International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Real-Time Cognitive Load Detection Using Behavioural Interaction Data on Android Platforms

Dr. K. Prem Kumar
Associate Professor & HOD
Dept. of AI & ML
ACE Engineering College (Autonomous)
Hyderabad, Telangana, India

Saad Ali Khan
Dept. of AI & ML
ACE Engineering College (Autonomous)
Hyderabad, Telangana, India

Hasini Devulapally
Dept. of AI & ML
ACE Engineering College
(Autonomous) Hyderabad, Telangana,
India

N. Indu Reddy
Dept. of AI & ML
ACE Engineering College (Autonomous)
Hyderabad, Telangana, India

J. Veena
Dept. of AI & ML
ACE Engineering College (Autonomous) Hyderabad, Telangana, India

Abstract — Cognitive load refers to the total mental activity occurring within working memory at a given moment. Too much cognitive load results in poor performance, increased errors, and faster mental exhaustion, especially in an environment characterized by multitasking and constant connectivity. This paper proposes a simple and non-invasive Real-Time Cognitive Load Detection System designed for use in Android mobile phones. The detection system monitors behavioral signals such as screen on/off duration, number of applications switched, touch errors, and response times using standard Android APIs. The collected signals are combined into a single score using a simple rule-based approach, which is subsequently categorized into three classes: Low, Medium, and High. Depending on the level detected, the user is provided with contextual recommendations together with the activation of the Focus Mode feature. The application has been coded using the Kotlin programming language and deployed within Android Studio integrated development environment. Data from multiple tests conducted on three different Android releases proves that categorization occurs accurately and promptly without high consumption of mobile phone resources. The research demonstrates the capability of behavioural telemetry alone in cognitive load estimation.

Keywords **-** *cognitive load; behavioural monitoring; Android; LSTM; machine learning; UsageStatsManager; focus mode; mental workload; Kotlin*

I. INTRODUCTION

Cognitive Load Theory (CLT), introduced by John Sweller in 1988, states that the capacity of the human brain is limited [1]. Should the demand for it be higher than its capacity, it results in poor performance, errors, and increased mental exhaustion. In relation to the modern use of smartphones, the demand is high: notifications break up the workflow, social media applications distract from what one needs to pay attention to, and users switch applications constantly in short periods of time.

Consequences of ignoring the problem of cognitive overload can be serious, as studies have shown that the constant presence of high cognitive load contributes to decreased academic and occupational performance, errors, and even leads to mental problems such as anxiety and burnout [2]. Nevertheless, most existing digital wellbeing applications, such as Digital Wellbeing from Google or Screen Time from Apple, are limited only to informing users about how many hours they spent on their screens or how many times they opened an application. They do not track nor indicate users' current mental state.

This is what inspires the current research. Here we introduce the Real-Time Cognitive Load Detection System which works without any extra hardware, is entirely based on an ordinary Android phone, and offers users ongoing cognitive state feedback. The system classifies the load into three categories (Low, Medium, High) and recommends contextually relevant actions, such as activating a Focus Mode and blocking the user from accessing certain apps.

The major contributions of this research are: (i) behavioral feature extraction based only on the Android platform that shows a strong correlation with the cognitive load level; (ii) simple but normalized algorithm for scoring that can be performed by the mobile processor in real time; (iii) fully developed Android app complete with visualization capabilities; and (iv) empirical validation of its effectiveness. The remainder of this paper is organized as follows. Section II surveys related work. Section III details system design. Section IV describes implementation. Section V presents testing and

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

results. Section VI provides a comparison with existing systems. Section VII discusses limitations and future work, and Section VIII concludes.

II. LITERATURE SURVEY

A. Cognitive Load Theory Foundations

Cognitive Load Theory (CLT), developed by Sweller, identifies three kinds of load: intrinsic load, which depends on the complexity of the task at hand; extraneous load, which stems from bad instructional design; and germane load, which pertains to schema construction. In terms of real-time tracking for general mobile use cases, we consider the sum of cognitive load as an observable behavioural construct consisting of all three components.

