lifetime (T_{req}), is estimated using an energy model that incorporates user usage patterns across various workload and context clusters, along with system configurations, such as DVFS and DB. It is expressed as:

$$E_{\text{req}}(\theta) = \sum_{i=1}^{m} \hat{P}_{\theta_i}(W_i, C_i) \cdot T_{\text{req}}, \tag{3}$$

where $\hat{P}_{\theta_i}(W_i, C_i)$ represents the estimated power consumption under configuration θ_i for workload cluster W_i and context cluster C_i, aggregated over m workload-context clusters. The term T_{req} denotes the user-specified lifetime, also referred to as the expected battery life, which corresponds to the operational duration or number of hours intended for device operation as defined by the user. The user-specified lifetime is communicated as a high-level temporal constraint (e.g., a "4-hour" target) provided through a system-level UI component. The resource manager then translates this discrete input into a target energy requirement (E_{req}) for the remaining discharge cycle. To ensure the fidelity of this model, we characterised its predictive accuracy by comparing estimated demand against actual consumption recorded during the user study, yielding a **Mean Absolute Percentage Error** (**MAPE**) of 6.4%. The power estimates, $\hat{P}_{\theta_i}$, are derived from a comprehensive pre-profiled power-state table. To account for intra-application variability, the model utilises the mean power draw captured over a 500 ms sliding window as its representative value. This approach filters transient power spikes whilst ensuring the joint *QoE* objective function remains grounded in sustained workload characteristics. Furthermore, to address the non-linear discharge characteristics of **Lithium-ion** (**Li-ion**) batteries, our framework operates primarily in the charge domain. By utilising Coulomb-counting metrics (mAh) provided by the Android **Hardware Abstraction Layer** (**HAL**) rather than raw voltage, the model significantly mitigates errors arising from **State-of-Charge** (**SOC**) dependent voltage drops.

This supervisory configuration capping provides a fundamental operational advantage over standard existing governors, such as `schedutil` or `ondemand`. Default governors react exclusively to localised, high-frequency utilisation spikes over brief millisecond intervals, remaining entirely agnostic to global system variables such as remaining battery energy (E_{rem}), display power consumption, or user-expected execution duration (T_{req}). Consequently, standard governors are incapable of enforcing a sustained energy budget across a prolonged application session. While a default

governor paired with a static frequency ceiling can limit peak power draw, it lacks the context-awareness required to dynamically scale performance boundaries when user interaction patterns or remaining battery capacities shift. The novelty of the proposed framework lies in its long-term, predictive energy budget computation, which systematically updates the operational boundaries of standard governors to ensure that short-term, reactive scheduling remains strictly compliant with long-term user battery life objectives.

To satisfy both acceptable QoE (in terms of workload performance and DB) and expected battery life constraints, the system evaluates the complete configuration space $\mathcal{C}$, identifying a subset of feasible configurations $\mathcal{F}_i \subseteq \mathcal{C}$ that satisfies both criteria for each workload and context pair:

$$\mathcal{F}_i = \{\theta \in \mathcal{C} \mid Q(\theta) \geq A_i\}, \tag{4}$$

where $Q(\theta)$ is the perceived quality of configuration θ, derived from user feedback collected through the evaluation of system performance and display characteristics under various DVFS and DB settings. The threshold A_i denotes the minimum acceptable QoE for the ith workload and context combination. From the feasible configuration set $\mathcal{F}_i$, the appropriate configuration θ^* is determined by minimising the absolute difference between E_{rem} and $E_{\text{req}}(\theta)$, ensuring a balance between user expectations and expected battery life:

$$\theta^* = \arg\min_{\theta \in \mathcal{F}_i} |E_{\text{rem}} - E_{\text{req}}(\theta)|. \tag{5}$$

To mitigate the impact of unpredictable user behaviour, the framework employs an iterative recalculation policy where the energy budget is re-evaluated every 30 s. This allows the system to dynamically adjust the feasible configuration set $\mathcal{F}_i$ to compensate for prior inaccuracies or shifts in user activity. This periodic update ensures that the overall energy target is maintained over the user-expected time, effectively anchoring the optimisation within a practical, closed-loop control cycle.

4.3 Online DVFS and Brightness Scaling

Considering that users typically switch from one workload to another and from one context to another many times within a single battery discharge cycle (typical usage scenario), it is crucial to select the appropriate frequency and brightness level for each workload and context if the user experience is to be maximised and the expected battery life is to be met. To address this, an approach similar to that described in [25], which assigns DVFS settings based on the ratio of the remaining energy to the predicted energy, is extended to include display brightness scaling as well as processes that account for the various types of workloads that users may engage with.

First, similar to approaches described in [25] and [23], we determine the frequency settings and ideal brightness level at which the QoE reaches its maximum. This point, referred to as *Maximum QoE*, represents the minimum threshold at which further increasing the frequency or brightness no longer improves or affects the QoE for each workload and context, as the user is already satisfied. Once these values are identified and allocated by the resource evaluation and allocation process, this means that the energy budget is computed to be sufficient to meet the expected battery life. Consequently, the online DVFS and brightness scaling agent sets the maximum operating frequencies of the respective processing elements to this value, specifically in the case of DVFS. This allows flexibility in adjusting the core frequencies, ranging from the capped values down to the lowest allowable frequency. This adaptability is crucial in accommodating the various patterns observed in mobile applications, such as waiting for user input or output, handling busy activities, and so on, while maintaining the quality required. Similarly, the brightness value is set to an ideal level for the context and changes only when the context or energy budget changes.

To ensure transparent integration with standard hardware architectures, the online manager is designed to operate as a privileged userspace background process that interfaces with the Linux kernel entirely through the sysfs virtual filesystem. Rather than directly overriding kernel-level task distribution or enforcing custom migration heuristics, the framework establishes dynamic performance capacity ceilings by modifying the `scaling_max_freq` attributes across the big and LITTLE core clusters. Under this architecture, the native Android **Energy-Aware Scheduler** (**EAS**) retains full execution authority over individual thread placement and load balancing. By continuously scaling these cluster ceilings based on the global energy budget, our manager systematically reshapes the capacity headroom and energy-cost landscape, allowing native EAS to implement scheduler-congruent, energy-efficient placement paths without incurring direct Userspace-to-Kernel migration overheads.

