JOURNAL OF ENGINEERING RESEARCH & SCIENCES

JENRS

www.jenrs.com
ISSN: 2831-4085

Volume 5 Issue 2
Fenruary 2026

EDITORIAL BOARD

Editor-in-Chief

Editorial Board Members

Editorial

The *Journal of Engineering Research and Sciences (JENRS)* is pleased to present a collection of research contributions that address contemporary challenges in agricultural automation, software reliability engineering, and neurotechnology-assisted healthcare. These studies showcase the growing integration of artificial intelligence, advanced sensing technologies, and data-driven methodologies to improve productivity, reliability, and human well-being. The research published in this issue reflects the multidisciplinary nature of modern engineering and highlights innovative solutions with significant practical and societal impact.

Advancements in agricultural automation are transforming traditional farming practices through the adoption of intelligent robotic systems. One contribution presents an AI-powered robotic solution for autonomous date harvesting, integrating computer vision, LiDAR sensing, and robotic manipulation technologies. By combining YOLO-based object detection with convolutional neural networks for fruit maturity classification, the proposed system enables accurate identification and harvesting of ripe dates while minimizing fruit damage. The study demonstrates the potential of intelligent harvesting systems to reduce labor dependency, improve crop quality, and support sustainable agricultural development, particularly in regions where date cultivation plays a significant economic role [1].

Software quality assurance remains a critical concern in large-scale enterprise environments where test automation forms the backbone of continuous integration and continuous deployment pipelines. A comprehensive investigation into test flakiness explores the underlying causes of unstable automated tests across user interface, application programming interface, mobile, and data-processing domains. The proposed multi-level stability framework incorporates robust synchronization mechanisms, deterministic locator strategies, resilient API validation, controlled test-data management, and cloud-based reliability patterns. Enhanced by AI-driven analytics for flakiness prediction and diagnosis, the framework demonstrates substantial reductions in unstable test executions and diagnostic effort, providing a practical roadmap for improving software reliability and operational efficiency in enterprise-scale systems [2].

Improving cognitive performance and attention among children affected by neurological disorders represents an important area of healthcare research. A study investigating the application of brainwave entrainment techniques combines virtual reality, binaural audio stimulation, and electroencephalogram analysis to evaluate their effectiveness in enhancing attentional learning. Using audio-visual entrainment at a frequency of 10 Hz, the research examines neurophysiological changes among children diagnosed with Attention Deficit Hyperactivity Disorder, Autism Spectrum Disorder, and comorbid conditions. Quantitative analysis of EEG signals reveals measurable improvements in attention and cognitive functioning among participants with ADHD and comorbid conditions, while more limited effects are observed among individuals with ASD. The findings contribute valuable insights into the potential role of non-invasive neurotechnology interventions in supporting cognitive development and therapeutic outcomes for children with neurological disorders [3].

Collectively, the studies featured in this issue demonstrate the transformative influence of artificial intelligence, intelligent automation, and advanced analytical techniques across diverse sectors. From sustainable agricultural practices and dependable software engineering to innovative healthcare interventions, these contributions advance both theoretical understanding and practical implementation. It is anticipated that the findings presented herein will encourage further interdisciplinary research and inspire the development of technologies that address pressing societal and industrial challenges in an increasingly connected world.

References:

[1] H.H.A. Adlan, R. Al Zamanan, L. Almufleh, T. Almuqrin, J. Alhassoun, "Harnessing the Power of Machine Learning and Sensor Detection in a Simulation for the Design of Smart Date Harvesting Robot," *Journal of Engineering Research and Sciences*, vol. 5, no. 2, pp. 1–8, 2026, doi:10.55708/js0502001.

[2] S.K. Tiwari, "How to Fix Automation Flakiness: Root Causes and Enterprise-Level Solutions," *Journal of Engineering Research and Sciences*, vol. 5, no. 2, pp. 9–23, 2026, doi:10.55708/js0502002.

[3] M. Mandapati, P. Ranjan, "Impact of Brainwave Entrainment using VR to Improve Attentional Learning in Children with ADHD, ASD and Comorbidity," *Journal of Engineering Research and Sciences*, vol. 5, no. 2, pp. 24–35, 2026, doi:10.55708/js0502003.

Editor-in-chief

Dr. Jinhua Xiao

JOURNAL OF ENGINEERING RESEARCH AND SCIENCES

Volume 5 Issue 2 **February 2026**

CONTENTS

JENRS

Received: 07 January 2026, Revised: 12 February 2026, Accepted: 14 February 2026, Online: 22 February 2026

DOI: https://doi.org/10.55708/js0502001

Harnessing the Power of Machine Learning and Sensor Detection in a Simulation for the Design of Smart Date Harvesting Robot

Hanan Hassan Ali Adlan, Reham Al Zamanan, Leen Almufleh, Tala Almuqrin, Jory Alhassoun

Princess Nourah bint Abdul Rahman University, Computer Science Department, P.O. Box 84428, Riyadh, 11671, Saudi Arabia

*Corresponding author: Hanan Hassan Ali Adlan, Dept. of Computer Science; Faculty of Computer and Information Sciences; PNU, KSA

hahassan@pnu.edu.sa, mehanan14@gmail.com

ABSTRACT: The Traditional date harvesting is labor-intensive and inefficient, leading to losses and quality issues. This paper introduces an AI-powered robotic system that automates date harvesting using computer vision, LiDAR sensors, and a robotic arm with a suction mechanism. The robot is capable of perception, it detects, classifies, and harvests ripe dates autonomously, ensuring minimal damage and improved efficiency. The work develops a CNN architecture and a YOLO model. Propose using sensors for detection purposes. By integrating YOLO for object detection and CNN for maturity classification, the system optimizes harvesting decisions. This solution is simulated and is supposed to enhance productivity, reduces costs, and improves date quality, contributing to empowering the agriculture sector with powerful tool that encourage date planting, nourish life through economic investment in dates, provide advancement in agricultural automation, and contributing to sustain date planting.

KEYWORDS: Machine Learning, Convolution Neural Network, YOLO, LiDAR

1. Introduction

The Kingdom of Saudi Arabia is distinguished in the production and export of dates; it ranked second as one of the largest countries in the world producing dates [1]. Dates are an essential part of Saudi culture and are considered a strategic national product and one of the most important food and economic sources [2].

Manual harvesting is the prevailing method used, it requires physical strength to climb and high endurance to work under the sun. This traditional method currently practiced in the field of date harvesting influences the date production cycle negatively. Results in wasted dates and time.

Automated harvesting is considered essential in the field of date harvesting. The transition from traditional harvesting to automated harvesting provides ease and speed in harvesting fruits and provides a fundamental solution to the problem of delayed harvesting that affects the date market [3, 4, 5]. In the traditional date harvesting process, farmers face major obstacles that lead to huge losses, such as high costs resulting from relying on many workers, in addition to the risks that threaten the safety of workers due to working at high heights under the sun, and waste resulting from human errors during harvesting or transporting dates [6]. Automatic harvesting using advanced algorithms and artificial intelligence proof effectiveness and efficiency. Computer vision. Such as YOLO (You Only Look Once) for real-time object detection [7, 8]. Convolutional neural network (CNN) algorithm, used to identify maturity of dates and their readiness for harvest [9, 10, 11]. LiDAR technology is found to be efficient in determining dimensions and distances of harvest sites. Supported by a computer vision unit for image analysis, improves accuracy of date recognition and harvest location determination [11, 12]. Vision systems based on CNN was found to be efficient, where it reached 99.1% recognition rate [13]. Robots are efficient in harvesting; many were developed for different fruits. Tomato [14], cucumber [15], strawberry [16], and palm oil [17], to name a few.

2. Leena Robot

Harvesting robots can be utilized to collect, process and analyses individual spatiotemporal data for targeted date type ripening time, harvest time, date status, date

defects in addition to environmental information related to farms and palm trees. These contribute to harvesting the largest possible amount of healthy dates free of damage. The presence of robots in the harvesting field is vital and encourage investment in this field. In this paper a proposed robot for harvesting dates is suggested.

The main parts of the date harvesting robot include Sensory system consisting of vision system, distance sensors, temperature and humidity Unit, and robotic arms for sucking the dates. Data processing and decision-making systems are applied to harvest dates with the inclusion of artificial intelligence algorithms.

Leena is a robot using arm with tube for sucking dates. Incorporating Camera and sensors that enable capture dates images to determine maturity of the date for harvest. Other functionalities include moving around and locating the palm. The robot is designed with the aim, Towards efficient Date Harvesting. The design objectives are to reduce the labor intensity, cost, and inefficiency of manual harvesting. The robot is designed and implemented on simulation. The field environment with palms is created, and the robot move around, locate the palm, detect the dates, classify its maturity level, then harvest the dates. The scope of the project is limited to the harvesting of the dates. Other functionalities of implementing the storage unit is beyond the scope of the current implementation. The design of the robot incorporates the following functional requirements:

- Develop a database of date bunch types and weights at different maturity levels to support automation in the date palm application.
- Move around the date farm to locate palm trees.
- Detect the presence of dates and distinguish the dates from other elements in scenes (e.g trunk, leaves, ground, and sky).
- Capture images of dates to determine the maturity of the fruits.
- Determine the maturity of the dates for decision making.