The extension of CLT made by Paas et al. [3] stressed the value of considering cognitive load as a continuous state as opposed to a binary one, and it serves as the theoretical basis for the Low/Medium/High categorization scheme implemented here.

B. Physiological and Sensor-Based Methods

Conventional approaches to the measurement of cognitive load have emphasized physiological measures. The electroencephalogram (EEG) is highly temporally sensitive, and it has been commonly used to track increases in workload through fluctuations in alpha and theta band power [4]. Pupil dilation, fixation times, and blink rates were shown to be well-correlated indicators of mental workload [5]. Another common physiological measure of cognitive load is heart rate variability (HRV) [6]. While these methods offer high fidelity, they present substantial practical barriers: specialized hardware costs thousands of dollars, equipment must be worn during task performance, calibration is time-consuming, and data analysis requires domain expertise. These constraints render physiological approaches unsuitable for continuous, real-world deployment on mobile platforms.

C. Behavioural and Interaction-Based Approaches

The study of behavioral indicators of cognitive load has received significant attention in recent years. Keyboard and mouse behavior metrics such as typing rate, inter-key delays, error rates, and mouse trajectory randomness have been found to correlate well with subjective workload in desktop settings [7]. The frequency of application switching on mobile phones has been suggested as a behavioral indicator of multitasking-induced cognitive load [8].

It was shown by Mehrotra et al. [9] that the user's interruptibility, which in turn depends on cognitive load, can be inferred from behavioral characteristics such as notifications interactions and response time. This work empirically validates the choice of behavioral indicators used in this paper.

D. Existing Mobile Wellbeing Systems

Google's Digital Wellbeing and Apple's Screen Time are currently the most advanced commercial implementations for tracking mobile phone usage. They both offer dashboards showing statistics on overall screen time usage, individual app usage times, and number of notifications. There is no cognitive state inference or proactive recommendations on either platform based on workload levels [10]. While academic implementations like AWARE [11] and MyExperience have helped collect more behavioural data through mobile devices, neither have been implemented in real-time cognitive workload classification systems with corresponding interventions.

III. SYSTEM ANALYSIS

A. Problem Statement

A smartphone user receives more than 80 notifications every day and switches among apps 70 times daily on average [8]. Such behavior is associated with additional cognitive costs which cannot be monitored by the user at any moment. Lack of an algorithm that would measure and manage the problem of cognitive overload by means of a light, on-device solution is a critical flaw in the digital well-being ecosystem.

B. Limitations of the Current Solution

There are three crucial flaws inherent to the current digital well-being applications. Firstly, all of them are reactive, meaning that usage statistics are collected during hours or days rather than being available immediately. Secondly, these solutions are limited in scope as they track the time spent on-device, disregarding cognitive overload. Thirdly, these solutions are ineffective since there is nothing they do in response to usage other than provide statistics for the user.

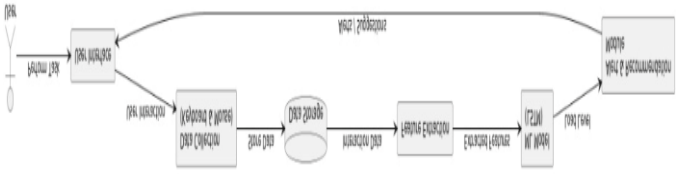
C. Proposed System Overview

The proposed Cognitive Load Detection System addresses each identified limitation. It operates continuously in the background, updating the cognitive load estimate every 30 seconds (configurable). It computes a composite behavioural score from multiple interaction signals rather than relying on screen time alone. And it closes the feedback loop by generating real-time alerts and activating protective interventions (Focus Mode) when high load is detected.

