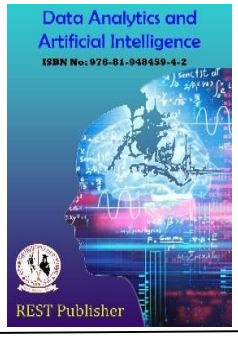
Data Analytics and Artificial Intelligence
Vol: 3(5), 2023
REST Publisher; ISBN: 978-81-948459-4-2
Website: http://restpublisher.com/book-series/daai/
DOI: https://doi.org/10.46632/daai/3/5/4



Android-Based Power-Saving Framework for Mobile Devices Using the DEMATEL Method

E.V.N.Jyothi, M.Kranthi
Pace Institute of Technology & Sciences, Ongole, A.P, India.
Rise krishna Sai Prakasam Group of Institutions, Ongole, A.P, India.
Corresponding Author Email: jyothiendluri@pace.ac.in

Abstract: *Android-based power-saving framework"that is universally recognized. However, I can provide you with information about power-saving techniques and strategies commonly used in Android development up to that point. Keep in mind that developments might have occurred after September 2021. Android devices are known for their versatility and feature-rich environment, but this can come at the cost of increased power consumption. To mitigate this issue, developers and device manufacturers have employed various power-saving techniques and frameworks. Here are some common strategies and frameworks: Doze Mode and App Standby: Android introduced Doze Mode, which helps conserve battery life by delaying background CPU and network activity when a device is idle. App Standby takes this further by putting apps into a low-power state when they aren't actively used, reducing their impact on battery life. Background Execution Limits: Android limits background execution of apps to prevent unnecessary battery drain. Apps can only run background tasks within specific restrictions, ensuring that they don't continuously consume resources. JobScheduler: This framework allows apps to schedule tasks at optimal times, which can help consolidate tasks and reduce the frequency of waking up the device, thus saving power. Battery Optimization: Android provides a battery optimization feature that allows users to prioritize apps and restrict background activity for specific apps, helping to save power. Location Services: Managing location updates efficiently can significantly impact battery life. Using lower accuracy settings or batching location updates can reduce the power consumed by location services. Wakelocks and Alarms: Developers can use wakelocks and alarms to keep the device awake for specific tasks. However, these should be used judiciously, as they can lead to increased power consumption if not managed properly. Optimized Networking: Using techniques like Volley or OkHttp for efficient network requests, and optimizing the use of background data syncing, can help reduce power consumption. Background Syncing: DEMATEL is widely accepted for analyzing the overall relationship of factors and classifying factors into cause-and-effect types. Therefore, this article considers each source as a criterion in decision-making. To deal with a mixture of conflicting evidence, the significance and level of significance of each piece of evidence can be determined using DEMATEL; however, expanding the DEMATEL method with the source theory is required for better conclusions. Screen brightness & colour scheme, CPU frequency, Network, Low power localization and Wi-Fi. Rank using the DEMATEL for Android-based power-saving framework in Screen brightness & colour scheme is got the first rank whereas is the CPU frequency is having the Lowest rank.*

Keywords: *Screen brightness & colour scheme, CPU frequency, Network, Low power localization and Wi-Fi.*

1. INTRODUCTION

Android-based power-saving framework" that is universally recognized. However, I can provide you with information about power-saving techniques and strategies commonly used in Android development up to that point. Keep in mind that developments might have occurred after September 2021. Android devices are known for their versatility and feature-rich environment, but this can come at the cost of increased power consumption. To mitigate this issue, developers and device manufacturers have employed various power-saving techniques and frameworks. Here are some common strategies and frameworks: Doze Mode and App Standby: Android