Unlike standard kernel governors (such as `schedutil` or `ondemand`) that operate reactively based on instantaneous short-term CPU utilisation spikes, our framework introduces a proactive, predictive control layer driven by a global energy budget. Standard governors possess no awareness of cross-subsystem energy tradeoffs, screen brightness power draw, or total remaining application execution duration; consequently, capping a standard governor via static thermal or power thresholds fails to guarantee that a specific battery lifetime constraint (T_{req}) will be met. Furthermore, our framework reshapes the EAS placement landscape by leveraging the non-linear power-performance curves of modern heterogeneous SoCs. In scenarios characterised by low-intensity background or interactive threads, executing workloads on the LITTLE cluster at a higher frequency efficiency point should regularly outperform throttling the big cluster under aggressive DVFS, as the big cluster incurs a higher static power floor even at lower frequencies. The native EAS is expected to automatically detect these capacity headroom shifts induced by our Userspace-to-Kernel dynamic performance caps. Consequently, when our manager constrains the big cluster's performance ceiling, the EAS internal energy model should naturally determine that migrating certain threads to the LITTLE cluster yields a more optimal energy-to-performance ratio, safely absorbing the negligible kernel-level thread migration overhead (typically under 10 µs) to achieve the targeted global energy predictability.

When the estimated energy budget is insufficient, the DVFS and brightness scaling manager is employed to dynamically adjust the CPU frequency and screen brightness levels based on the real-time energy ratio (i.e., remaining (E_{rem}) to the required energy (E_{req})) and contextual workload clusters. For example, if the energy ratio is around 85 %, frequency scaling is selectively applied according to workload characteristics, while brightness is adjusted to the ideal level, reflecting the fact that CPU scaling in standard applications has a minimal impact on perceived user experience. Conversely, for compute-intensive workloads, simultaneous DVFS and brightness adjustments are employed to minimise the difference between the energy gap (i.e., E_{req} and E_{rem}). At lower energy ratios, a conservative strategy is adopted by exploiting the QoE space defined between the maximum and satisfactory thresholds for both tuning knobs. These thresholds guide the tradeoff between power savings and QoE, ensuring that adaptations remain acceptable to users while ensuring that the expected battery life is met.

To support this adaptation strategy, our system continuously monitors a suite of performance and power metrics using on-device profiling tools integrated within the SoC. This includes CPU and GPU utilisation, memory frequency, power draw, and temperature readings from both the CPU and GPU thermal sensors. These monitoring capabilities provide the raw data needed for real-time workload classification and energy estimation. Table 2 summarises the monitored hardware and software parameters, which serve as input to the adaptation controller. While our control loop primarily controls CPU DVFS and screen brightness, it benefits from the broader system insight. In particular, by incorporating GPU utilisation, temperature and memory frequency, our

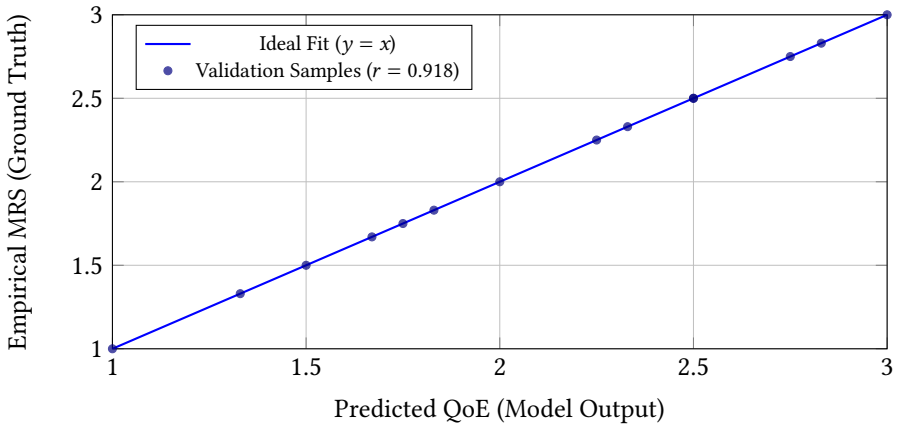

Fig. 6. Correlation analysis between the framework's Predicted QoE and the empirical Mean Rating Scores (MRS) from the user study dataset, confirming an exact Pearson correlation coefficient of $r = 0.918$.

classifier can more effectively differentiate among diverse workload behaviours. These features help refine context identification and ensure adaptation decisions are responsive to workload shifts across different system components. Our approach also leverages the interdependencies within modern mobile SoCs. Adjustments to CPU frequency, for instance, can indirectly influence GPU performance and memory access patterns, particularly in big.LITTLE architectures where shared thermal and power domains exist. This systemic interaction is inherently captured through the monitored parameters, allowing our approach to maintain a lightweight control strategy while benefiting from comprehensive observability. Although frequency changes are constrained by thermal considerations, the system's on-device thermal profiling facilitates DVFS decisions that respect safe operational bounds, while the processor's native throttling provides a final hardware-level safety net. The processor's built-in thermal throttling acts as a safety net, but our adaptive mechanism proactively avoids exceeding thermal thresholds by factoring in both current and projected thermal states.

To evaluate the framework's efficacy, we conducted an online cross-validation analysis comparing real-time resource allocations against the empirical MRS models. Figure 6 illustrates the correlation between the framework's internal QoE predictions and the ground-truth user ratings collected across diverse operational workloads. The framework achieves high alignment with experimental realities, demonstrating an MAE of 0.151 on the 3-point scale with a Pearson Correlation Coefficient (r) of 0.918. This high degree of linear fit confirms that the joint QoE formulation serves as a highly accurate proxy for human perceptual tradeoffs across the platform's primary operating limits.