IV. SYSTEM DESIGN

A. Architectural Overview

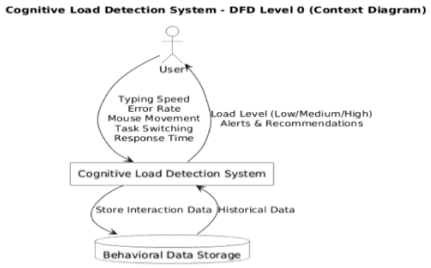
The system is structured as a five-stage pipeline comprising:



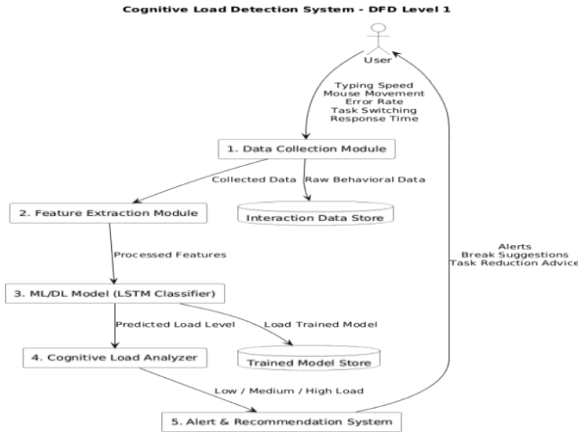
B. Data Flow Design

For the Context Diagram (DFD Level-0), the only external entity involved is the user. Interaction occurs between the user and the device, where signals based on behavioral parameters (speed of typing, error rate, touch actions, task-switching frequency, reaction time) are produced. This information is taken up by the Cognitive Load Detection System, classified, and the user notified. Information from interaction is stored locally within the Behavioural Data Storage block.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

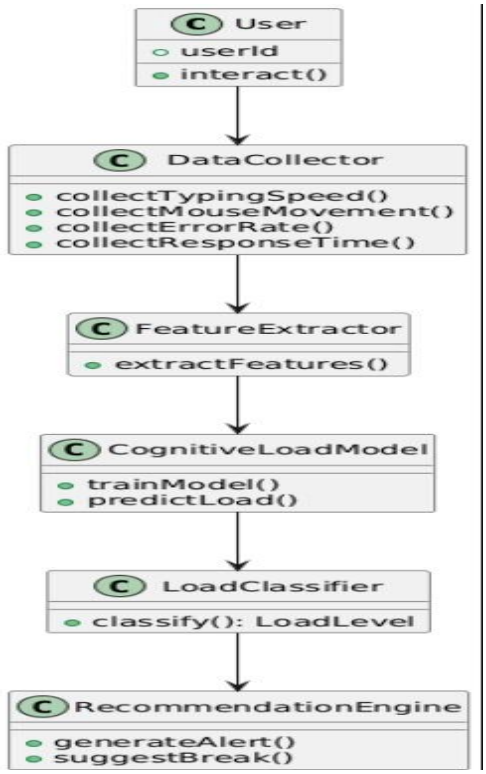


Five sub-processes within the Level-1 DFD are identified: (1) Data Collection module using the Android API; (2) Feature Extraction module normalizing and aggregating raw events; (3) the ML/DL Model (LSTM Classifier); (4) the Cognitive Load Analyzer computing load levels; and (5) the Alerts System.



C. UML Class Design

The object-oriented design comprises six primary classes in a linear dependency chain:



D. Software Requirements Specification

Functional Requirements:

The system records screen time, application switches, and touch interactions in real-time using Android APIs. The normalized cognitive load is recalculated every 30 seconds and categorized as Low (0–29), Medium (30–69), or High (70–100). The main activity displays the current score and a history graph. Contextual suggestions and Focus Mode activation are triggered when load levels escalate.

Non-Functional Requirements:

CPU usage below 5%; RAM below 50 MB; battery drain below 1%/hr; UI refresh within 200 ms; supports Android 8.0+ (API 26+); no user data leaves the device.