introduced Doze Mode, which helps conserve battery life by delaying background CPU and network activity when a device is idle [1]. App Standby takes this further by putting apps into a low-power state when they aren't actively used, reducing their impact on battery life. Background Execution Limits: Android limits background execution of apps to prevent unnecessary battery drain. Apps can only run background tasks within specific restrictions, ensuring that they don't continuously consume resources. Job Scheduler: This framework allows apps to schedule tasks at optimal times, which can help consolidate tasks and reduce the frequency of waking up the device, thus saving power [2]. Battery Optimization: Android provides a battery optimization feature that allows users to prioritize apps and restrict background activity for specific apps, helping to save power. Location Services: Managing location updates efficiently can significantly impact battery life. Using lower accuracy settings or batching location updates can reduce the power consumed by location services. Wakelocks and Alarms: Developers can use wakelocks and alarms to keep the device awake for specific tasks. However, these should be used judiciously, as they can lead to increased power consumption if not managed properly. Optimized Networking: Using techniques like Volley or OkHttp for efficient network requests, and optimizing the use of background data syncing, can help reduce power consumption [3]. Background Syncing: Minimizing the frequency of background data syncing and batching multiple sync operations can reduce the device's active time, saving power. UI Optimization: Efficiently managing UI updates and animations can also impact battery life. Minimizing unnecessary redraws and animations can help save power. Battery Historian and Profiling Tools: Developers can use tools like Battery Historian and Android Profiler to analyze power usage patterns in their apps, identify bottlenecks, and optimize accordingly [4]. Android stands out as an open platform, gaining immense popularity as an operating system. It boasts open-source code that's easy to manage, allowing users to access and utilize new content and applications on their mobile devices. Built upon a Linux kernel, Android devices are being implemented at an impressive rate. However, managing the power consumption of these devices has emerged as a notable challenge, primarily due to the prevalent issue of limited battery life [5]. This isn't solely confined to regular occurrences; even smartphones face these challenges. With the proliferation of power-hungry technologies such as GPS, 3G, and 3GS, the situation becomes more complex. Addressing this, a client-server approach, devoid of intricate structures, has been adopted. Leveraging machine learning, this approach oversees the applications in use, monitors battery consumption, and gauges contextual factors. Over a designated time span, this system gathers pertinent information, which then fuels the machine learning process [6]. In the contemporary era, the energy consumption has surged considerably, especially with the advent of robust technologies like GPS, 3G, and Wi-Fi. These applications, along with others such as video streaming services like YouTube, exert a significant drain on power resources. Users frequently find that such applications consume substantial battery power. The resolution to these challenges lies in comprehending the mechanisms that contribute to phone battery depletion. By doing so, viable solutions can be devised to address these issues. A common occurrence on Android phones is the automatic initiation of numerous applications. This occurrence is driven by two primary factors: the apps starting automatically upon certain triggers and apps being permitted to operate in the background [7]. These applications often engage in network activities, which further escalates their power usage. While developers do so with the intention of improving the features and functionalities of nearby applications, this practice presents certain hurdles. Determining which applications are more power-intensive and which offer energy-efficient utility poses a significant challenge for most end-users. The assessment of energy usage within applications on mobile phones led us to develop the Battery Viewer system. The process entails several steps. Initially, the application is launched, followed by an evaluation of power usage across mobile devices. This evaluation involves gathering data on application-specific power consumption. Subsequently, the collected data is utilized to present information on the chosen application in a tab-based format on the screen [8]. The final step involves the representation of energy consumption using graphical representations, particularly through a graph depicting application-based energy consumption over time. The system for optimizing battery usage through location-triggered alarms is an Android application. In the current era, mobile devices are abundant, with their prevalence attributed to the expansive nature of Android, an open-source operating system. This Android-based application is developed with the objective of generating alerts based on user-defined locations or set times. The application's core function lies in delivering reminders to users, be it through specified geographical locations or scheduled times. While conventional time-based alerts are commonplace, this system introduces a novel approach by incorporating location-based triggers. Thus, the alarm system doesn't just rely on time factors; it also activates based on the user's geographical context [9]. The application is designed to cater to users, enabling them to input an address. Subsequently, this address undergoes a transformation into latitude and longitude coordinates through GPS or

network-based location providers. These coordinates serve as reference points for issuing alerts when the user approaches the specified destination. The user can define the targeted location, be it their own or another individual's. This mutual configuration leads to reciprocal notifications. Each party receives notifications based on their defined proximity thresholds. GPS, which stands for Global Positioning System, relies on satellite-based information to determine the Android device's current location. Network providers also supply location data via cell towers [10]. By employing this technology, users can establish alarms that trigger as they draw near their set destinations. This negates the need for continuous manual checks. This approach aligns with the current trend of heightened carpooling activities. Our app is optimized for multi-user scenarios and facilitates location-based notifications between users, especially beneficial for carpooling or coordinating major transportation services [11]. The application's foundation rests on a location-based system tailored for Android devices. It effectively retrieves the device's current geographical coordinates using the Global Positioning System (GPS) satellite data. This occurs within the confines of our battery-efficient application, ensuring minimal battery drain. Regardless of whether the device is charging via a USB, AC adapter, or discharging, the application adapts its updates and background service frequency accordingly. Wireless services are consistently expanding their reach, becoming ubiquitous across various environments. A surge in multimedia services underscores the necessity to cater to diverse quality-of-service (QoS) requirements. QoS signifies the ability to deliver service in accordance with network capacity, while quality of experience (QoE) gauges users' perceptual satisfaction [12]. This entails meeting specified service levels while the service is active, ensuring usability and quality of service. More precisely, usability pertains to user satisfaction based on factors like access, service retention, and integrity. Mobile phone users might encounter performance inadequacies due to factors such as limited bandwidth or fluctuations caused by movement during multimedia transmissions. Android stands at the forefront of mobile operating systems, embodying the latest trends in this domain. It offers a comprehensive environment for developers to create applications, encompassing middleware, a phone application stack, and a Linux kernel that is both feature-rich and versatile [13]. Despite these extensive features, Android tackles limitations at the kernel level, particularly concerning energy management for specific Android instances. It endeavours to address these limitations by actively engaging in the Linux ecosystem and formulating potential solutions. The objective is to overcome challenges in energy efficiency, with a focus on continuous power management. Solutions developed include proposing suitable governor algorithms and modifying their parameters, as well as implementing daemon processes that regulate voltage and frequency scaling [14]. To enhance the energy efficiency of Android applications, techniques are presented for application developers to improve their software. The research articulates the project's objectives and implements these strategies, using OMAP3530-based low-power techniques as a foundation for experimentation. These techniques are put to the test in a variety of scenarios, including 2D/3D rendering, movie playback, and data decompression. The results are evaluated based on performance metrics such as execution time, total current, waiting time, and battery life [15].