3. Methodology

Leena robot integrates hardware and software. Design of the hardware was performed on a simulation environment. Hypothetical hardware mechanics is proposed to enable the functional requirement and are simulated on webots (webots simulator is an open source, cross-platform robotics simulation environment) and Blender for Palm trees.

LiDAR technology enables accurately determine dimensions and distances of the harvest sites. The principle of its operation is through a pulsed laser to calculate the variable distances of an object from the ground surface. It measures the time it takes to return after bouncing off a surface. These light pulses, together with the information collected, generate accurate three-dimensional details on the ground surface and the target object. Supported by a computer vision unit for image analysis, improves the accuracy of date recognition and harvest location determination.

The Software component develops the system make use of YOLO (You Only Look Once). Yolov8 is one of the Yolo model series, is an AI model used to detect, classify, and track objects within images or videos it is used to distinguish dates from the stem and other unimportant parts in the image. After Yolov8 locates dates in the images, it sends them to a CNN module to classify the dates into mature and immature.

Convolution Neural Networks (CNN) is a class of neural networks designed for data processing and is particularly suitable for image recognition and processing tasks [18]. It is integrated in the system as part of the computer vision system, where it undertook the task of processing the data set and classifying it into ripe Barhi dates (which are pickable) and unripe Barhi dates (which are not pickable).

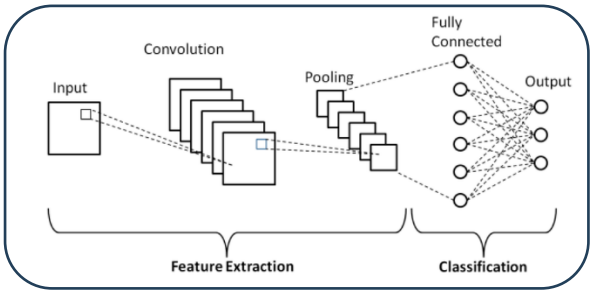

Figure 1: CNN Model

Figure 1 depicts the main components of a general CNN model. It consists of convolutional layers, pooling layers, and fully connected layers. It proved success in identifying the maturity of dates and their readiness for harvest .

4. System Architecture and the Robot Structure

The architectural components of the robot system are given Figure 2.

Figure 2 shows the architecture of the date harvesting robot. It illustrates the main components of the robot as follows:

Control Panel: Provides the user interface that enable operating and control of the robot. The administrator

(farmer) can send commands such as starting the robot, starting the harvesting process, or unloading the dates from the warehouse. This is connected to the control unit.

Control Unit: The central node of the system, manages coordination between the various components. It receives commands from the control panel and distributes instructions to the other components. It is connected to the mobility unit. If the farmer initiates the harvesting process from the control panel, the control unit will send a signal to the motion unit to begin its work.

Mobility Unit: This unit is responsible for the robot's movement within the farm and among the palm trees. It receives commands from the control unit to initiate its movement. It contains sensors to suit working in challenging agricultural environments.

The robotic arm: This is the component for reach for the date. It uses special camera to locate the date and capture images for processing to if the date is confirmed to be ripe. This process is controlled by the control unit. The control unit transmits the date's ripeness status to the hand. Based on the ripeness status, the hand picks the date.

The camera: Capture images of the dates for processing their ripeness percentage. It is mounted in the robot's hand and takes pictures of the dates on the trees. These images are sent to the integrated artificial intelligence module to analyze the degree of ripeness and classify the date into ripe or unripe.

The storage unit: This is the unit where the picked dates are stored until they are unloaded. It is connected to a display panel that displays the storage unit's status (full or empty), along with other storage status details.

The architectural components communicate via internal links, ensuring real-time and accurate information exchange and achieving synchronization of operations.

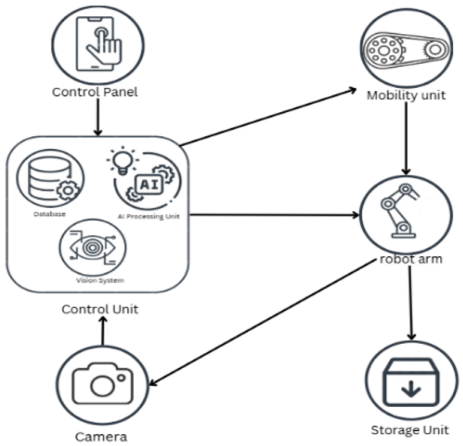

Figure 2: System Architecture

Figure 3 shows hypothetical mechanical parts of the robot, which play pivotal roles in performing the required tasks

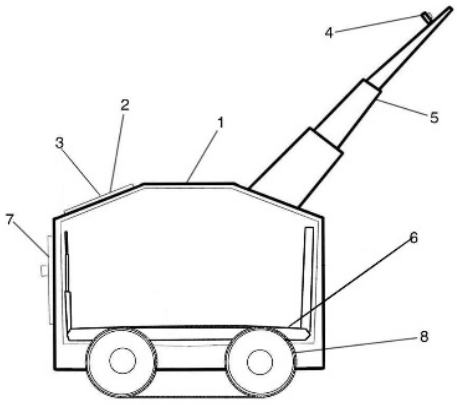

Figure 3: Hypothetical Structural Design of the Robot

In Figure 3 the numbers indicate the following functionalities:

1. Basic structure: The structure on which the robot design is based, as it supports all components and allows them to be fixed firmly and effectively.
2. Control screen: The control screen allows the worker to monitor and control the robot, and allows changing the language to facilitate control of the robot by different users.
3. Start button on screen: Operates the robot and prepare it to be ready for the harvesting process.
4. Camera: To identify dates and determine whether the date is ripe or not to harvest.
5. Robotic arm: This arm is a tube with attached camera used to move to capture date images and suck the ripe date and pass it to the storage box.
6. Storage box: The storage box is carefully designed storage unit, used to protect the dates during the harvesting process. It is made of spongy surface that provides protection for the dates so that they do not deform during the fall, and helps maintain their quality.
7. Unloading dates door: A back door used to unload the storage box to get the harvested dates, this door open automatically allowing the surface to descend to drop the dates into a container.
8. Tracked wheels: Crawler wheels used to enable the robot to move, allowing it to walk on rough surfaces, aims to enhance ability to navigate farms and uneven lands.

5. Implementation and Results

The robot is developed on webots simulator. Initial prototypes is displayed in Figures 4 and Figure 5 respectively.

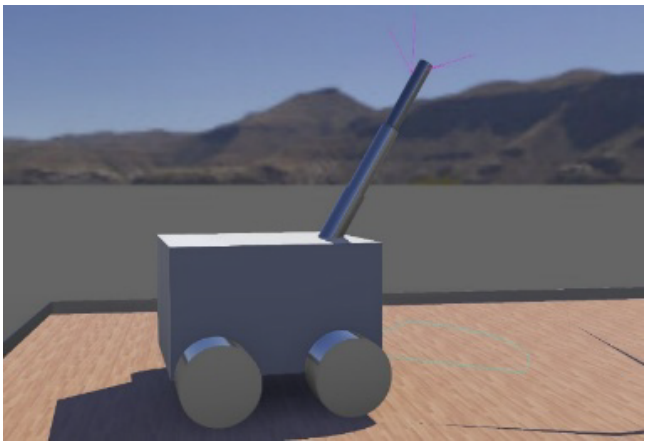

Figure 4: robot Side simulation

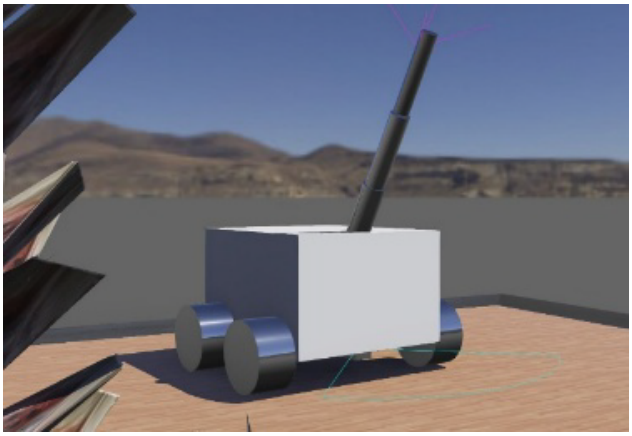

Figure 5: Robot front Simulation

Figures 4 and 5 shows part of the robot structure. The structure was designed using a rectangular box (1 m long, 0.9 m wide, and 0.6 m high) to ensure even weight distribution and stability during movement, as it serves as the foundation upon which the rest of the components are mounted. The structure was painted gray (0.8, 0.8, 0.8) to reflect the robot's practical, industrial appearance. Wheels Were constructed from cylinders (with a radius of 0.18 m and a height of 0.2 m). This shape was chosen to suit the agricultural environment and ensure stable movement that could support the main structure and enable movement on uneven terrain. A hinge joint was used to connect them to the main structure, enabling them to operate in a limited circular motion. The arm is designed with three moving cylinder-shaped layers, using a slider joint that allows movement for each layer. Each layer is controlled by a motor, providing precise linear movement and easy length control. It can be extended to reach remote locations. The first layer, arm1, is located closest to the palm tree and is equipped with a camera. The second layer, arm2, is in the middle. The third layer, arm3, provides precise movement and extends to reach target locations. A camera is mounted on the top of Arm1 and is designed to capture live images of the surrounding scene. To determine the location and ripeness of the dates, the camera captures the dates and sends the data to the image processing algorithms to identify the dates and determine whether they are ready to be picked.