To evaluate the structural sensitivity of the resource manager to its internal design weights, we analysed the system's operational decision path as the tuning parameter α scales across its primary spectrum from 0.3 to 0.7 under a constrained energy budget. This parameter dictates the tradeoff priority within our joint formulation, directly determining how the coupled QoE goals map to physical resource adjustments. At the lower bounds ($\alpha = 0.3$), the framework prioritises visual fidelity, treating display illumination as the dominant driver of user experience. Consequently, under a constrained energy budget, the manager proactively restricts the performance capacity headroom of the processing element clusters to reserve the device's remaining charge domain for maximum display brightness. This is executed by dynamically lowering the maximum allowable frequency boundaries via `sysfs`. As α increases to 0.4, this tight computational constraint relaxes, allowing

the framework to trade minor visual dimming for increased processing headroom. At the selected baseline ($\alpha = 0.5$), the joint optimisation goal achieves a balance, effectively preventing either the processing or the display subsystem from monopolising the shared energy budget. Moving toward the upper bounds ($\alpha = 0.6$ and $\alpha = 0.7$), the optimisation landscape flips to privilege computational responsiveness. In this high-α regime, the system opens the performance ceilings, allowing the processing elements to burst freely toward their peak hardware limits. Simultaneously, the manager executes aggressive display brightness throttling to ensure the target battery life constraint (T_{req}) is satisfied. This predictable, monotonic shift in hardware allocation behaviour demonstrates that the joint objective function acts as a robust, well-behaved proxy for user experience across varying environmental and workload conditions.

5 Experimental Evaluation and Analysis

5.1 Implementation and Evaluation Setup

To evaluate the proposed adaptive DVFS and display brightness scaling approach, a series of experiments were conducted on a Google Pixel 3 mobile device running Android 11 (Red Velvet Cake) with kernel version 4.9.248. This mobile device is equipped with a Qualcomm Snapdragon 845 SoC [18], which integrates CPU clusters based on ARM's big.LITTLE multi-core architecture. The scaling frequency of its big and LITTLE CPU clusters can be fine-tuned within the ranges of 0.82 GHz to 2.64 GHz and 0.30 GHz to 1.76 GHz, respectively [36]. While evaluated on the Snapdragon 845, the framework's reliance on standardised Android HAL metrics ensures portability to contemporary tri-cluster and quad-cluster SoCs. To ensure experimental fairness and workload determinism, all evaluations were executed using a workload automation framework. This provided identical interaction sequences, including fixed scroll velocities and navigation timings, across all tested configurations. Consequently, observed power variances are strictly a function of the resource manager's scaling decisions rather than user-induced noise.

Given that real-life smartphone usage varies across different ambient contexts and applications, our evaluation scenario was carefully designed to reflect these diverse activities. We incorporated a range of ambient lighting conditions similar to those described in [23], encompassing dim light $[0-100)$ lx, indoor normal light $[100-400)$ lx, bright light $[400-1000)$ lx, and extremely bright light $[1000 - +\infty)$ lx. In addition, to account for the dynamic nature of user behaviour, we incorporated scenarios where users transition between different applications, such as checking email or engaging with social media, ensuring a comprehensive assessment of real-world usage patterns (further details in Sections 5.1.1 and 5.1.2). We evaluated a selection of real-world mobile workloads with diverse requirements and characteristics, such as YouTube, Chrome, Google Maps, Google Mail (Gmail), PDF reader, and Genshin [1]. Our selection criteria considered not only the popularity and widespread use of these applications but also their diverse functionalities across entertainment, productivity, gaming, education, and travel categories [2, 4].

For a holistic evaluation within our framework, we measured the total system energy consumption during typical usage scenarios, inherently accounting for the impact of various system components, including data transmission over 5G/4G LTE and Wi-Fi networks. All reported results incorporate the incurred overhead of our approach and it remains below 2 % of total battery energy consumption. The one-time offline centroid training completes within 4 s, while the runtime inference latency remains below 2 ms, ensuring real-time responsiveness to workload shifts.

5.1.1 Experimental Scenarios. To ensure our evaluation is based on real-world mobile usage patterns, we analysed typical usage scenarios from both the participants in Section 3 and new individual users, considering the proportion of daily time spent on each mobile workload. We determined this by using the daily app usage summary, a feature commonly available on modern smartphones.

By aggregating this data from the broad range of users, we identified three scenarios that reflect user interactions with their devices. *Scenario 1* consists of a mix of streaming and bursty workloads. Streaming workloads such as YouTube, TikTok, and Netflix constitute 50 % of the scenario, while bursty workloads, including Google Maps, Gmail, and web browsers (e.g., Chrome), constitute the other 50 % in varying proportions. In *Scenario 2*, bursty workloads account for ≈ 72 % of the total workload. Within this category, web browsers (e.g., Chrome, Safari) account for 54 %, emails (e.g., Gmail) account for 8 %, and navigation workloads (e.g., Google Maps) account for 1 %. Streaming activities, such as YouTube, account for the remaining 28 % of the workload. For *Scenario 3*, gaming workloads (e.g., Genshin) account for about 46 % of the total, while streaming activities account for 32 %. The remaining 22 % of the workload is distributed among web browsers (e.g., Chrome), emails (e.g., Gmail), and navigation (e.g., Google Maps). To further show the adaptability of the various approaches to changes in user usage patterns or energy constraints, we evaluated the three scenarios with initial battery levels ranging from 100 % to 60 %, which are denoted as 100 %, 90 %, 80 %, 70 %, and 60 %.

5.1.2 Comparison Metrics. In our study, we compared the battery life and average QoE of our approach to state-of-the-art approaches. For QoE evaluation, we employed the QoE model represented by Equation (2), which effectively combines the QoE for the DVFS and display subsystem. Higher QoE while meeting expected battery life indicates more satisfied users. However, these objectives can be conflicting, as maximising one may result in a decrease in the other. To address this, we propose **QoE-Expected Time Index** (**QETI**) as a suitable metric for evaluation, where the **Expected Time Index** (**ETI**) is defined as the ratio between the actual usage time of a device by a user and the duration of time the user expects the device to last, i.e., the ETI quantifies the extent to which the device's actual usage time relates to the user's expected usage time (expected battery life). QETI reflects how effectively an approach maximises QoE while meeting the expected battery life by multiplying the QoE and expected time index, expressed as QETI = QoE × ETI. This index represents the approach's efficiency in terms of both QoE and time utilisation. A higher QETI score indicates a resource management approach that successfully maximises user QoE and expected time, resulting in an efficient and effective mobile experience. It is important to note that QoE and ETI are both normalised within the range of 0 and 1, making the maximum value for QETI also 1. In this article, we assigned equal weight to QoE and ETI in the evaluation. However, in future studies, different weights can be assigned to these two factors to tailor the resource management approach according to specific requirements and preferences.