2. MATERIALS AND METHOD

2.1. Screen Brightness & Colour Scheme: Adaptive Brightness: Android devices often include an adaptive brightness feature that automatically adjusts screen brightness based on ambient lighting conditions. Enabling this can help save power by reducing brightness in well-lit environments. Dark Mode: Many modern Android versions offer a dark mode option, which uses darker colours for the user interface. Dark mode can be easier on the eyes and also save power, especially on devices with OLED displays.

2.2. CPU Frequency: CPU Governors: Android devices use CPU governors to manage the CPU frequency dynamically. Performance governors can increase the CPU speed when needed, but they consume more power. Power-saving governors aim to keep the CPU frequency lower when the device is idle or under light usage. CPU Throttling: Throttling the CPU during less demanding tasks can help save power. Techniques like dynamic voltage and frequency scaling (DVFS) adjust the CPU frequency based on workload.

2.3. Network: Network Type: Different network types (2G, 3G, 4G, 5G) consume varying amounts of power. Switching to a lower network type when high-speed data isn't necessary can save battery life.

Background Data: Restricting background data usage for apps can prevent unnecessary network activity when the device is not in use.

2.4. Low Power Localization: Passive Location: Using passive location updates instead of actively polling for location can reduce power consumption. Passive location relies on location data provided by other apps or services. Geofencing: Instead of continuous GPS usage, geofencing allows apps to be notified when a device enters or exits a predefined geographical area.

2.5. Wi-Fi: Wi-Fi Scanning: Disable Wi-Fi scanning when not needed. Scanning for Wi-Fi networks in the background can consume power. Wi-Fi Sleep Policy: Adjust the Wi-Fi sleep policy to disconnect from Wi-Fi when the screen is off or when the device is idle.

2.6. Method: The DEMATEL method quickly separates the complex set of factors into a sender organization and a receiving institution, and then translates that information into the appropriate strategy for selecting a management tool. Also, the ZOGP model enables businesses to fully utilize their limited funds for planning to develop ideal management systems by combining different configurations with Explicit Priorities [16]. DEMATEL methods. This impact and causality can be attributed to affected group barricades. Therefore, to effectively implement electronic waste management, barriers belonging to a causally Influential subgroup should be given special consideration. Decision-makers must therefore identify hurdles in order to reduce their impact or influence, guarantee that the legal is strong, and ensure that appropriate barriers are in place [17]. Therefore, der methods ISM and DEMATEL methods, the results are somewhat consistent results grated ISM DEMATEL results for e-was determination constraints determine not only the structure of fire but also the structure of the interactions DEMATEL research, specific applications for DEMATEL. as for which DEMATEL is only. categories: factors or only relationships between criteria the first type of clarification is: and causal Group barriers pro or Source for affected group barriers can be considered due. Therefore, in order to effectively implement electronic waste management, barriers belonging to a causal or an influential group should be considered on a priority basis [18]. Therefore, decision makers need to determine obstacles the legal framework is strong make sure there is controllable in order to minimize impact or influence barriers. Therefore, derived structure of the interactions between these barriers is determined by the integrated ISM DEMATEL results for e-waste management constraints [19]. DEMATEL research, specific applications for DEMATEL. categories: factors or only relationships between criteria the first type of clarification involves identifying the main factors in terms of causal relationships and interrelationship size, while the second involves identifying the criteria for relationship and impact level analysis. DEMATEL method. As a result, the preliminary disadvantage (cluster one) was about topics such as the comparative weights of selection makers in the DEMATEL approach, which now does not take into account linking to team decision-making [20]. Obviously, in a group decision-making hassle, regular decision-makers can always trust their point of view and count on it to be prevalent among other selection-makers. This way, the very last evaluation guides must be close to their judgments, and if the very last assessment effects are close to their critiques, the choice maker is willing to simply accept it; otherwise, they may deny it. It is believed that methods based on unstructured comparisons, such as DEMATEL, play a significant role in the aforementioned discrepancies [21]. DEMATEL is widely accepted for analyzing the overall relationship of factors and classifying factors into cause-and-effect types. Therefore, this article considers each source as a criterion in decision-making. To deal with a mixture of conflicting evidence, the significance and level of significance of each piece of evidence can be determined using DEMATEL; however, expanding the DEMATEL method with the source theory is required for better conclusions [22].

3. RESULT AND DISCUSSION

TABLE 1. Android-based power-saving framework

	Screen brightness & colour scheme	**CPU frequency**	**Network**	**Low power localization**	**Wi-Fi**	**Sum**
Screen brightness & colour scheme	0	19	17	15	18	69
CPU frequency	9	0	5	17	15	46
Network	19	15	0	14	15	63
Low power localization	17	13	12	0	11	53
Wi-Fi	13	15	11	19	0	58

Table 1 shows that DEMATEL Decision making trail and evaluation laboratory in Android-based power-saving framework Screen brightness & colour scheme: 69, CPU frequency: 46, Network: 63, Low power localization: 53, Wi-Fi: 58 it is also Sum of values.