A simulated Lidar sensor, mounted at the bottom of the robot on the front, is used to obtain three-dimensional depth information in the surrounding environment. It measures distances in meters based on OpenGL rendering. It is used to provide accurate distance information in the surrounding environment and helps the robot avoid obstacles. It is an ideal tool for agricultural harvesting robots to accurately locate trees and obstacles.

The control unit of the robot system is responsible for task allocation and execution. The main part that control and enables functioning and automation of the harvesting process. The main modules of the control unit consist of Vision module, that implements the CNN module for classifying the date. Yolo model is used to distinguish dates from the stem and other unimportant parts in the image. Yolo-v8 locates dates in the images then passed them to CNN for classifying mature and immature dates.

The data set used for training the YOLO is developed by labelling images extracted from original data set. The dataset contains 5 date varieties, Naboot Saif; Khalas; Barhi; Menei; and Sullaj. Data Labelling and YOLO Processing performed by training the YOLO model on labeled datasets. The labelling was carried into two stages. first labelling of 100 models manually. The manually trained models are used to train a mini model automatically. The process was iterated every 100 images. The final labelled dataset results in 8875. The labeling adopted a confidence level of 60%, the extent to which the model is certain that the detected object belongs to a specific category. Increasing the confidence level increases the precision level due to reducing the number of false detections. a 70% overlap level was then adopted, the extent of overlap of the bounding boxes drawn around objects. The data was divided into 70% for training, 20% validate, and 10% test. The first preparation of the data utilized Roboflow. Roboflow used to manage deep learning processes, allowing easily upload, organize, and process massive datasets. It has been used to label date clusters to distinguish them from palm trees.

Figure 6 illustrates the labelling process. The tagged dataset is utilized for further processing. The labelling process produced 8875 images. 70% of the labelled images used for training, 20% for validation, and 10% for testing. Yolo trained on these images to detects dates as mentioned previously Figures 7.

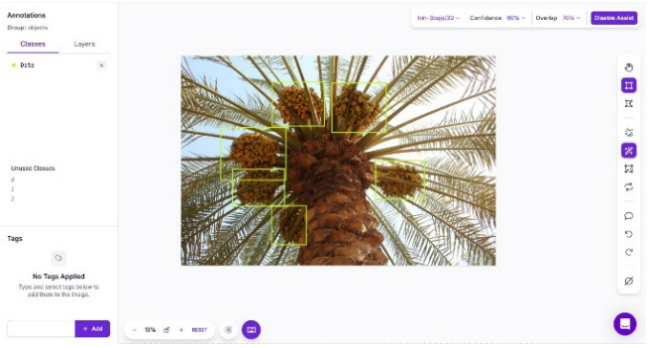

Figure 6: Labelling using Roboflow

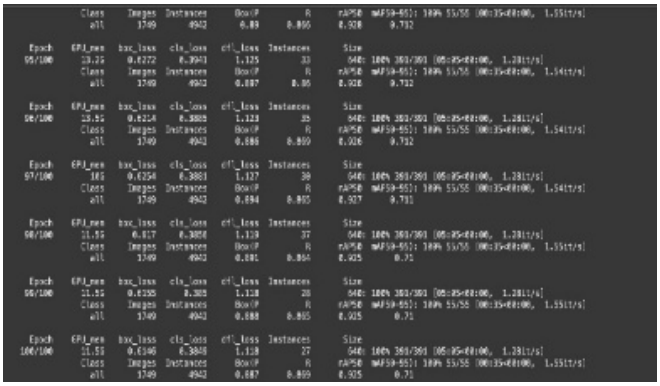

Figure 7: YOLO Performance during the training of the model

Figure 7 shows part of the training process, and the progress of the training as 100 epochs expired for the Yolo module. Table 1 displays one of the of training results, as it approaches 88.8%.

Table 1: The output percentages of the training

Precision: 88.89%
Recall: 84.681
mAP_50: 91.722
mAP_50_95: 70.749

The dataset abstracted to a single type, the Barhi date and the harvesting process was limited to the last three stages of the muter of the Barhi date named Khalal, Rutab, and Tamar stages.

Figure 8 displays examples of maturity stages of Bahri dates. Developing the CNN based on utilizing the tagged images. The Dataset reorganized into two main categories, for training and testing. The dataset consists of Barhi images. Training data set contains mature and ripe dates with 1,066 images (Khalal stage , Rutab stage , and Tamar stage). Immature and unripe dates with 738 images contain the first stage of Barhi dates (Immature dates). The second category of the testing data contains mature and ripe dates with 427 image contains of mature Bahri date (Khalal stage , Rutab stage , and Tamar stage), and immature and unripe dates with 344 image contains the first stage of barhi dates.

Figure 8: Date maturity stages

The convolution neural network architecture is developed. Figure 9 details the layers of the CNN architecture used to classify the maturity of dates based on the input images. The architecture consists of several Convolutional layers, followed by pooling layers of Max Pooling, then Flattened, converted the value to Dense connected layers, for a Sigmoid activation output layer. CNN architecture developed to enable the robot to determine the maturity of the date. The CNN receives a color image of 150x150x3. Three convolution blocks are applied with increasing filter sizes. The first layer used 64 filters, each of 3x3 kernels. The second layer used 128 filters, each of size 3x3 kernel. The third convolution layer is 256 filters with 3x3 kernels. The fourth convolution layer is 512 filters with 3x3 kernels. Each of the previous used ReLU (Rectified Linear Unit). The flatten layer is composed of 1034 neurons and a sigmoid activation.

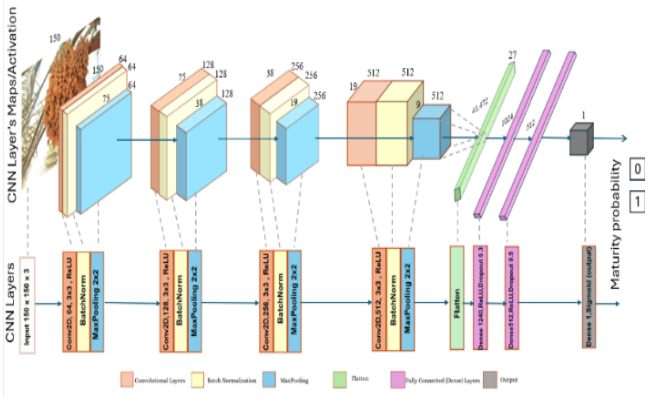

Figure 9: CNN Model Architecture

Portion of the extensive training of the CNN model is captured as in Figure 10. 100% recognition was achieved. Testing the model produced 96.23%.

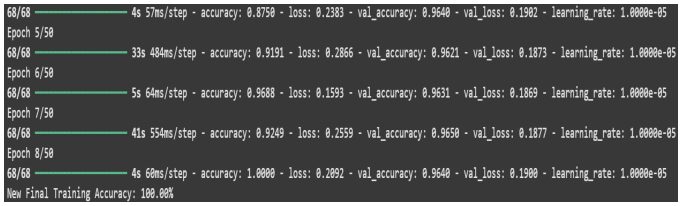

Figure 10: CNN Final Training Accuracy

Further model investigation on accuracy through validation of the model is depicted in Figure 11.

Figure 11: Accuracy Curves and Loss Curves

Figure 11 shows the Accuracy Curves and Loss Curves. These graphs show the evolution of the model's performance during the training process by tracking the training accuracy and validation accuracy. Training accuracy shows a positive trend as epochs reached 8, then it started to decline. Where the validation shows continuous positive trend. The loss curve harmonizes with the accuracy as it reflects the training behavior and the validation as well.

5.1. Environment

The term environment refers to the palm farm where the robot will operate. Palm farms are considered a unique agricultural environment, characterized by specific distribution of trees and limited spaces available for robot movement, Figure 12.

Figure 12: 1Real date palm farm environment.

The environment is simulated.by creating a simulated palm farm environment. The palm trees were designed in Blender and imported into Webots to achieve a more realistic representation. The goal of this simulation is to provide an accurate virtual environment to test the robot's movement and evaluate its performance in a setting that closely resembles real-world agricultural conditions Figure 13.

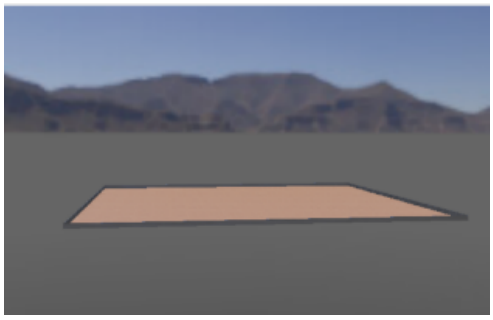

Figure 13: Creating the Environment

To create the simulated palm farm environment in Webots, a 60x60 meter terrain designed to allow placing 35 palm trees in organized rows with equal spacing of approximately 2 meters between each tree. The palm tree model was originally a pre-made asset, which is then converted to a compatible format and imported into Blender, Figure 14.