5.1.3 Relevant State-of-the-art Approaches for Comparison. Although we aimed at evaluating our approach against a broad spectrum of existing literature, several practical constraints necessitated a targeted benchmarking strategy. Technical barriers, including hardware-specific dependencies on PMCs, proprietary control knobs, and platform-compatibility issues, preclude a direct empirical comparison with relevant specialised approaches, including learning-based methods. While these learning-based methods offer high adaptivity, their reliance on extensive offline and online training, combined with significant runtime computational overhead and the aforementioned hardware restrictions, poses challenges for immediate deployment on standardised production hardware. To address this, we provide a qualitative comparison in Table 3 that contrasts our proposed framework against representative **Reinforcement Learning** (**RL**) and **Deep Learning** (**DL**) paradigms established in contemporary literature, such as [15, 20, 27]. Our approach prioritises methodological portability and low-overhead adaptation: unlike RL-based methods that can suffer from cold-start performance degradation or unpredictable oscillations while the agent explores the state-space, our K-means clustering isolates workload shifts using standardised Android HAL data. By utilising a lightweight, cluster-based lookup strategy, our framework positions itself as a practical tradeoff between

Table 3. Qualitative Comparison of Resource Management Approaches

Feature	**Heuristic** (Rule-Based)	**Learning-Based** (RL/DL) [15, 20, 27]	**Proposed Approach**
Adaptation Mechanism	Reactive/static	Predictive/dynamic	Adaptive/dynamic
Adaptation Speed	Instant	Slow (convergence delay)	Immediate (cold-start)
Training Requirement	None	Heavy (offline/online)	Minimal
Computational Cost	Negligible	High (inference/update)	Low (lookup)
Data Requirements	Static thresholds	Reward function + history	Standard HAL metrics
Stability	High	Low (stochastic exploration)	High (discrete clusters)
Handled Objectives	Single	Multi (e.g., QoE, energy)	Multi (QoE, energy)

the simplicity of reactive governors and the heavy microarchitectural requirements of deep learning models, ensuring the manager's computational footprint remains negligible. Because we do not provide direct empirical tracking against these learning-based architectures, our assessment highlights a distinct design-space alternative focused on deployment efficiency rather than claiming performance superiority. Furthermore, to provide a rigorous empirical baseline against deployed systems, we evaluate our approach against the industry standard: the stock Android *Schedutil* governor. As the default DVFS policy on the Google Pixel 3, *Schedutil* inherently integrates **Energy Aware Scheduling** (**EAS**) [13] alongside standard adaptive brightness scaling. Additionally, we compare our approach against a modified implementation of **User- and Battery-Aware Resource management** (**UBAR**) [40], which accounts for both user QoE and long-term battery cycle life in its optimisation.

Schedutil with Static Brightness Scaling (DF-SBS): This approach is designed to optimise energy consumption and QoS by leveraging the diverse power and performance capabilities of the **System-on-Chip's** (**SoC**) multi-cores, as well as a brightness scaling that allows users to manually set the brightness level to either provide optimal brightness or maximise battery life. The term *static brightness scaling* refers to the non-automatic nature of this setting, in which the initial level of brightness remains constant until manually adjusted again. DF-SBS was chosen for comparison due to its widespread accessibility and popularity on most, if not all, smartphones. To perform the comparison, we selected a fixed brightness level of 50 %, which represents a typical range for indoor lighting conditions that is neither too high to drain more energy nor too low to compromise the QoE.

Schedutil with Automatic Brightness Scaling (DF-ABS): aims at achieving an energy-to-QoS balance by automatically determining the optimal DVFS and screen brightness based on ambient light conditions [19]. It utilises the brightness scaling option present in modern smartphones, that learns typical user lighting conditions and habits. By eliminating manual adjustments, DF-ABS increases battery life while maintaining user satisfaction. We chose to compare our approach to DF-ABS as it is prevalent in commercially flagship smartphones, where it has been shown to provide a balance between energy efficiency and user satisfaction by effectively managing the tradeoff between energy consumption and QoE.

Modified User- and Battery-Aware Resource Management (UBAR-M): We modified the *UBAR* approach from [40], which optimises **Quality of Experience** (**QoE**) and battery ageing by considering user preferences, plug-in/out patterns, and **State of Charge** (**SOC**) levels. The energy minimisation strategy in this approach involves reducing the performance reference (or performance target). When SOC is above 70 %, application performance requirements are set to the

Table 4. Evaluation of the Capability of Each Approach to Meet the Expected Battery Life Under the Three Scenarios Considered

Battery life expectation for Scenario 1					
Approach	**Met expected battery life at different initial battery life (IBL)?**				
	100% IBL	**90% IBL**	**80% IBL**	**70% IBL**	**60% IBL**
DF-SBS	Yes	Yes	No (−13 min)	No (−53 min)	No (−1h 33 min)
DF-ABS	Yes	No (−14 min)	No (−49 min)	No (−1h 33 min)	No (−2h)
UBAR-M	Yes	Yes	Yes	No (−37 min)	No (−1h)
Ours	Yes	Yes	Yes	Yes	No (−45 min)
Battery life expectation for Scenario 2					
Approach	**Met expected battery life at different initial battery life (IBL)?**				
	100% IBL	**90% IBL**	**80% IBL**	**70% IBL**	
DF-SBS	Yes	Yes	No (−34 min)	No (−1h 11 min)	
DF-ABS	No (−2 min)	No (−34 min)	No (−1h 7 min)	No (−1h 40 min)	
UBAR-M	Yes	Yes	No (−22 min)	No (−41 min)	
Ours	Yes	Yes	Yes	No (−7 min)	
Battery life expectation for Scenario 3					
Approach	**Met expected battery life at different initial battery life (IBL)?**				
	100% IBL	**90% IBL**			
DF-SBS	No (−38 min)	No (−1h 7 min)			
DF-ABS	No (−51 min)	No (−1h 19 min)			
UBAR-M	No (−43 min)	No (−47 min)			
Ours	Yes	No (−11 min)			

Note: Entries marked as "No (−*X*)" indicate that the battery was depleted *X* time earlier than expected.

highest level. When SOC drops below 70 %, the performance target is gradually relaxed, initially to 80 % and subsequently to 80 %. Instead of directly predicting plug-in times, the UBAR-M approach leverages expected battery life as input. To ensure a fair comparison, we integrate automatic brightness scaling, utilising the test device's native brightness adjustment. This enhancement accounts for both display power consumption and user battery life expectations, enabling a more comprehensive evaluation of energy-aware resource management.