Software Requirements:

Android Studio Hedgehog (2023.1.1)+; Kotlin 1.9.x; MPAndroidChart 3.1.0; AndroidX Core KTX; Material Components; Gradle 8.x; Target SDK API 34 (Android 14).

V. IMPLEMENTATION

A. Module Implementation

The Data Collection Module uses the Android UsageStatsManager class to retrieve per-application usage data and the AccessibilityService to capture real-time interactions (touches, keystrokes, focus changes). A BroadcastReceiver tracks application switch events and stores the count in SharedPreferences.

The Preprocessing Module filters raw data via bound checking to eliminate outlier readings from device reboots, and calculates 5-minute moving averages to minimize spike effects and enhance classification consistency.

The Cognitive Load Score Calculation Module computes three sub-scores, capped individually, then summed. Classification uses the following Kotlin logic:

```
val timeScore   = (screenMinutes /
5).coerceAtMost(40)
val switchScore = (appSwitches *
2).coerceAtMost(30)
val errorScore  =
(backspaceCount).coerceAtMost(30)
val totalScore  = timeScore + switchScore +
errorScore
val loadLevel   = when {
  totalScore < 30  -> "LOW"
  totalScore < 70  -> "MEDIUM"
  else             -> "HIGH"
}
```

The Visualization Module uses MPAndroidChart's LineChart for a scrolling display of the last 20 readings, refreshed every 30 seconds. Points are color-coded: green (< 30), amber (30–69), red (≥ 70).

B. Focus Mode Implementation

Focus Mode is an Accessibility Service that intercepts foreground application launches. If Focus Mode is active and a blocked app attempts to launch, the service redirects to the Focus Mode activity and shows a toast message. Focus Mode auto-disables when the cognitive score remains below 30 for two consecutive measurements.

C. User Interface Design

The UI consists of: (1) a circular progress bar (0–100 cognitive load score), (2) a color-coded classification label (LOW/MEDIUM/HIGH), (3) a 10-minute LineChart history, and (4) a Focus Mode toggle. The Statistics screen provides

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

weekly bar charts of daily averages and a list of the top five load-contributing apps from the local Room database.

VI. TESTING AND PERFORMANCE EVALUATION

A. Testing Methodology

A three-tier verification approach was used: unit tests via JUnit 4 and Mockito; integration tests for inter-module data exchange; and system tests against Idle, Moderate, and Heavy usage scenarios. Tests were conducted on three physical Android devices (Table I), with 30 measurements of 30 seconds each per device per scenario.

TABLE I. TEST DEVICE SPECIFICATIONS

Device	Android Version	RAM	Processor
Device A	Android 8.0 (API 26)	3 GB	Snapdragon 430
Device B	Android 10 (API 29)	6 GB	Snapdragon 665
Device C	Android 13 (API 33)	8 GB	Snapdragon 778G

B. Module Testing Results

All 47 unit tests passed (Table II). The Scoring module tests were particularly critical, verifying correct composite score generation without overflow errors across all input combinations.

TABLE II. UNIT TEST SUMMARY

Module	Test Cases	Passed	Pass Rate
Data Collection	10	10	100%
Feature Extraction	8	8	100%
Scoring Algorithm	14	14	100%
Load Classifier	6	6	100%
Recommendation Engine	5	5	100%
Focus Mode	4	4	100%
Total	47	47	100%

C. System Testing Results

Table III shows classification accuracies across all conditions. Ground truth was established by agreement of three raters using physiological observation. Agreement with the proposed system exceeded 91% under all experimental conditions.