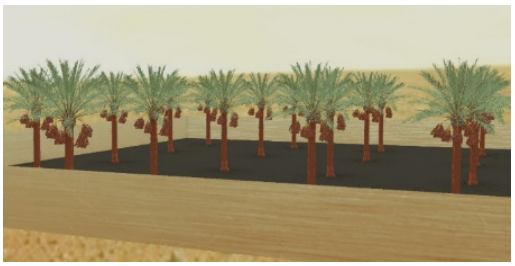

Figure 14: Simulated Date Farm

Figure 15 shows a simulated palm environment with date trees organized to resemble a real date farm. The integrated system, with the robot touring around the farm is shown in Figure 16. Portions of the control panel displays are attached with the corresponding robot movement.

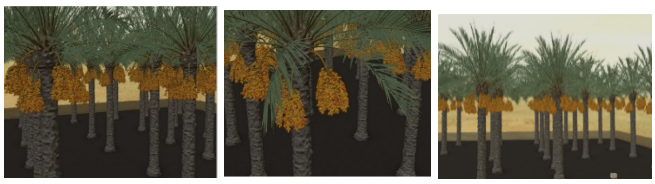

Figure 15: Simulated dates

Figure 16 shows movement of the robot in the date farm. In a, the robot was operated to locate the tree, it moves around and when detected a palm it stopped to capture images for the yolo module. When date are detected the images are passed to the CNN to determine

the maturity level and the system decide on the harvesting.

Figure 16: a. The robot in the field with the interface of the control panel, b. The robot moving on the field, c. The robot reached a palm and stop using the LiDAR. D. The robot use the camera, capture images that sent to the YOLO Module

6. Conclusions and Recommendations for Future Work

Currently the field of date harvesting lacks automated robots and harvesting mechanisms. The Date harvesting robot was designed and implemented in simulation environment for cost effective experiments. This motivates the production of a real robot. The field environment with palms was created and simulated to monitor the robot implementation on a real scenario.

Recommendations that develop the current model can be summarized in training a CNN model on several date varieties. The YOLO's performance can revisited by creating a new database preferably containing more images of palm trees and dates. Implementing the system on a real robot devised by the simulation to test actual performance in a real palm tree environment.

Conflict of Interest

The authors declare no conflict of interest.

Acknowledgment

The authors would like to acknowledge Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2026R925), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

References

[1] Ministry of Environment Agriculture and Water Report, Accessed: Nov. 27, 2024. [Online]. Available: https://www.mewa.gov.sa/ar/MediaCenter/News/Pages/News6742020.aspx.

[2] Ministry of Environment, Agriculture, and Water Report Accessed: Nov. 27, 2024. [Online]. Available: https://www.mewa.gov.sa/ar/MediaCenter/News/Pages/News4732020.aspx

[3] National Center for Date, Accessed: Nov. 27, 2024. [Online]. Available: https://old.ncpd.gov.sa/services/page/farm1/palm-care-guide

[4] M. Samir El-Habbab and A. M. El-Saad, "Assessment of Post-Harvest Loss and Waste for Date Palms in the Kingdom of Saudi Arabia," 2017.

[5] M. Abounajmi, "Mechanization of dates fruit harvesting," in *ASAE Annual International Meeting 2004*, 2004, pp. 171–178. doi: 10.13031/2013.16131.

[6] H. Altaheri, M. Alsulaiman, G. Muhammad, S. U. Amin, M. Bencherif, and M. Mekhtiche, "Date fruit dataset for intelligent harvesting," *Data Brief*, vol. 26, Oct. 2019, doi: 10.1016/j.dib.2019.104514.

[7] Mahmoud Yousif "Know the Biggest 10 Date Production in the World, Accessed: Nov. 27, 2024. [Online]. Available: https://www.aljazeera.net/ebusiness/2024/3/21.

[8] Zoumana KEITA, "YOLO Object Detection Explained", data camp. Accessed: Nov. 27, 2024. [Online]. Available: https://www.datacamp.com/blog/yolo-object-detection-explained

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection", [Online]. Available: http://pjreddie.com/yolo/

[10] "Introduction to Convolution Neural Network", geeksforgeeks. Accessed: Nov. 27, 2024. [Online]. Available: https://www.geeksforgeeks.org/introduction-convolution-neural-network/

[11] K. Schmid *et al.*, "Lidar 101: An Introduction to Lidar Technology, Data, and Applications," 2012. [Online]. Available: www.csc.noaa.gov

[12] Bhupendra Sharma, "What is LiDAR technology and how does it work?," geospatial world. Accessed: Nov. 27, 2024. [Online]. Available: https://geospatialworld.net/prime/technology-and-innovation/what-is-lidar-technology-and-how-does-it-work/

[13] Hamdi Taher Altaheri, Mansour Alsulaiman Ghulam Muhammad, "A VISION SYSTEM FOR DATE HARVESTING ROBOT," 2019. [Online]. Available: https://sites.google.com/view/daterobotic

[14] L. L. Wang *et al.*, "Development of a tomato harvesting robot used in greenhouse," *International Journal of Agricultural and Biological Engineering*, vol. 10, no. 4, pp. 140–149, 2017, doi: 10.25165/j.ijabe.20171004.3204.

[15] E. J. Van Henten *et al.*, "An Autonomous Robot for Harvesting Cucumbers in Greenhouses," 2002.

[16] Y. Xiong, Y. Ge, L. Grimstad, and P. J. From, "An autonomous strawberry-harvesting robot: Design, development, integration, and field evaluation," *J Field Robot*, vol. 37, no. 2, pp. 202–224, Mar. 2020, doi: 10.1002/rob.21889.

[17] A. F. Japar, H. R. Ramli, N. M. H. Norsahperi, and W. Z. W. Hasan, "Oil Palm Loose Fruit Detection using YOLOv4 for an Autonomous Mobile Robot Collector," *IEEE Access*, 2024, doi: 10.1109/ACCESS.2024.3446890

[18] *IEEE DataPort*, "Date Fruit Dataset for Automated Harvesting and Visual Yield Estimation,", [Online]. Available: https://ieee-dataport.org/documents/date-fruit-dataset-automated-harvesting-and-visual-yield-estimation

Hanan Hassan Ali Adlan has done her bachelor's degree from University of Khartoum 1992. She has done her master's degree from University Putra Malaysia in 1999. PhD degree in Computer Systems Engineering from University Putra Malaysia in 2005.

She is the developer of the Rough Shared Weight Neural Network. One of the developers of ElHa Neural network. Her research interest includes Intelligent Systems, Neural Networks, Pattern recognition, deep learning, computer vision, and Robotics.

RIHAM FAHAD Al ZAMANAN has done her bachelor's degree from Princess Nourah University in 2025.

She experienced developing in User Interface Design (Front-End and Back-End) with service integration. Her research interest towards developing software and designing systems.

LEEN ABDULAZIZ EL MUFLEH has done her bachelor's degree from Princess Nourah University in 2025.

She has experience in AI and machine learning. Her research interest includes computer vision, CNN, and image recognition

TALA AL MUGRIN has done her bachelor's degree from Princess Nourah University in 2025.

She is experienced in software development and interested in AI systems and Robotics. Her research interest includes developing software and building machine learning models

JORY Waleed AL HASSOUN has done her bachelor's degree from Princess Nourah University in 2025.

She is experienced in software development, and interested in building intelligent systems

JENRS

Received: 18 December 2025, Revised: 03 February 2026, Accepted: 06 February 2026, Online: 23 February 2026

DOI: https://doi.org/10.55708/js0502002

How to Fix Automation Flakiness: Root Causes and Enterprise-Level Solutions

Sujeet Kumar Tiwari*

Independent Researcher, Durham, North Carolina, 27540, USA

*Corresponding author: Durham, North Carolina, 27540, USA, sujeet0414@gmail.com

ABSTRACT: Flakiness in automation is one of the most intractable barriers to dependable enterprise CI/CD, in which organizations can run more than 50M tests daily, and a 5-10% flaky rate may spoil thousands of builds. The paper brings together empirical research and industrial case studies on UI, API, mobile, and data pipelines to describe the prevalent flakiness root causes, such as asynchronous UI, discontinuous DOM, unsteady test data, environment latency spikes, concurrency defects, and the absence of synchronization. It advances a multi-level stability design which incorporates deterministic locator strategies, clever wait handling, resilient API contracts, controlled test-data administration, and reliability designs of cloud-based environment patterns. The method is supported with the help of AI-supported analytics, where execution telemetry, the flakiness probability score, and heatmaps would be utilized to identify unstable tests early and devote effort to remediation. Large-scale settings reported case studies indicate a 5-10% to less than 1% drop in the rate of flaky tests, infrastructure savings of up to 2 times, and savings in diagnostic effort of up to 30-50%. The paper ends with providing explicit future directions, such as achieving a >95% accuracy on automated flakiness location, benchmark and KPI standardization, and stability engineering and integration with wider practices of reliability and governance. These guidelines will help ensure that test stability is an SRE first-class objective.