5.2 Evaluation of Battery Life

Table 4 presents the evaluation of each approach's capability to meet the expected battery life for the three scenarios considered. In Scenario 1, all approaches meet the required battery life at an initial battery level of 100 %. However, as the initial battery level decreases, DF-ABS fails to meet the battery life requirements at 90 %, 80 %, 70 %, and 60 %, while DF-SBS fails at 80 %, 70 %, and 60 %, primarily due to their lack of expected battery life-aware adaptation. UBAR-M sustains battery life down to 80 % initial battery but fails at 70 % and 60 %. In contrast, our approach dynamically adjusts DVFS settings and brightness based on workload and context, achieving expected battery life down to 70 %. Compared to UBAR-M, DF-SBS and DF-ABS, our approach extends battery life by up to 12.63 % (0.62 h), 19.13 % (0.88 h) and 34.70 % (1.55 h), respectively. Similarly, in scenario 2, characterised by bursty workloads with intermittent resource requirements, our approach still meets battery life expectations up to 80 % initial battery level, demonstrating a 7.14 % (0.37 h), 11.50 %

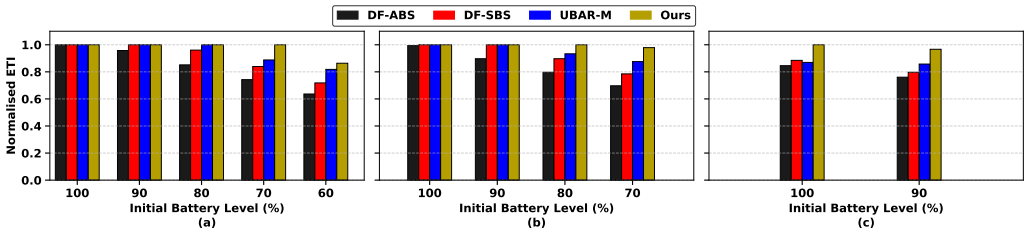

Fig. 7. Comparison of the Expected Time Index (ETI) for the three considered scenarios, (a) Scenario 1, (b) Scenario 2 and (c) Scenario 3 for considered approaches with varying initial battery levels (%).

(0.57 h) and 25.48 % (1.12 h) battery life improvement compared to UBAR-M, DF-SBS, and DF-ABS, respectively. For Scenario 3, where users engage in consistent resource-intensive workloads (e.g., gaming), both DF-SBS and DF-ABS fail to meet the expected battery life due to their unawareness of battery life goals. Similarly, UBAR-M fails to meet the expected battery life goal due to its initial higher performance target UBAR and also being workloads agnostic. In contrast, our approach, enabled by the resource evaluator and allocator, dynamically determines appropriate DVFS and display brightness levels to achieve the expected battery life. As a result, improving battery life by 14.98 % (0.72 h), 13 % (0.63 h) and 18.28 % (0.85 h) compared to UBAR-M, DF-SBS and DF-ABS, respectively. This demonstrates the effectiveness of our approach to meet battery life in a wide range of usage scenarios, exceeding the capabilities of existing approaches.

In addition, the impact of the approaches' inability to meet the expected battery life is shown in Figure 7, which depicts the normalised ETI for each approach evaluated at various initial battery levels. Our approach maintains 100 % ETI across nearly all initial battery levels considered, ensuring consistency throughout. This results in significant improvements of up to 5.6 %, 20.3 % and 35.7 % in ETI for scenario 1, compared to the UBAR-M, DF-SBS and DF-ABS approaches, respectively. Similarly, in scenario 2, our approach improves ETI by 11.8 %, 24.7 % and 40.4 % relative to the UBAR-M, DF-SBS and DF-ABS approaches, respectively. Additionally, in scenario 3, we achieve an ETI improvement of 15 %, 21.3 % and 27.1 % over UBAR-M, DF-SBS and DF-ABS, respectively. The significant increase in ETI validates the effectiveness of our approach in meeting the expected battery life, as this metric measures how closely the device's actual usage time corresponds to the user's expected battery life.

5.3 Comparison of Average QoE

Figure 8 shows the evaluated average QoE for each of the three scenarios considered with varying initial battery levels. The QoE is computed using the QoE model in Equation (2), and the average is normalised to the ideal rating score. In Scenario 1 (Figure 8(a)), where streaming and interactive workloads are evenly distributed, our approach outperforms UBAR-M, DF-SBS and DF-ABS by delivering a higher average QoE at 90 % and 100 % initial battery levels, with improvements of up to 4.6 %, 9.1 % and 2.13 %, respectively. As we decrease the initial battery level further, our approach effectively reduces energy consumption to meet the expected battery life, while still delivering a comparable higher average QoE of roughly 80 % for all other initial battery levels, which is higher than UBAR-M. Although our approach has a slightly lower QoE than DF-SBS and DF-ABS at 70 % and 60 % initial battery life, it is important to note that these approaches fail to meet the expected battery life starting at 80 % initial battery levels. Our approach achieves a balance by reducing energy consumption while still providing an acceptable QoE to meet the expected battery life. In Figure 8(b) and (c), similar patterns can be observed for Scenarios 2 and 3. Notably, in Scenarios 2

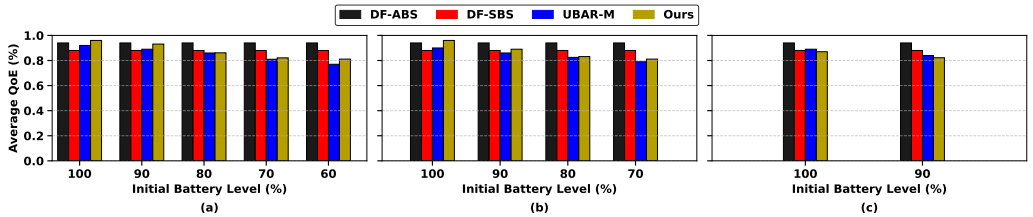

Fig. 8. Comparison of average QoE for the approaches under scenarios, (a) Scenario 1, (b) Scenario 2 and (c) Scenario 3 considered at varying initial battery level (%).