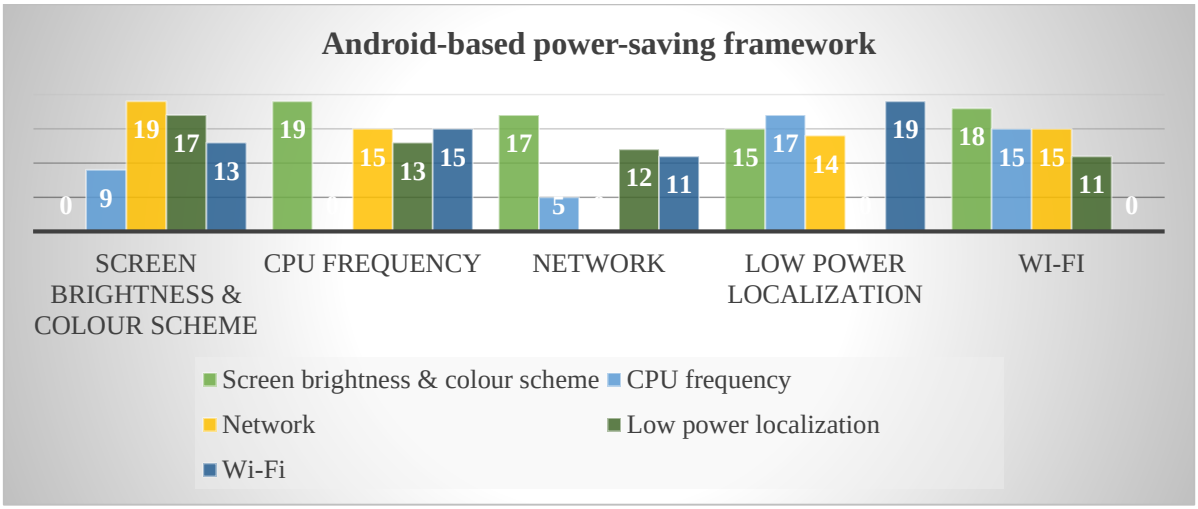


FIGURE 1. Android-based power-saving framework

Figure 1 shows that DEMATEL Decision making trail and evaluation laboratory in Android-based power-saving framework Screen brightness & colour scheme: 69, CPU frequency: 46, Network: 63, Low power localization: 53, Wi-Fi: 58 it is also Sum of values.

TABLE 2. Normalization of direct relation matrix

	Screen brightness & colour scheme	**CPU frequency**	**Network**	**Low power localization**	**Wi-Fi**
Screen brightness & colour scheme	0	0.275362319	0.246376812	0.217391304	0.260869565
CPU frequency	0.130434783	0	0.072463768	0.246376812	0.217391304
Network	0.275362319	0.217391304	0	0.202898551	0.217391304
Low power localization	0.246376812	0.188405797	0.173913043	0	0.15942029
Wi-Fi	0.188405797	0.217391304	0.15942029	0.275362319	0

Table 2 shows that the Normalizing of the direct relation matrix in Android-based power-saving framework is Screen brightness & colour scheme, CPU frequency, Network, Low power localization Wi-Fi the diagonal value of all the data set is zero.

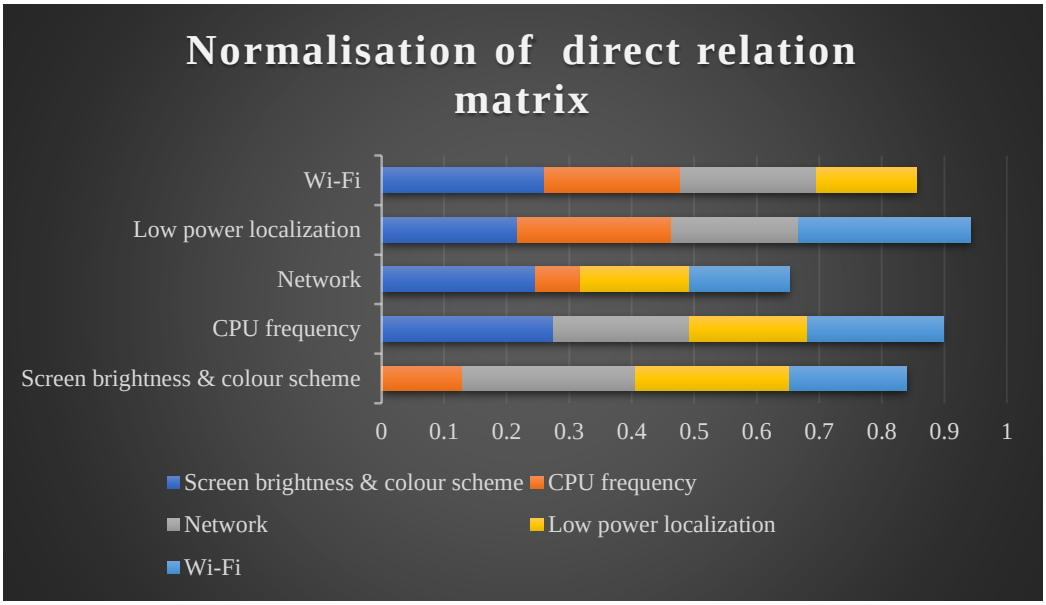


FIGURE 2. Normalization of direct relation matrix

Figure 2 shows that the Normalizing of the direct relation matrix in Android-based power-saving framework is Screen brightness & colour scheme, CPU frequency, Network, Low power localization Wi-Fi the diagonal value of all the data set is zero.