KEYWORDS: Automation Flakiness, CI/CD Pipelines, Multi-layer Stability Framework, AI-driven Flakiness Analytics, Enterprise Test Automation

1. Introduction

Automation flakiness is the phenomenon when automated tests are intermittently passed and intermittently failed, even in the absence of code or configuration interaction. This nondeterminism contradicts the assumption of the meaningful interpretation of the state of the system among tests that happen in CI/CD pipelines. It is magnified at the enterprise level. Google conducts more than 50M automated tests each day, where even a percent of intermittent failures can create thousands of bad releases and result in costly retests [1]. Flakiness is the first in the cross-layer and results in asynchronous user interfaces, dynamically augmented Document Object Model, volatile microservice API responses, variability in mobile devices, non-deterministic pipelines, and infrastructure problems. This study makes four primary contributions to the field of enterprise-scale test automation stability engineering. First, it provides a systematic taxonomy and synthesis of automation flakiness root causes across multiple layers, including UI, API, mobile, data pipeline, and infrastructure environments, based on empirical literature and industrial evidence. Second, it proposes a multi-layer stability framework that connects specific engineering interventions—such as deterministic locator design, synchronization strategies, controlled test data management, hermetic environments, and resilient API contracts—to distinct categories of flakiness, enabling actionable stability improvements at architectural and pipeline levels. Third, the paper introduces a metrics-driven stability measurement model, including flakiness rate, rerun overhead, diagnosis and repair time, and telemetry-based probabilistic flakiness scoring, supported by heatmap-style analytics for prioritizing remediation. Fourth, the study synthesizes industrial case studies and

operational practices from large-scale CI/CD environments, including predictive test selection, AI-assisted root-cause localization, and self-healing automation, and integrates them into a unified conceptual and operational model to guide organizations in improving pipeline reliability, reducing maintenance cost, and achieving enterprise-grade automation maturity.

In large organizations, strategic and not accidental is flaky automation. Constant flakiness causes loss of confidence by the developer in the testing suite; engineers are willing to rerun any jobs, turn a blind eye to failures, or even turn known checks off, undermining quality gates. Flaky suites slow down the release engineering process and cause tension between engineering and operations [2]. Cost is quantified on industrial evidence. The developers are estimated to devote at least 2.5% of their time to researching, fixing, or maintaining the flaky tests, instead of creating functionalities [3]. On the platform scale, providers like Google say that they dedicate between 2% and 16% of infrastructure spending to reruns due to flakiness instead of actual defects.

This research addresses three major objectives. It analytically categorizes the major root factors of automation flakiness in user interface, application programming interface, mobile, and data pipeline layers through synthesizing empirical and industrial evidence. For Design, the study provides a multi-layer stability framework, which incorporates deterministic locator schemes, powerful moves to synchronize, controlled test data, and environment standardization. It analytically describes and constructs AI-motivated flakiness detection and prediction, like probabilistic scores and predictive test selection models that rank high-value checks in continuous integration.

Its field of interest is the pipeline of large-scale delivery enterprises. When there are more than 10k automated tests and about 100k high-volume continuous integration jobs daily, then large mono repos and federated microservice ecosystems are said to be large mono repos. Within this space, the study takes into consideration automation of web user interfaces, APIs, mobile clients, and data processing workloads, such as cross-cutting end-to-end scenarios. Instead of conducting an analysis of a single proprietary dataset, it constructs existing published empirical results and case studies of goods to construct generalizable engineering patterns and standards.

To achieve its objectives, this study has been divided into different chapters. The literature review Chapter offers a literature review on flaky tests, root causes, and mitigation methods. The methods and techniques Chapter analyzes methods of research and the suggested multi-layer framework of stability. The experiment and results chapter summarizes evidence and case observable on the involvement of measurement of the effect of interventions on flaky test rates, rerun overhead, and diagnostic effort. The discussion chapter talks about results and trade-offs, and organizational implications. The study presents future research directions of quality analytics and stability engineering. The study concludes by offering practical recommendations.

1.1. Study Positioning and Validation Approach

This work is explicitly positioned as a systematic synthesis and engineering framework paper rather than a controlled experimental study introducing a proprietary dataset. The study integrates findings from peer-reviewed empirical research, industrial case studies, and operational reports from large-scale CI/CD environments to construct a unified, actionable stability engineering model.

2. Literature Review

2.1. Conceptualizing Automation Flakiness and Test Stability

Flakiness of automation denotes an automated test property where the results do not change significantly regardless of the system being tested, which challenges the fact that test results are consistent predictors of system behavior [4]. A flaky test is a test case that periodically passes, periodically fails on the indicated sequence, and a flaky test failure is a single failing on the sequence. Non-deterministic results are opposite to deterministic tests, where repeated performances with the same condition lead to similar results. Non-determinism may be in the test code, in the application itself, or in the platform itself, and thus harder to accomplish root-cause attribution.

Previously, several types of taxonomies of flakiness have been distinguished: user interface flakiness due to asynchronous rendering or dynamic Document Object Model updates; environmental flakiness due to operating system, browser, or network variability; and flakiness due to concurrency due to thread interleaving, race conditions, or shared mutable state [5]. Others include dependency, test order dependency, the outcome depends on the temporal sequence, or configuration flakiness because of version drift in third-party libraries or infrastructure.

Figure 1 provides a detailed view of how sever always of detecting flaky tests are implemented in automation contexts. The approaches demonstrated are rerunning of failed tests, running tests in parallel, and running tests in a different environment, which are used to detect inconsistent test results. The figure also emphasizes the significance of test results and log analysis, and special tools and frameworks to identify and address flakiness. The strategies follow the mentioned forms of flakiness, including UI flakiness, environmental flakiness, concurrency flakiness, and test-order flakiness. The detection methods will enhance the reliability and stability of the test.

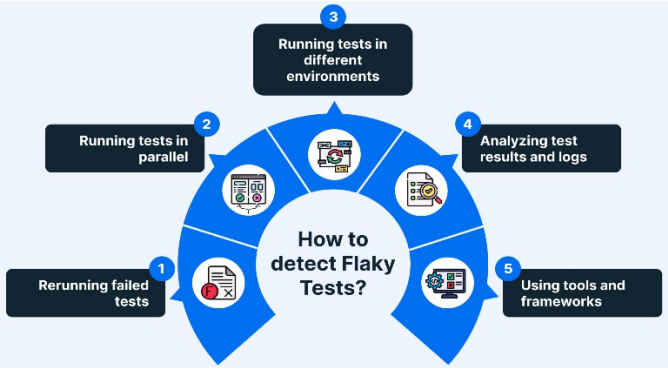

Figure 1: The techniques to find the flaky tests are the re-execution of the failed tests, parallel execution, environment variability, log inspection, and special detection tools and frameworks.

2.2. Prevalence and Cost of Flaky Tests in Large-Scale Systems

At Google, using test suites to the tune of 4.2 million, around 16% of test suites are flaky, such that tens of thousands of suites cannot be reliably deterministic. With 24 open Java systems sampled, a cross-project study, it had 810 flaky tests, of which about 75% were clusters of about 13.5 tests, implying systemic correlations of sharing fixtures or common infrastructure.

In dynamically typed systems, a Python study had shown a running probability of the absence of flaky test approaching 95% in a study that would have required about 170 reruns, which is impractical in a typical continuous integration feedback loop [6]. The data on industrial continuous integration also helps to show the economic cost: one case study reported that 0.63% of executions were rerun, but flakiness costs at least 2.5% of developer time in investigation, triage, and repair .This documentation is sufficient to justify a narrow stability-engineering agenda in which flakiness is no longer viewed as a side-effect of noise.

2.3. Root Causes across UI, API, Mobile, and Data Pipelines

Server-side and library code contain a lot of synchronous waits, concurrency defects, and test order dependency, especially when the tests must access a common database, a common cache, or a common message queue [5]. With web-based user interfaces the most problems are improper or unsynchronized assertions, assertions to an element whose state has not settled, frail locators as elements change attributes, and environmental factors, such as another browser engine or time zone. Test-runner API misuse and excessive use of generic exception handling are also identified as contributors to empirical studies of UI test suites, in particular when the is a large Selenium or Cypress test suite, of thousands of tests sharing infrastructure [7].

Mobile tests on the mobile platforms revealed that approximately 36% of flaky test failures have been traced to the mismatch of synchronization between the test thread and the test application and device fragmentation, background services, and unreliable network conditions [8]. Further causes of non-determinism in data-pipeline tests also include situations like race conditions during job scheduling, eventual consistency with distributed storage, and external use of data (through third-party APIs or upstream batch processing), which is revealed in multivocal reviews of flaky tests in continuous delivery systems. These patterns indicate that the flakiness is not simply the random noise, but in many cases, it hints at the systematic views in sync, isolation, or environmental control.

2.4. Existing Detection, Quarantine, and Mitigation Strategies

The most common and simplest mechanism that is deployed is the rerun-on-failure mechanism, where failed tests will be retried a specified number of times and labeled as flaky when the retries succeed. Although cheap to run, reruns prolong the pipeline time and fail to fix even the root causes. Quarantine testing of tests that exceed specific flakiness metrics is common in many large organizations, though, which block them on release blocks but still execute them in order to obtain information [9]. More advanced methods are based on either fixed or dynamic analysis to localize root causes, including the correlations between failure patterns in code changes and environment variables or concurrency primitives and clustering algorithms to determine systemic flakiness of sets of tests; both automated localization and humans can achieve about 82% accuracy with 428 projects at Google [10].