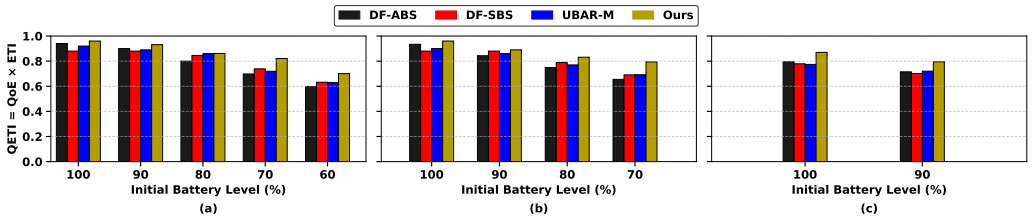

Fig. 9. Comparison of QoE-Expected Time Index (QETI = QoE × ETI) for the three scenarios ((a) Scenario 1, (b) Scenario 2 and (c) Scenario 3) and approaches considered.

and 3, both DF-SBS and DF-ABS approaches fail at 90 % initial battery level and at all initial battery levels, respectively.

In Figure 9, we present the QETI to demonstrate that our approach can achieve a balance between the competing goals of QoE and expected battery life. Our approach consistently achieves the highest QETI value across all three scenarios, indicating that it achieves the appropriate balance between QoE and battery life. As shown in Figure 9(a), our approach can result in 11.16 %, 14.17 %and 17.66 % improvement in QETI compared to DF-SBS, UBAR-M and DF-ABS approaches for scenario 1. Similar average QETI enhancements of 14.75 %, 14.95 % and 21.18 % are achieved for Scenario 2, and 12.40 %, 13.33 % and 11.16 % for Scenario 3, when compared to the UBAR-M, DF-SBS and DF-ABS approaches, respectively. The results demonstrate that, in comparison to existing approaches, our approach does well at maximising QoE while meeting the expected battery life across diverse scenarios.

While our approach was designed using data collected from the participants in Section 3, its evaluation across new users and scenarios establishes its foundational functional viability under varying workload conditions. For alternative hardware platforms where empirical parameters are not yet profile-calibrated, the framework is designed to initially utilise nominal baseline values while its structural model parameters are adjusted when core hardware specifications, such as cluster frequencies and target battery capacities, are provided. Similarly, for entirely new scenarios, the approach leverages closely matched centroids for initial workload classification, progressively adapting as additional execution data are processed.

6 Limitations and Generalisability

While this study demonstrates significant QoE and battery improvements on a commercial Pixel 3 platform, we acknowledge specific architectural shifts in the mobile ecosystem that necessitate a recalibration of the model's empirical coefficients.

Heterogeneous Architectures. Modern SoCs (e.g., Snapdragon 8 Gen 3) have moved beyond the dual-cluster big.LITTLE model to tri-cluster or quad-cluster configurations (comprising Prime, Performance, and Efficiency cores). While our specific β coefficients in Equation (1) are calibrated for two clusters, the Workload Clustering logic remains architecturally agnostic. Extending our model to a tri-cluster system simply requires the addition of a third term, $\beta_3 \cdot f_{prime}$, to the linear objective function. The underlying feature sets (CPU utilisation, memory frequency) remain standard across all contemporary Android **Hardware Abstraction Layers** (**HALs**). Furthermore, while modern NPUs aggressively offload ML inference workloads, the online nature of our K-means clustering allows the framework to adapt to these altered CPU/GPU utilisation patterns automatically.

*Display Technologies and **Operating System (OS)** Evolution.* The emergence of **Low-Temperature Polycrystalline Oxide** (**LTPO**) displays enables **Variable Refresh Rates** (**VRR**) from 1 Hz to 120 Hz. While the Pixel 3 utilised a fixed 60 Hz OLED, our joint management framework provides the necessary brightness ceiling, which acts as a primary power constraint. VRR scaling can be integrated as a third resource knob within our adaptation engine, where refresh rate is modulated alongside DVFS and brightness to further expand the energy-to-experience tradeoff space. In the context of OS evolution, our manager is designed to operate in tandem with the **Energy Aware Scheduler** (**EAS**) found in Android 11 through 15. Rather than implementing an active, dual-layer control architecture that directly overrides kernel-level thread migration, our framework structurally influences the scheduler by modifying the maximum allowable frequencies for each core cluster via sysfs. This cluster-based performance capping alters the capacity headroom evaluated by the EAS internal energy model. Consequently, the native scheduler retains exclusive authority over task placement, automatically executing migration decisions within the performance bounds established by our userspace process. This decoupled, non-intrusive relationship ensures that our high-level, user-centric energy goals are strictly enforced without interfering with or incurring the software overhead of low-level kernel thread migration logic.

Concurrent Tasks and Methodological Portability. A primary limitation involves the handling of multiple concurrent intensive applications. While our clustering mechanism identifies the profile of the active foreground process to ensure a seamless interaction, it does not explicitly decouple the resource demands of background tasks from the primary thread. However, since background noise is typically consistent across 200 ms windows, the K-means centroids effectively treat this as a base power floor, allowing the manager to focus on foreground-induced variances. Future iterations could incorporate a per-process accounting strategy (leveraging Android's cgroup or taskstats interfaces) to isolate foreground signatures. Finally, while we have validated this approach on a single SoC vendor (Qualcomm Snapdragon), the framework relies on standardised Android HAL interfaces. This structural reliance supports theoretical methodological portability to other architectures, such as MediaTek or Samsung Exynos, provided a one-time calibration of the power-performance baseline curves is executed. However, because our empirical evaluation was restricted to a single hardware platform, these findings demonstrate architectural feasibility rather than universal generalisable applicability. Consequently, empirical cross-platform validation across modern heterogeneous architectures, variable refresh rate displays, and dedicated neural processing hardware remains a necessary objective for future work.