TABLE 3. Calculate the Total Relation Matrix

	Screen brightness & colour scheme	**CPU frequency**	**Network**	**Low power localization**	**Wi-Fi**
Screen brightness & colour scheme	0	0.275362319	0.246376812	0.217391304	0.260869565
CPU frequency	0.130434783	0	0.072463768	0.246376812	0.217391304
Network	0.275362319	0.217391304	0	0.202898551	0.217391304
Low power localization	0.246376812	0.188405797	0.173913043	0	0.15942029
Wi-Fi	0.188405797	0.217391304	0.15942029	0.275362319	0

Table 3 Shows the Calculate the total relation matrix in Android-based power-saving framework with Screen brightness & colour scheme, CPU frequency, Network, Low power localization Wi-Fi is Calculate the Value.

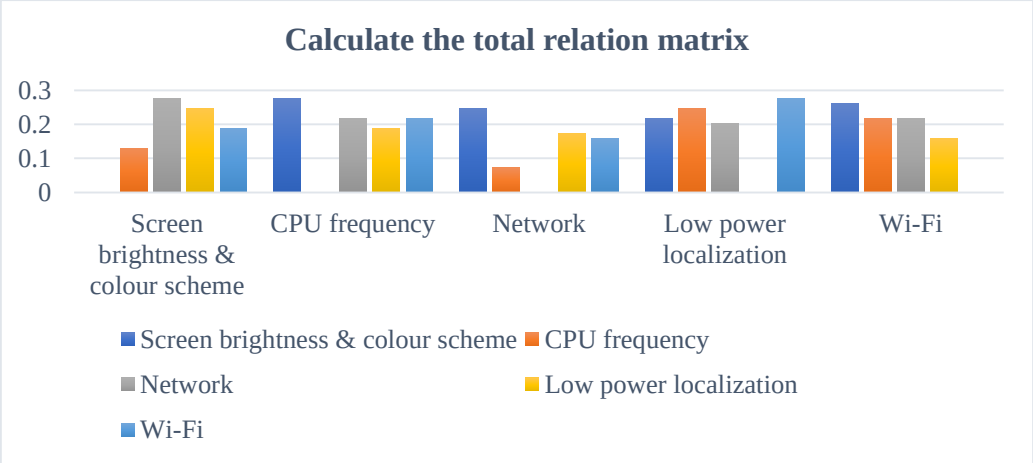


FIGURE 3. Calculate the total relation matrix

Figure 3 Shows the Calculate the total relation matrix in Android-based power-saving framework with Screen brightness & colour scheme, CPU frequency, Network, Low power localization Wi-Fi is Calculate the Value.

TABLE 4. T= Y(I-Y)-1, I= Identity matrix

I				
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Table 4 Shows the T= Y(I-Y)-1, I= Identity matrix in Android-based power-saving framework is Screen brightness & colour scheme, CPU frequency, Network, Low power localization Wi-Fi is the common Value.

TABLE 5. Y Value

Y				
0	0.2753623	0.2463768	0.2173913	0.2608696
0.1304348	0	0.0724638	0.2463768	0.2173913
0.2753623	0.2173913	0	0.2028986	0.2173913
0.2463768	0.1884058	0.173913	0	0.1594203
0.1884058	0.2173913	0.1594203	0.2753623	0

Table 5 Shows the Y Value in Android-based power-saving framework is Screen brightness & colour scheme, CPU frequency, Network, Low power localization Wi-Fi is Calculate the total relation matrix Value and Y Value is the same value.

TABLE 6. I-Y Value

I-Y				
1	-0.275362319	-0.246376812	-0.217391304	-0.260869565
-0.130434783	1	-0.072463768	-0.246376812	-0.217391304
-0.275362319	-0.217391304	1	-0.202898551	-0.217391304
-0.246376812	-0.188405797	-0.173913043	1	-0.15942029
-0.188405797	-0.217391304	-0.15942029	-0.275362319	1

Table 6 Shows the I-Y Value in Android-based power-saving framework is Screen brightness & colour scheme, CPU frequency, Network, Low power localization Wi-Fi table 4 T= Y(I-Y)-1, I= Identity matrix and table 5 Y Value Subtraction Value.

TABLE 7. (I-Y)-1 Value

(I-Y)-1				
1.984256264	1.268368455	0.99706193	1.280519894	1.214257788
0.807115864	1.73529332	0.634510675	0.974347333	0.88105748
1.143771056	1.167613903	1.754906269	1.202271871	1.125372061
1.000412351	1.014964848	0.802150678	1.894085469	0.957957904
1.007121531	1.081830069	0.826439088	1.166298306	1.863500091

Table 7 shows the (I-Y)-1Value in Android-based power-saving framework is Screen brightness & colour scheme, CPU frequency, Network, Low power localization Wi-Fi Table 6 shows the Minverse shows used.

TABLE 8. Total Relation matrix (T)

Screen brightness & colour scheme	0.984256264	1.2683685	0.9970619	1.2805199	1.2142578
CPU frequency	0.807115864	0.7352933	0.6345107	0.9743473	0.8810575
Network	1.143771056	1.1676139	0.7549063	1.2022719	1.1253721
Low power localization	1.000412351	1.0149648	0.8021507	0.8940855	0.9579579
Wi-Fi	1.007121531	1.0818301	0.8264391	1.1662983	0.8635001

Table 8 shows the Total Relation Matrix (T) the direct relation matrix is multiplied by the inverse of the value that the direct relation matrix is subtracted from the identity matrix.