Simultaneously, algorithms based on machine-learning predict chances that a particular test will fail or act flakily on a specific set of change, and predictive test selection can be used to select high-value, low-noise tests, or identify faulty changes during continuous integration; the application of Predictive Test Selection at Meta with machine-learning methods has reduced the cost of testing infrastructure by half while seeing over 95% of individual test failure or more than 99.9% of defective change [11]. For compliance-based AI pipe-lines, where telemetry, traceability, and risk scoring are taken into account in the architecture to fulfill requirements of the regulatory environment, such as IEC 62304 and PCI-DSS, flakiness management is practically required to have observability end-to-end and settlement points in the test and deployment workflow [12].

Figure 1 illustrates an overview of the main approaches that have been applied in flaky test detection and mitigation during automation. Strategies related to these comprise rerun-on-failure, quarantine testing, localization of root causes, machine-learned-predictive-test selection, and specialized tools and frameworks. Each approach can be characterized by its purpose, performance, and the most common applications in the enterprise-level automation environment. The table shows the practical

Table 1: A summary of detection, isolation, and mitigation of flaky tests, their approaches, their effectiveness, and how flaky tests can be used in large-scale CI/CD.

Detection/Strategy	Description	Metrics/Accuracy	Compliance/Use Cases
Rerun-on-Failure	Failed tests are retried a specified number of times and labeled flaky if reruns succeed. Prolongs pipeline time and doesn't address root causes.	—	Low cost, basic mechanism
Quarantine Testing	Tests that exceed specific flakiness metrics are blocked from release but executed for diagnostic information.	—	Diagnostic purposes, doesn't fix root causes
Root Cause Localization	Correlating failure patterns with code changes, environment variables, or concurrency issues. Achieves ~82% accuracy in Google's automated localization.	82% accuracy with 428 projects.	Used for systemic flakiness detection
Machine-Learning Based Predictive Test Selection	Machine learning predicts the likelihood of failure for tests based on changes. Predictive selection reduces testing infrastructure cost by 50%, identifying > 95% of failures and > 99.9% of faulty changes.	Reduces infrastructure cost by 50%, 95% failure detection, and 99.9% defect detection.	Applicable in CI/CD to prioritize tests based on predictive models

uses of these techniques, such as the possibility to mitigate the cost of testing infrastructure, optimize diagnosis accuracy, and systemic breakage, all of which are intended to make continuous integration pipelines more stable and efficient.

2.5. Gaps in Current Research and Practice

Published work also concentrates on each of the layers specifically, such as unit tests or user interface automation, rather than on portfolios of cross-layers, which combine with user interface locators, API contracts, test data management, and environment stability. Standardized enterprise key performance indicators and benchmarks also such as what proportion of tests flake across risk category or maximum acceptable investigation time before a flaky failure is quarantined also lacking [13]. Proactive analytics has yet to be exploited: various organizations continue to triage by hand and do not exploit execution telemetry, temporal flake patterns, or spatial clustering to produce heatmaps of flakiness and a backlog of priority remedial actions, which is starting to be explored as showing that manifestations of systemic flakiness may be concentrated in tissues of co-failing tests that can be visualized in the form of a heatmap [14].

The experience of the related fields, such as BIM in field inspection workflows in zero-paper construction sites, indicates that end-to-end digital traceability, organized data collection, and feedback between field observations and design models can greatly enhance the inspection reliability and minimize the occurrence of rework [15]. Using similar principles to test engineering would imply that the exploration according to engineering should leverage the incorporation of rich telemetry, visual analytics, as well as automated decision support into stability-engineering practice, along with structured governance and continuous-improved cycles at enterprise scale.

3. 3. Methods and Techniques

3.1. Study Design and Research Questions

The practices embraced in this study are empowered within an enterprise setting where it uses continuous integration and continuous delivery pipelines to run tens of thousands of tests each day. The quantitative and observation design is implemented, in which the longitudinal data on the CI/CD telemetry and repository will be used to define flakiness, evaluate the mitigation strategies, and serve as a source of data-driven recommendations. Three research questions guide the analysis.

- RQ1 will answer what the overarching root causes of automation flakiness are between layers (UI, API, mobile, and data pipelines) in enterprise ecosystems, and their prevalence in the real world.
- RQ2 investigates the problem of the efficiency of the multi-layer stability framework in reducing the rate of flaky tests and rerun overhead, and the diagnostic effort in the real delivery pipelines.
- RQ3 focuses on analytics, and the question is in what ways analytics utilizing AI-based telemetry, such as failure history and runtime measurements, as well as

flakiness heatmap, may be used to anticipate and localize flakiness at scale with reasonable accuracy and recall.

3.2. Data Collection Methods

The data collection is focused on enterprise sources. The most significant one is the CI/CD server telemetry of such platforms as Jenkins, GitHub Actions, GitLab CI, Azure DevOps, or cloud native orchestrators [16]. To obtain a representative perspective, the research presumes the availability of logs containing 6 to 12 months' worth of actions, such that there are 20.2M of job runs during each day and tens of thousands of testing activities per day. Test results (pass, fail, skip), reruns are also included in each job record along with environment metadata, including operating system image, browser version, type of device, region, and result container image hash.

Build artefacts (logs and screenshots) are stored to provide diagnostics support. The version control systems can offer a record of the commit history, a branch history, and examinations of the change of size, whereas issue trackers can connect flaky failures to bug reports and remedy work. Analogous to how the operators of AI-optimized spine-leaf network fabrics compile fine-grained counters and traces to optimize routing and congestion control within the high-throughput data center, this model of large-scale telemetry collection is used [17].

3.3. Data Analysis and Flakiness Classification

The analysis step begins with the building of a pertest history of execution and labelling results. A test is considered to be flaky based on passing a test at least once and failing at least once, without there being any record of any code or configuration modification. Based on this, the flaky test rate is calculated by dividing the flaky tests by the total number of different tests in the suite. In support of this, flaky failure rate concentrates on failures and is described as the ratio of failing executions caused by flakiness, as opposed to defect revealing behavior [9]. Systemic flakiness can be observed by aspects of cluster-wide data: cluster size counts the number of tests that fail consistently across one time period, and cluster density counts the extent to which failures in clusters are time or component.

Operation process metrics such as Mean Time to Diagnose, Mean Time to Repair flaky tests can be calculated using timestamps of failure detection, creation of issue, fix commits, successful verification run, industrial case studies have revealed that (operational) process metrics can quantify time lost to flakiness by developers, as well as justify investment in automation [3]. Time-series techniques are used with rerun rates and failure to track its occurrence, correlation analysis is used to be able to indicate the relationship between flakiness to environment variables, API latencies, or DOM change frequency, clustering of failure co-occurrence to highlight systemic hotspots.

Table 2 provides a summary of the main metrics used in the analysis and classifications of flaky tests in CI/CD pipelines. The Flaky Test Rate is a measure of the proportion of non-deterministic tests, and the Flaky Failure Rate differentiates between test failures that are caused due to flakiness and those caused due to fundamental defects. Cluster Size determines systemic flakiness in terms of the count of failing tests that flask at the same time, and Cluster Density is a measure of where these flaks are concentrated over a specific set of time or components [18]. Mean Time to Diagnose and Mean Time to Repair provide an insight into how the flaky tests operate, and the effect of their cooperation in detecting and resolving the problems is achieved. These metrics, when combined, provide a holistic picture of the instability of the test to make specific improvements and optimization of the processes.

Table 2: Key indicators of categorizing and examining flaky tests, causes, and operational effects of CI/CD pipelines, based on industrial case studies.

Metric	Definition	Purpose	Source
Flaky Test Rate	The ratio of flaky tests to the total number of distinct tests in the suite.	To quantify the prevalence of flaky tests within a suite.	[9]
Flaky Failure Rate	The ratio of failures caused by flakiness versus defect-revealing behavior.	To determine the proportion of failures caused by flakiness.	[9]
Cluster Size	The number of tests that regularly fail at the same time.	To identify systemic causes that affect groups of tests.	[3]
Cluster Density	The concentration of failures within a defined time or component.	To understand the density of flaky failures within a defined scope.	[3]
Mean Time to Diagnose	The time spent diagnosing flaky tests from failure detection to successful verification.	To assess the effort required to diagnose flaky tests.	[3]
Mean Time to Repair	The time spent repairing flaky tests from issue creation to successful verification.	To assess the time spent repairing flaky tests and its impact on productivity.	[3]

3.4. Multi-Layer Stability Framework Design

In response to RQ2, the study identifies a multi-layer stability framework with the interventions of which

observed root causes are mapped. The UI and mobile layer is concerned with deterministic locators such as stable attributes of data or accessibility identifiers, whereas self-healing locator mechanisms can be augmented to overcome minor structural alterations [19]. Smart wait, as opposed to a hard sleep, involves explicit wait for predicted conditions and visual failure, and a network check of asynchronous rendering issues. The API and service layer is targeted at process reliable contracts and patterns of communication: refining endpoints are idempotent when they can be, clients use the retry and timeouts facilities, and cycling of the circuit-breaker process by limited back-off to ensure defect coverage in transient failure cases [20].