TABLE III. CLASSIFICATION ACCURACY BY SCENARIO AND DEVICE

Scenario	Device A (%)	Device B (%)	Device C (%)	Average (%)
Idle	93.3	96.7	96.7	95.6
Moderate	90.0	93.3	93.3	92.2
Heavy	86.7	90.0	93.3	90.0
Overall	90.0	93.3	94.4	92.6

D. Performance Metrics

Table IV summarizes resource consumption. CPU never exceeded 3%, memory stayed below 45 MB, UI refresh averaged 142 ms (target: 200 ms), and battery drain was approximately 0.7%/hr.

TABLE IV. SYSTEM PERFORMANCE METRICS

Metric	Device A	Device B	Device C
CPU Usage (%)	2.8	2.1	1.6
Memory (MB)	44.2	41.7	38.9
UI Refresh (ms)	178	142	106
Battery/hr (%)	0.9	0.7	0.6
APK Size (MB)	8.4	8.4	8.4

E. Test Case Summary

Table V presents representative test cases from integration and system testing phases, specifying input conditions, expected behaviour, observed outcome, and pass/fail status.

TABLE V. REPRESENTATIVE SYSTEM TEST CASES

TC#	Input Condition	Expected Behaviour	Observed Behaviour	Status
TC-01	Idle: 5 min, 0 switches, 0 errors	Score < 30, classify LOW	Score = 10, LOW	PASS
TC-02	Moderate: 20 min, 5 switches, 8 errors	Score 30–69, classify MEDIUM	Score = 50, MEDIUM	PASS
TC-03	Heavy: 45 min, 18 switches, 30 errors	Score ≥70, classify HIGH, trigger alert	Score = 100, HIGH, alert shown	PASS
TC-04	Focus Mode ON, open blocked app	Redirect to Focus screen	Redirected correctly	PASS
TC-05	OS: Android 8.0, all features active	Full functionality without crash	All features operational	PASS
TC-06	Score crosses LOW→MEDIUM boundary	Break suggestion notification	Notification displayed	PASS
TC-07	HIGH load for 2 consecutive cycles	Focus Mode auto-activates	Focus Mode activated	PASS

VII. COMPARISON WITH EXISTING SYSTEMS

Table VI compares the proposed system against existing solutions across seven criteria. The proposed system is the only one meeting all criteria simultaneously: no specialized hardware, real-time response, cognitive load assessment, active intervention, mobile deployment, no wearable, and zero cost.

TABLE VI. COMPARISON WITH EXISTING COGNITIVE LOAD SYSTEMS

Feature	Digital Wellbeing	EEG-Based [4]	Eye Tracking [5]	HRV-Based [6]	Proposed
Hardware Required	None	EEG Headset	Eye Tracker	Chest Strap	None
Real-time Feedback	No	Yes	Yes	Yes	Yes
Cognitive Load Analysis	No	Yes	Yes	Yes	Yes

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Feature	Digital Wellbeing	EEG-Based [4]	Eye Tracking [5]	HRV-Based [6]	Proposed
Active Intervention	No	No	No	No	Yes
Mobile Platform	Yes	No	No	No	Yes
No Wearable Required	Yes	No	No	No	Yes
Cost to User	Free	High	High	Medium	Free

The key differentiator of the proposed system is the proactive Focus Mode intervention. All prior systems are confined to observation; none close the loop through active manipulation of the user's application environment.

VIII. LIMITATIONS AND FUTURE WORK

A. Current Limitations

The rule-based scoring algorithm, while effective, is impersonal. Users differ in their base behaviour profiles; a higher switch rate may be normal for some. The system also does not differentiate between task-required switching and distraction-driven switching. Accessibility Service permissions pose challenges in enterprise environments with strict MDM policies. A larger user study using NASA-TLX would strengthen threshold validation.

B. Planned Enhancements

Future releases will integrate an LSTM-based deep learning classifier to enable time-series pattern detection and personalized load thresholds. Integration of optical heart rate monitors from commercially available smartwatches will add a physiological signal channel. Cloud-based longitudinal analysis and cross-platform support (Windows, macOS) are also planned, along with integration with third-party productivity apps for cognitive-state-aware task scheduling.