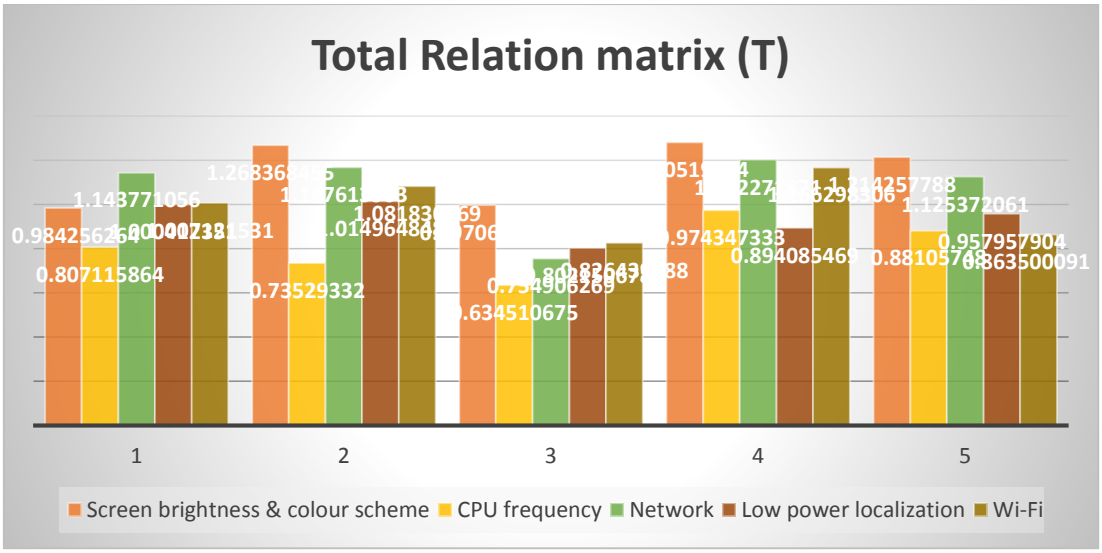


FIGURE 4. Total Relation matrix (T)

Figure 4 shows the Total Relation Matrix (T) the direct relation matrix is multiplied with the inverse of the value that the direct relation matrix is subtracted from the identity matrix.

TABLE 9. Android-based power-saving framework Ri & Ci Value

	Ri	Ci
Screen brightness & colour scheme	5.7444643	4.9426771
CPU frequency	4.0323247	5.2680706
Network	5.3939352	4.0150686
Low power localization	4.6695712	5.5175229
Wi-Fi	4.9451891	5.0421453

Table 9 shows the Android-based power-saving framework Ri & Ci Value Android-based power-saving in Screen brightness & colour scheme, CPU frequency, Network, Low power localization and Wi-Fi in Android-based power-saving framework in Screen brightness & colour scheme is showing the Highest Value for Ri and CPU frequency is showing the lowest value. Low power localization is showing the Highest Value for Ci and Network is showing the lowest value.

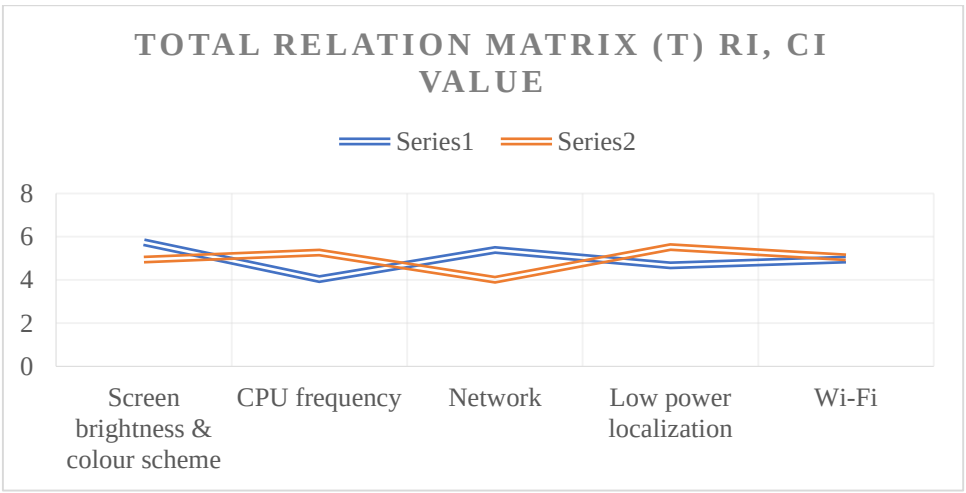


FIGURE 5. Total Relation Matrix (T) Ri, Ci Value

Figure 5 shows the Android-based power-saving framework Ri & Ci Value Android-based power-saving in Screen brightness & colour scheme, CPU frequency, Network, Low power localization and Wi-Fi in Android-based power-saving framework in Screen brightness & colour scheme is showing the Highest Value for Ri and CPU frequency is showing the lowest value. Low power localization is showing the Highest Value for Ci and Network is showing the lowest value.