7 Conclusions

Maximising user experience on mobile devices has become increasingly challenging due to technological imbalances among various mobile subsystems, coupled with increasing functionalities and diverse workloads explored by users. To address this challenge, this article presents an adaptive, hierarchical DVFS and brightness scaling approach that effectively balances QoE and battery life

within a user-acceptable range. This adaptive approach continuously adapts to varying workloads, contextual conditions, and the available energy budget to meet the expected battery life. The approach first identifies workload clusters and context at runtime, then selects appropriate DVFS and brightness settings based on workload type and context to achieve the target user-specified operational duration. This is achieved by employing workload- and context-based clustering, resource evaluation, and allocation, along with online adaptive DVFS and brightness scaling. Experimental results demonstrate significant improvements in QoE and battery life, with up to 9 % and 34.7 % enhancements compared to existing approaches, respectively. Furthermore, the proposed approach outperforms existing approaches by 21.2 % in terms of the QETI. These results underscore the effectiveness of our approach in balancing QoE and energy consumption to achieve the expected battery life. Our future work will prioritise large-scale user studies on the entire system to gather a broader range of qualitative user feedback on real-world performance and user perception.

Acknowledgments

All data supporting this work are openly available from the University of Southampton repository at https://doi.org/10.5258/SOTON/D3985.

References

[1] 2023. Genshin impact - step into a vast magical world of adventure. Retrieved from https://genshin.hoyoverse.com/en/

[2] 2023. UK top apps by reach 2023 | Statista. Retrieved August 10, 2023 from https://www.statista.com/statistics/1282098/top-mobile-apps-uk-by-reach

[3] Bhojan Anand, Karthik Thirugnanam, Jeena Sebastian, Pravein G. Kannan, Akhihebbal L. Ananda, Mun Choon Chan, and Rajesh Krishna Balan. 2011. Adaptive display power management for mobile games. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services*. 57–70.

[4] Martin Armstrong. 2020. The Apps Americans Can't Live Without | Statista. Retrieved February 10, 2021 from https://www.statista.com/chart/23230/apps-people-cant-do-without-united-states

[5] K. R. Basireddy, A. K. Singh, B. M. Al-Hashimi, and G. V. Merrett. 2020. AdaMD: Adaptive mapping and dvfs for energy-efficient heterogeneous multicores. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 10 (2020), 2206–2217. DOI: https://doi.org/10.1109/TCAD.2019.2935065

[6] Ethan Chen, Sushant Kondguli, Carl Marshall, and Yuhao Zhu. 2025. Modeling and exploiting the time course of chromatic adaptation for display power optimizations in virtual reality. *ACM Transactions on Graphics* 44, 6, Article 212 (2025) (December 2025), 1–21. DOI: https://doi.org/10.1145/3763294

[7] Kenneth Chen, Thomas Wan, Nathan Matsuda, Ajit Ninan, Alexandre Chapiro, and Qi Sun. 2024. Pea-pods: Perceptual evaluation of algorithms for power optimization in xr displays. *ACM Transactions on Graphics* 43, 4 (2024), 1–17.

[8] Wei-Chung Cheng and M. Pedram. 2004. Power minimization in a backlit TFT-LCD display by concurrent brightness and contrast scaling. *IEEE Trans. on Consumer Electronics* 50, 1 (2004), 25–32. DOI: https://doi.org/10.1109/TCE.2004.1277837

[9] Sofiane Chetoui and Sherief Reda. 2021. Workload-and user-aware battery lifetime management for mobile socs. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*. IEEE, 1679–1684.

[10] Yonghun Choi, Seonghoon Park, and Hojung Cha. 2019. Graphics-aware power governing for mobile devices. In *Proceedings of the Annual International Conference on Mobile Systems, Applications, and Services*. 469–481.

[11] Po-Kai Chuang, Ya-Shu Chen, and Po-Hao Huang. 2017. An adaptive on-line CPU-GPU governor for games on mobile devices. In *Proceedings of the 2017 22nd Asia and South Pacific Design Automation Conference*. 653–658. DOI: https://doi.org/10.1109/ASPDAC.2017.7858398

[12] Ryan Cochran, Can Hankendi, Ayse K. Coskun, and Sherief Reda. 2011. Pack and cap: Adaptive dvfs and thread packing under power caps. In *Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture*. 175–185.

[13] Adam Conway. 2018. *How EAS Helps Make the Google Pixel the Fastest Android Phone*. Retrieved February 2022 from https://www.xda-developers.com/google-pixel-fastest-android-phone-eas/

[14] Anup Das, Akash Kumar, Bharadwaj Veeravalli, Rishad Shafik, Geoff Merrett, and Bashir Al-Hashimi. 2015. Workload uncertainty characterization and adaptive frequency scaling for energy minimization of embedded systems. In *Proceedings of the 2015 Design, Automation and Test in Europe Conference and Exhibition*. IEEE, 43–48.

[15] Somdip Dey, Amit Kumar Singh, Xiaohang Wang, and Klaus McDonald-Maier. 2020. User interaction aware reinforcement learning for power and thermal efficiency of CPU-GPU mobile MPSoCs. In *Proceedings of the 2020 Design, Automation and Test in Europe Conference and Exhibition*. IEEE, 1728–1733.