The data and pipeline layer supports meandered record information, snapshot segregation of perforated databases, on artificial case information, and man-made case set sequencing to obstruct ordering based races. The environment and infrastructure layer is standardized on top of hermetic test environments with container images, a golden base operating system template, a reproducible browser stack, and Nightmare network virtualization to model the network latency and packet loss. Layers align with the best practice of cloud infrastructure automation, where declarative configuration and repeatable provisioning are used that minimize configuration drift and incidents caused by the environment [21].

3.5. AI-Driven Telemetry, Heatmaps, and Self-Healing Integration

The last methodological element will answer RQ3 by specifying an AI-driven analytics pipeline. On each execution of the test, telemetry is gathered, which consists of exit status, duration, resource usage, environment fingerprint, and key log characteristics [22]. This information is summarized as feature vectors on the test case and job levels. Gradient-boosted trees or logistic regression is a supervised learning model that uses historical failures that are classified as flaky or defect-revealing to predict a flakiness probability score of each test and situation. These scores have the effect of being used in predictive test selection, in which tests with large failure signals and low probability of flaking are selected for important execution, and tests with known large probability of low signal flaking are executed less frequently or in non-blocking phases.

Meanwhile, the opportunity cost of this visualization is flakiness heatmaps showing services, components, environments, and time windows to identify hotspots within the system. The architecture project also anticipates the combination with self-healing automation tools like AI-controlled locators, dynamically adjusted wait tuning, and declarative recovery steps, which, if found in failed tests, have behavioral patterns characteristic of flakes, and are invoked automatically.

Figure 2 illustrates an analytics pipeline where AI can be used to improve automated test management. The pipeline stores telemetry data regarding each of the run tests, including exit status, time, resource usage, and a fingerprint of the environment, and part of the key logs. AI analysis tools, including a defect predictor and a test package analyzer, use this data to create flakiness probability scores based on learned anything supervised models, including gradient-boosted trees or logistic regression [23]. These scores are used to guide prediction tests, with higher failure prediction tests that have high flakiness rates ranking higher. The system also provides self-healing capabilities and displays heatmaps of flaky test failures with flakey services, components, and environments, using flakiness heatmaps.

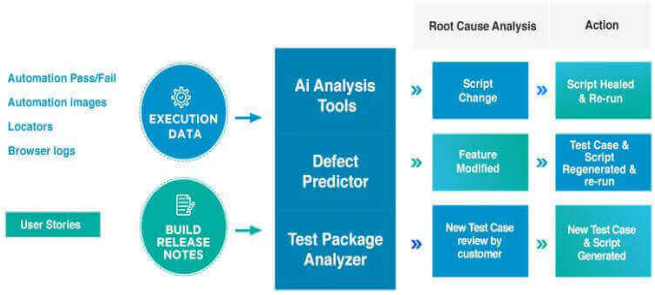

Figure 2: An AI-based analytics pipeline to run a test, such as telemetry, predicting flakiness, heatmaps, and auto metrics to manage weighted and broken automated tests, and usage to fix the issues.

3.6. Illustrative Telemetry-Based Validation Method

In addition to synthesizing empirical findings, this study includes a small illustrative validation designed to demonstrate the operational applicability of the proposed framework. Because proprietary enterprise CI/CD datasets are typically confidential and unavailable for public analysis, this validation uses a simulated telemetry dataset constructed using representative distributions reported in prior industrial studies [3], [6], [11].

The simulated dataset models a representative enterprise pipeline with the following characteristics:

- 10,000 automated tests
- 30 days of execution history
- 300,000 total test executions
- Baseline flaky test rate of 6%
- Failure clustering affecting 20% of flaky tests
- Environment variability across 5 execution environments

Each simulated test execution record includes:

- Test identifier
- Execution timestamp
- Pass/fail outcome

- Environment identifier
- Execution duration
- Historical failure count

4. Experiments and Results

4.1. Baseline Flakiness Profiles from Existing Studies

This chapter combines empirical findings and case studies of enterprises to put a real-world level of baseline on automation flakiness before mapping the gains onto the multi-layer stability framework. Rather than new experiments, it correlates existing discoveries into a profile that businesses can refer to in conducting benchmarking. The three ecosystems offer a lot of insight: Java and Apache codebase scale, Android applications, and Python applications with large scale automated tests. In these environments, the percentage of tests that flake out is usually isolated in a small section of the entire tests, the cost of re-crushing out is non-trivial, and the diagnostic effort is taking visible capacity of the developer.

Empirical studies of Apache and other Java-based projects have reported 201 unique network test fixes out of a collection of industrial repositories and open source repositories [24, 25]. The most recurring root causes in such a corpus are wrong asynchronous waits, concurrency errors such as race conditions, and the order dependency of the tests, which are affected by the order in which the tests are executed. Timing and synchronization failures take the pre-eminent place with concurrency and order effects as groups of minor, though significant categories.

Table 3 offers a summary of the empirical studies about the profile of the flake base in the three significant ecologies, such as Java and Apache codebase, Android projects, and Python ecosystems. It displays the most popular flakiness causes in every platform, including asynchronous waits, concurrency issues in Java, test-order problems in Java, and synchronization problems in mobile apps. The table also highlights the high price of flakiness detection in Python, in which more than 170 reruns are necessary to be 95% sure that a failing test is not flaky [26]. The most important observations of these research works are that focused mitigation strategies should be applied, including enhanced synchronization utility of mobile applications and test isolation, and design of Python-based systems. The results of these works guide the best practices in curtailing flakiness in automated tests.

In [27], the authors found 77 flakiness related commits in 29 Android projects, which they categorized into 5 categories of root causes, of which approximately 36% could be related to synchronization minds between the test code and the application under test [27]. In practice, these problems will be in the form of tests that expect activities or fragments to be on screen or fully set up when they should not be, or a background service will finish in a set delay time, independent of the device model or the operating system version. Others include setting up the environment, network/backend flexibility, and bad treatment of asynchronous methods to call-back or event-queues. The underlying patterns are rather limited as they indicate that mobile flakiness can be reduced down to a considerable extent, but when teams invest in the standardized synchronization utility and effective device farm management.

For Python ecosystems, in [6], the authors found the operationally expensive cost of detecting flakiness with high confidence: the analysis they performed estimated it would take more than 170 reruns to be 95% certain that a failing test does not show flakiness conditions [6]. This statistic demonstrates that it is practically impossible to use naive rerun strategies with continuous integration systems. Instead, companies need to mix goal-specific reruns with improved isolation, improved test design, and analytics-informed prioritization, which is comparable to how the engineers behind the NoSQL platform make tradeoffs between latency, consistency, and throughput when selecting a data store between MongoDB and Aerospike to support large-scale transactional workloads [28].

4.2. Case Study 1 – Google: Root Cause Localization and Quarantine

Google has had experience with flakiness that would give an in-depth picture of what flakiness entails when automated testing is handled at hundreds of millions of daily executions. The automatic localization system described by [10] was applied to 482 projects and reached an accuracy of around 82% in isolating the code that caused flaky behavior through correlation of failure patterns with code changes, stack traces, and metadata of the context of the shift [10]. This precision allows localization to be implemented within developer processes, assigning failures their probable root cause and the owner of the candidates.

To supplement localization, in [6], the authors estimated the cost of relying on reruns and quarantine strategies and estimated that between 2 and 16% of Google testing infrastructure resources are actually spent rerunning flaky tests instead of exposing new defects. These results are closely associated with the environment and management layer of concurrency fields in the proposed multi-layer framework: the hermetic environment and deterministic scheduling can prevent some forms of flakiness, and localization and quarantine policies will manage and provide a feedback mechanism. Key performance indicators that can be obtained by enterprises on the basis of this case include restricting the test infrastructure dedicated to reruns to just single digit

Table 3: A summary of baseline flakiness images of Java, Android, and Python ecosystems, plus the key reasons, insights, and operational difficulties of automated testing.

Ecosystem	Study Findings	Root Causes	Key Insights	Source
Java & Apache Codebase	201 unique network test fixes; root causes include async waits, concurrency errors, and test order dependency.	Asynchronous waits, concurrency errors (race conditions), and test order dependency.	Flakiness is concentrated in a small subset of tests, with concurrency and order issues as common causes.	[6]
Android Projects	77 flakiness related commits in 29 projects; 36% attributed to synchronization issues between test code and app under test.	Synchronization issues, environment configuration, network variability, and improper handling of async callbacks.	Mobile flakiness can be reduced by investing in synchronization utilities and effective device farm management.	[27]
Python Ecosystem	Over 170 reruns required to be 95% confident that a failing test is not flaky.	Flakiness detection costs are high, with over 170 reruns required for confidence in non-flakiness.	Naive rerun strategies are inefficient; companies need improved isolation, test design, and analytics informed prioritization.	[6]

percentages and aiming at obtaining more than 80% accuracy on the localization of root causes in cases of flaky failures.