TABLE 10. Calculation of Ri+Ci and Ri-Ci to Get the Cause and Effect

	Ri+Ci	Ri-Ci	Rank	Identity
Screen brightness & colour scheme	10.68714139	0.8017873	1	cause
CPU frequency	9.300395265	-1.2357459	5	effect
Network	9.409003799	1.3788665	4	cause
Low power localization	10.18709412	-0.8479516	2	effect
Wi-Fi	9.987334407	-0.0969562	3	effect

Table 10 shows the Calculation of Ri+Ci and Ri-Ci to Get the Cause and Effect. Android-based power-saving framework, Screen brightness & colour scheme, CPU frequency, Network, Low power localization and Wi-Fi Android-based power-saving framework in Screen brightness & colour scheme, Network is Showing the highest Value of cause. Android-based power-saving framework in CPU frequency, Low power localization and Wi-Fi is showing the lowest Value of effect.

TABLE 11. T matrix value

T matrix				
0.984256264	**1.268368**	**0.997062**	**1.28052**	**1.214258**
0.807115864	0.7352933	0.6345107	0.9743473	0.8810575
1.143771056	**1.167614**	0.7549063	**1.202272**	**1.125372**
1.000412351	**1.014965**	0.8021507	0.8940855	0.9579579
1.007121531	**1.08183**	0.8264391	**1.166298**	0.8635001

Table 11. Shows the T matrix calculate the average of the matrix and its threshold value (alpha) **Alpha 0.99141938** If the T matrix value is greater than threshold value then bold it.

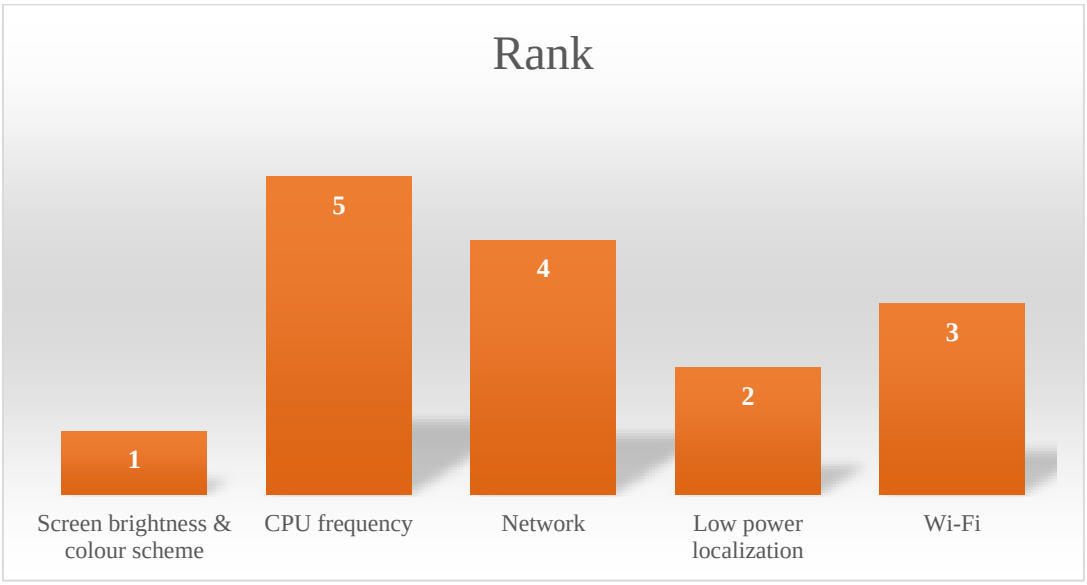


FIGURE 6. Shown the Rank

Figure 6 shows the Rank using the DEMATEL for Android-based power-saving framework in Screen brightness & colour scheme is got the first rank whereas is the CPU frequency is having the Lowest rank.

4. CONCLUSION

Android-based power-saving framework" that is universally recognized. However, I can provide you with information about power-saving techniques and strategies commonly used in Android development up to that point. Keep in mind that developments might have occurred after September 2021. Android devices are known for their versatility and feature-rich environment, but this can come at the cost of increased power consumption. To mitigate this issue, developers and device manufacturers have employed various power-saving techniques and frameworks. Here are some common strategies and frameworks: Doze Mode and App Standby: Android introduced Doze Mode, which helps conserve battery life by delaying background CPU and network activity when a device is idle. App Standby takes this further by putting apps into a low-power state when they aren't actively used, reducing their impact on battery life. Background Execution Limits: Android limits background execution of apps to prevent unnecessary battery drain. Apps can only run background tasks within specific restrictions, ensuring that they don't continuously consume resources. Job Scheduler: This framework allows apps to schedule tasks at optimal times, which can help consolidate tasks and reduce the frequency of waking up the device, thus saving power. Battery Optimization: Android provides a battery optimization feature that allows users to prioritize apps and restrict background activity for specific apps, helping to save power. Location Services: Managing location updates efficiently can significantly impact battery life. Using lower accuracy settings or batching location updates can reduce the power consumed by location services. Wake locks and Alarms: Developers can use wake locks and alarms to keep the device awake for specific tasks. To enhance the energy efficiency of Android applications, techniques are presented for application developers to improve their software. The research articulates the project's objectives and implements these strategies, using OMAP3530-based low-power techniques as a foundation for experimentation. These techniques are put to the test in a variety of scenarios, including 2D/3D rendering, movie playback, and data decompression. The results are evaluated based on performance metrics such as execution time, total current, waiting time, and battery life. the DEMATEL for Android-based power-saving framework in Screen brightness & colour scheme is got the first rank whereas is the CPU frequency is having the Lowest rank.