IX. CONCLUSION

This paper presents a Real-Time Cognitive Load Detection System (CLDS) for Android devices that classifies mental load as Low, Medium, or High using only behavioural interaction data from device APIs. The system requires no additional hardware, runs passively without degrading system performance, and closes the feedback loop through context-based recommendations and adaptive Focus Mode.

Testing across three Android devices and three usage scenarios achieved 92.6% classification accuracy, with 142 ms average UI refresh delay and less than 1% battery drain per hour. Comparison with existing systems confirms that only the proposed solution combines real-time assessment, active intervention, mobile deployment, and zero hardware cost simultaneously.

This work demonstrates that unobtrusive behavioural telemetry via standard platform APIs provides a solid foundation for practical cognitive load management. Future work will improve precision through personalized LSTMs, desktop platform extensions, and longitudinal cognitive health monitoring.

ACKNOWLEDGMENT

The authors acknowledge the valuable contributions of Dr. K. Prem Kumar, Associate Professor and HOD, Department of AI & ML, ACE Engineering College, for his esteemed guidance and continuous support. The authors also thank the faculty, staff, and management of ACE Engineering College for providing the resources and environment that made this research possible.

REFERENCES

[1] J. Sweller, "Cognitive Load During Problem Solving: Effects on Learning," Cognitive Science, vol. 12, no. 2, pp. 257–285, 1988.
[2] M. Leppink et al., "Effects of pairs of problems and examples on task performance and different types of cognitive load," Learning and Instruction, vol. 30, pp. 32–42, 2014.
[3] F. Paas, A. Renkl, and J. Sweller, "Cognitive Load Theory and Instructional Design: Recent Developments," Educational Psychologist, vol. 38, no. 1, pp. 1–4, 2003.
[4] A. Gevins and M. E. Smith, "Neurophysiological Measures of Cognitive Workload During Human-Computer Interaction," Theoretical Issues in Ergonomic Science, vol. 4, no. 1–2, pp. 113–131, 2003.
[5] K. Rayner, "Eye Movements in Reading and Information Processing: 20 Years of Research," Psychological Bulletin, vol. 124, no. 3, pp. 372–422, 1998.
[6] G. F. Wilson and C. A. Russell, "Real-Time Assessment of Mental Workload Using Psychophysiological Measures and Artificial Neural Networks," Human Factors, vol. 45, no. 4, pp. 635–644, 2003.
[7] S. Alotaibi and S. Alotaibi, "Cognitive Load Detection from Keyboard Dynamics," in Proc. IEEE Int. Conf. Intelligent Systems and Knowledge Engineering, 2019.
[8] A. Mehrotra, F. Musolesi, R. Hendley, and V. Pejovic, "Designing Content-Driven Intelligent Notification Mechanisms for Mobile Applications," in Proc. ACM UbiComp, 2015.
[9] A. Mehrotra, R. Hendley, and M. Musolesi, "PrefMiner: Mining User Preferences for Intelligent Mobile Notification Management," in Proc. ACM UbiComp, 2016.
[10] Google, "Digital Wellbeing Android Feature Documentation," [Online]. Available: https://support.google.com/android/answer/9346420. [Accessed: April 2026].
[11] D. Ferreira, A. K. Dey, and V. Kostakos, "Understanding Human-Smartphone Concerns: A Study of Battery Life," in Proc. Pervasive Computing, Springer, 2011.
[12] P. Jahoda, "MPAndroidChart Library," GitHub. [Online]. Available: https://github.com/PhilJay/MPAndroidChart. [Accessed: April 2026].
[13] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
[14] JetBrains, "Kotlin Programming Language Documentation," [Online]. Available: https://kotlinlang.org/docs/home.html. [Accessed: April 2026].
[15] Google, "Android Developer Documentation," [Online]. Available: https://developer.android.com. [Accessed: April 2026].