[16] Gaurav Dhiman and Tajana Simunic Rosing. 2007. Dynamic voltage frequency scaling for multi-tasking systems using online learning. In *Proceedings of the International Symposium on Low Power Electronics and Design*. 207–212.
[17] Budmonde Duinkharjav, Kenneth Chen, Abhishek Tyagi, Jiayi He, Yuhao Zhu, and Qi Sun. 2022. Color-perception-guided display power reduction for virtual reality. *ACM Transactions on Graphics* 41, 6 (2022), 1–16.
[18] Andrei Frumusanu. [n. d.]. The Google Pixel 3 Review: The Ultimate Camera test. Retrieved August 4, 2020 from https://www.anandtech.com/show/13474/the-google-pixel-3-review
[19] Scott Adam Gordon. 2018. Here's why Android Pie's Adaptive Brightness mode is so efficient. Retrieved June 16, 2022 from www.androidauthority.com/android-pie-adaptive-brightness-927080
[20] Manjari Gupta, Lava Bhargava, and S. Indu. 2022. Deep neural network learning for power limited heterogeneous system with workload classification. *Computing* 104 (2022), 95–122.
[21] Matthew Halpern, Yuhao Zhu, and Vijay Janapa Reddi. 2016. Mobile CPU's rise to power: Quantifying the impact of generational mobile CPU design trends on performance, energy, and user satisfaction. In *Proceedings of the International Symposium on High Performance Computer Architecture*. 64–76.
[22] Ali Iranli and Massoud Pedram. 2005. DTM: Dynamic tone mapping for backlight scaling. In *Proceedings of the Annual Design Automation Conference*. 612–617.
[23] Samuel Isuwa, David Amos, Amit Kumar Singh, Bashir Al-Hashimi, and Geoff V. Merrett. 2023. Maximising mobile user experience through self-adaptive content-and ambient-aware display brightness scaling. *Journal of Systems Architecture* 145 (2023), 103023.
[24] Samuel Isuwa, David Amos, Amit Kumar Singh, Bashir M. Al-Hashimi, and Geoff V. Merrett. 2023. Content-and lighting-aware adaptive brightness scaling for improved mobile user experience. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*. IEEE, 1–2.
[25] Samuel Isuwa, Somdip Dey, Andre P. Ortega, Amit Kumar Singh, Bashir M. Al-Hashimi, and Geoff V. Merrett. 2022. Quarem: Maximising QoE through adaptive resource management in mobile mpsoc platforms. *ACM Transactions on Embedded Computing Systems* 21, 4 (2022), 1–29.
[26] ITU-R. 2023. BT.500: Methodologies for the subjective assessment of the quality of television images. Retrieved September 15, 2023 from https://www.itu.int/rec/R-REC-BT.500-15-202305-I/en
[27] Seyeon Kim, Kyungmin Bin, Sangtae Ha, Kyunghan Lee, and Song Chong. 2021. zTT: Learning-based DVFs with zero thermal throttling for mobile devices. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*. 41–53.
[28] Xianfeng Li and Gengchao Li. 2021. An adaptive CPU-GPU governing framework for mobile games on big.LITTLE architectures. *IEEE Trans. Comput.* 70, 9 (2021), 1472–1483. DOI: https://doi.org/10.1109/TC.2020.3012987
[29] Chengdong Lin, Kun Wang, Zhenjiang Li, and Yu Pu. 2023. A workload-aware dvfs robust to concurrent tasks for mobile devices. In *Proceedings of the Annual International Conference on Mobile Computing and Networking*. 1–16.
[30] Weikai Lin, Sushant Kondguli, Carl Marshall, and Yuhao Zhu. 2025. PowerGS: Display-rendering power co-optimization for neural rendering in power-constrained xr systems. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*. 1–12.
[31] Sparsh Mittal. 2014. A survey of techniques for improving energy efficiency in embedded computing systems. *International Journal of Computer Aided Engineering and Technology* 6, 4 (2014), 440–459.
[32] Iqbal Mohomed, Jim Chengming Cai, Sina Chavoshi, and Eyal De Lara. 2006. Context-aware interactive content adaptation. In *Proceedings of the International Conference on Mobile Systems, Applications and Services*. 42–55.
[33] Daniele Jahier Pagliari, Santa Di Cataldo, Edoardo Patti, Alberto Macii, Enrico Macii, and Massimo Poncino. 2021. Low-overhead adaptive brightness scaling for energy reduction in oled displays. *IEEE Transactions on Emerging Topics in Computing* 9, 3 (2021), 1625–1636. DOI: https://doi.org/10.1109/TETC.2019.2908257
[34] Jurn-Gyu Park, Chen-Ying Hsieh, Nikil Dutt, and Sung-Soo Lim. 2017. Synergistic CPU-GPU frequency capping for energy-efficient mobile games. *ACM Trans. on Embedded Computing Systems* 17, 2 (2017), 1–24.
[35] Chao Qian, Daibo Liu, and Hongbo Jiang. 2022. Harmonizing energy efficiency and QoE for brightness scaling-based mobile video streaming. In *Proceedings of the IEEE/ACM 30th International Symposium on Quality of Service*. IEEE, 1–10.
[36] Qualcomm. [n. d.]. Snapdragon 845 Mobile Platform. Retrieved June 2, 2020 from https://https://www.qualcomm.com/products/snapdragon-845-mobile-platform
[37] V. J. Reddi, H. Yoon, and A. Knies. 2018. Two billion devices and counting. *IEEE Micro* 38, 1 (2018), 6–21. DOI: https://doi.org/10.1109/MM.2018.011441560
[38] Onur Sahin, Lothar Thiele, and Ayse K. Coskun. 2018. Maestro: Autonomous qos management for mobile applications under thermal constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38, 8 (2018), 1557–1570.
[39] Matthew Schuchhardt, Susmit Jha, Raid Ayoub, Michael Kishinevsky, and Gokhan Memik. 2014. CAPED: Context-aware personalized display brightness for mobile devices. In *Proceedings of the 2014 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*. 1–10.

[40] Elham Shamsa, Alma Pröbstl, Nima TaheriNejad, Anil Kanduri, Samarjit Chakraborty, Amir M. Rahmani, and Pasi Liljeberg. 2021. Ubar: User-and battery-aware resource management for smartphones. *ACM Transactions on Embedded Computing Systems* 20, 3 (2021), 1–25.

[41] Alex Shye, Berkin Ozisikyilmaz, Arindam Mallik, Gokhan Memik, Peter A. Dinda, Robert P. Dick, and Alok N. Choudhary. 2008. Learning and leveraging the relationship between architecture-level measurements and individual user satisfaction. In *Proceedings of the International Symposium on Computer Architecture*. 427–438.

[42] Amit Kumar Singh, Somdip Dey, Klaus McDonald-Maier, Karunakar Reddy Basireddy, Geoff V. Merrett, and Bashir M. Al-Hashimi. 2020. Dynamic energy and thermal management of multi-core mobile platforms: A survey. *IEEE Design and Test* 37, 5 (2020), 25–33.

[43] Wei Sun, Xiongkuo Min, Guangtao Zhai, Ke Gu, Siwei Ma, and Xiaokang Yang. 2020. Dynamic backlight scaling considering ambient luminance for mobile videos on lcd displays. *IEEE Transactions on Mobile Computing* 21, 1 (2020), 110–124.

[44] Tan Kiat Wee and Rajesh Krishna Balan. 2012. Adaptive display power management for OLED displays. In *Proceedings of the ACM International Workshop on Mobile Gaming*. 25–30.

[45] Zhisheng Yan, Qian Liu, Tong Zhang, and Chang Wen Chen. 2018. CrowdDBS: A crowdsourced brightness scaling optimization for display energy reduction in mobile video. *IEEE Transactions on Mobile Computing* 17, 11 (2018), 2536–2549.

Received 6 October 2025; revised 26 June 2026; accepted 13 August 2026