REFERENCES

[1]. Zeebaree, S., and Hajar M. Yasin. "Arduino based remote controlling for home: power saving, security and protection." International Journal of Scientific & Engineering Research 5, no. 8 (2014): 266-272.

[2]. Bala, Rimpy, and Anu Garg. "Battery power saving profile with learning engine in Android phones." International Journal of Computer Applications 69, no. 13 (2013): 38-41.

[3]. Saranya, G., S. Lavanya, and S. Sivasankari. "An efficient power saving technique-based location alarm for smart phones." In Journal of Physics: Conference Series, vol. 1000, no. 1, p. 012121. IOP Publishing, 2018.

[4]. Chukwunonso, Franklyn, Roliana Binti Ibrahim, Ali Bin Selamat, Adamu Idama, and Wadzani A. Gadzama. "The impact of the Internet and World Wide Web on distance and collaborative learning." ICCGI 2013 (2013): 7-15.

[5]. Almasri, Abdullah, and Luis Gouveia. "S3: A Study on the Efficiency of Current Power-Saving Approaches Used Among Android-Application Development & Usage Stages." International Journal of Computing and Digital Systems 9, no. 4 (2020): 725-733.

[6]. Rohini, G., and A. Srinivasan. "Dynamic Transition of Bandwidth and Power Saving Mechanism to Support Multimedia Streaming Using H. 264/SVC over the Wireless Networks." International Journal of Computer Networks and Applications 2, no. 2 (2015): 57-63.

[7]. Almasri, Abdullah, and Luis Borges Gouveia. "Reviewing the efficiency of current power-saving approaches used among different stages of an Android-application lifecycle." Internal Report* TRS 05/2019 (2019).

[8]. Kamble, Atul V., Sandeep S. Bidwai, and Varsha B. Chougule. "AUTOMATION AND POWER SAVING OF HOME USING ANDROID MOBILE." IJRET, eISSN: 2319-1163.

[9]. Sameh, Ahmed, and Abdulla Al-Masri. "Smartphone preventive customized power saving modes." International Journal of UbiComp, IJU 8, no. 1 (2017): 1-15.

[10]. Almasri, Abdullah, and Luis Borges Gouveia. "Adding Energy Star Rating Schema to Android Applications on Google Play Store an Example of a Preventive Power Saving Model." Internal Report* TRS 06/2019. Technology, Networks and Society Group (2019).

[11]. ALMASRI, ABDULLAH, and LUIS GOUVEIA. "Investigating the Amount of Power Consumed by Power-Optimizing Applications on Android Smartphones." (2020).

[12]. Kosjer, Vladimir, Dragan Mitić, Aleksandar Lebl, Vladimir Matić, and Žarko Markov. "Power saving technique based on traffic channels reallocation in gsm mobile network." Journal of Electrical Engineering 71, no. 2 (2020): 103-109.

[13]. Lee, Jae-Beom, Myeongjin Kim, and Eui-Young Chung. "Schedule-aware DVFS Algorithm on Android Platforms for Energy Minimization." In The 29th International Technical Conference on Circuit/Systems Computers and Communications (ITC-CSCC), Phuket, Thailand. 2014.

[14]. Thimmarayaswamy, K., Mary M. Dsouza, and G. Varaprasad. "Low Power Techniques for an Android Based Phone." Computer Architecture News 40, no. 2 (2012): 26.
[15]. Al-Hayanni, Mohammed A. Noaman, Ashur Rafiev, Fei Xia, Rishad Shafik, Alexander Romanovsky, and Alex Yakovlev. "PARMA: Parallelization-aware run-time management for energy-efficient many-core systems." IEEE Transactions on Computers 69, no. 10 (2020): 1507-1518.
[16]. Lee, Wen-Shiung, Alex YiHou Huang, Yong-Yang Chang, and Chiao-Ming Cheng. "Analysis of decision making factors for equity investment by DEMATEL and Analytic Network Process." Expert Systems with Applications 38, no. 7 (2011): 8375-8383.
[17]. Tsai, Wen-Hsien, and Wen-Chin Chou. "Selecting management systems for sustainable development in SMEs: A novel hybrid model based on DEMATEL, ANP, and ZOGP." Expert systems with applications 36, no. 2 (2009): 1444-1458.
[18]. Kumar, Ashwani, and Gaurav Dixit. "An analysis of barriers affecting the implementation of e-waste management practices in India: A novel ISM-DEMATEL approach." Sustainable Production and Consumption 14 (2018): 36-52.
[19]. Si, Sheng-Li, Xiao-Yue You, Hu-Chen Liu, and Ping Zhang. "DEMATEL technique: A systematic review of the state-of-the-art literature on methodologies and applications." Mathematical Problems in Engineering 2018 (2018).
[20]. Yazdi, Mohammad, Faisal Khan, RouzbehAbbassi, and RiszaRusli. "Improved DEMATEL methodology for effective safety management decision-making." Safety science 127 (2020): 104705.
[21]. Zhang, Weiquan, and Yong Deng. "Combining conflicting evidence using the DEMATEL method." Soft computing 23, no. 17 (2019): 8207-8216.
[22]. Lee, Hsuan-Shih, Gwo-HshiungTzeng, WeichungYeih, Yu-Jie Wang, and Shing-Chih Yang. "Revised DEMATEL: resolving the infeasibility of DEMATEL." Applied Mathematical Modelling 37, no. 10-11.