4.3. Case Study 2 – Meta/Facebook: Probabilistic Flakiness and Predictive Test Selection

The contribution of predictive test selection by Meta shows how machine learning can help reshape the economic dynamics of the management of flakiness. In [11], the authors stated that the predictive test selection model at Meta saved approximately two times the overall cost of testing infrastructure and managed to detect more than 95% of individual test failures and more than 99.9% of the defective changes introduced by developers. The fundamental idea is the Probabilistic Flakiness Score, which approximates the probability of failure in a particular test and scenario being based on flakiness instead of an actual regression.

The high probability tests with low historical reveals the power of faults are deprioritized, whereas the high possibility tests with low flakiness are executed more or earlier in the queue. Within the framework of this study, the Probabilistic Flakiness Score undergoes the AI-driven telemetry and heatmap layer, which enables teams to see where the flakiness is focused as well as establish policy limits on whether to skip or quarantine tests [26]. These evaluation metrics, such as false positive flaky classification, precision, and recall in predicting flaky and defect revealing failures. The proportion of pipeline time devoted to high value tests itself becomes fundamental to operations decision making.

Figure 3 shows the test optimization process that has been employed by Meta to enhance the means of managing flakiness with the help of machine learning. It starts with the input of all test cases, and then probabilistic flakiness is analyzed to estimate what probabilities different test cases have the possibility of failure because of flakiness, as opposed to the presence of a defect. The execution optimization step is based on such scores of flakiness to organize tests with the most efficient order, using the tests that have a low score of flakiness and high failure signals. Outcomes and callouts are analyzed to determine significant outcomes. This procedure leads to substantial cost savings on tests since it can be illustrated by the example of Meta, which saved about two times the total cost of testing infrastructure and discovered more than 95% of failures and 99.9% defective changes.

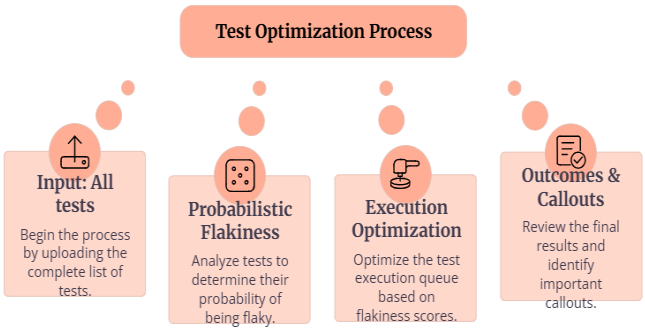

Figure 3: The test optimization process through probabilistic flakiness analysis and predictive test selection to rank the tests by the criteria of flakiness scores to enhance efficiency in executing the tests.

4.4. Case Study 3 – Industrial CI Cost Analysis and Systemic Flakiness

Industrial case studies of companies like CQSE have demonstrated that although flakiness may be applied to only a small part of the tests, its economic effect is disproportionate. In [3], the authors reported a case in

which reruns alone required $3/month in direct infrastructure expenditure, was incurred. However, developers were spending at least 2.5% of their productive effort on flakiness related activities, such as investigation, repair, monitoring, and communication. Another industrial report revealed that about 1.28 of the developer time and about $2250 every month was sunk into fixing flaky tests by a mid-sized organization that has a modern CI pipeline.

These figures bring out the reality that human work, rather than the cost of computing, can be the dominating factor in the total cost of flakiness. Further studies of Systemic flakiness indicate that around 75% of flaky tests are in clusters with an average of around 13.5 tests, suggesting that a high number of tests have common introducing factors as shared fixtures or external dependencies. For enterprise returns, it implies that even a small number of clusters can eliminate significant proportions of the flaky incidents. It will be equivalent to circular economy efforts in aerospace, where a small set of high-impact materials is targeted to achieve large payoffs in terms of recycling efficiency and cost savings .

4.5. Self-Healing Tests and Multi-Layer Stability: Evidence from Tools and Practice

The recent progress in self-healing test automation gives support that flakiness may be overcome through the use of a deterministic design coupled with adaptive tooling. Commercial AI-based testing and open source infrastructure, including Helenium, states that they have found wide spread reductions in maintenance workloads in case self-healing locators and smart wait tooling are operated at scale. In [29], the authors explains deployments in which AI native self-healing features have helped low-er the cost of test maintenance up to 35% by automatically updating selectors and timing plans as controlled interface changes happen .

The tools are used in the multi-layer stability framework primarily on the UI and mobile layers with a more stable synchronization and, when more brittle selectors are excluded, fewer false alarms, and more meaningful reruns being achieved at the expense of infrastructure by spending less time on diagnostic superficial failures. Together with telemetry and heatmap features that organizations such as Meta use, self-healing tools can assist enterprises in transitioning to proactive stability engineering instead of reactive firefighting [19]. Measures of success might be the flaky test rate reduction to initial ranges of between 5 and 10% to less than 1%, and the decrease in mean time to repair flaky tests, where the use of improved diagnostics, automation, and prioritization reduces it by 30% to 50%.

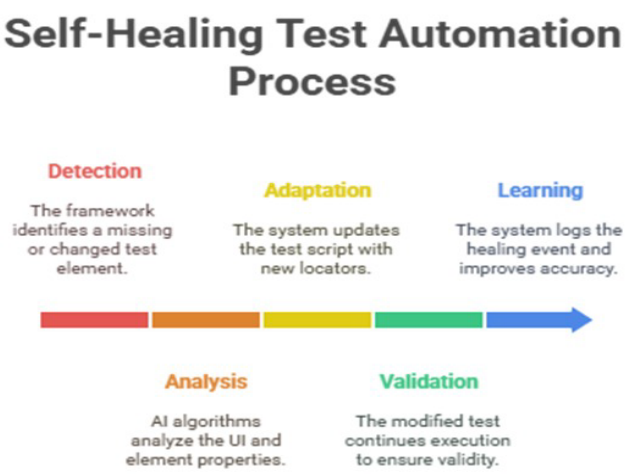

Figure 4: The self-healing test automation process, with the help of AI-driven tools that are used to detect, adapt, analyze, and validate tests, minimizes maintenance costs and enhances test stability.

Figure 4 illustrates the process of automation of self-healing testing, and it helps minimize the effects of flaky tests due to AI-based automation. This is triggered by the Detection step, where the framework identifies missing or manipulated elements of tests. The adaptation phase involves the system that modifies the test script with new locators to account for any change in the test environment. During Analysis, the AI will process the UI and element properties based on consistency, and it is preceded by the step of Validation, where the changed test must be run to ensure its correctness. The Learning phase records the healing occasion and increases accuracy in the future. This strategy includes intelligent wait devices, self-recovery locators, and decreases maintenance financial assets by approximately 35%, increases the rate of synchronization, and limits fake alarms as experienced in structures such as Helenium and the execution of Meta.

4.6. Illustrative Validation Using Simulated CI/CD Telemetry

To demonstrate the practical applicability of the proposed framework, illustrative validation was conducted using the simulated telemetry dataset described in Section 3.6. The dataset represents a CI/CD pipeline executing 10,000 automated tests over a 30-day period, with a baseline flaky test rate of 6%, consistent with reported industrial ranges [3], [6].

4.6.1. Baseline Stability Metrics

Initial analysis identified:

- Total tests: 10,000
- Flaky tests detected: 612 (6.12%)
- Total executions: 300,000
- Flaky failures: 8,420 (2.81% of executions)
- Mean simulated diagnosis time: 42 hours
- Failure clustering analysis revealed that:
- 73% of flaky tests belonged to clusters
- Average cluster size: 11.8 tests

- Cluster hotspots concentrated in 3 simulated services.

These findings align with previously reported systemic clustering behavior in enterprise systems [14].

5. Discussion

5.1. Synthesis with Prior Work on Flaky Tests and Stability Engineering

The multi-layer stability framework suggested is an extension of the existing literature of flaky testing to include causes, detection, and mitigation as one systems-oriented picture. Current surveys focus on timing, concurrency, environment, and test-order problem taxonomies and advise mitigation measures like reruns or quarantine, yet little to no linkage is built between such categories and actual architectural levers in the UI and API space, mobile space, data space, and infrastructure space.

The framework relates each category to specific controls on engineering, deterministic locators, and smart waits in the UI and mobile layer, patterns of contracts and idempotency in the API layer, snapshotting and synthetic tests in the pipeline layer, and hermetic and reproducible environments in the infrastructure layer. This resembles data-driven engineering in high performance computing and financial customers, in which a reliability objective is achieved by making explicit data model selection, scheduling policy, and re-source utilization technique decisions. Its stability is an emergent feature of a vast number of coordinated design decisions, and not a test in itself.

5.2. Trade-Offs Between Stability, Speed, and Cost

Flakiness reduction also presents quantifiable trade-offs between stability, speed, and cost, which organizations must quantify and not to qualitatively assess. Additional reruns minimize spurious failure rates and jump up the consumption and feedback latency curve, which can slow down release rates. Self-healing locators, the additions will have higher engineering and licensing costs. Nonetheless, as indicated by empirical reports, AI-native self-healing enables one to cut test maintain expenses by approximately 30% to 35% through automatic updating of selectors and synchronization rules as interfaces change. Those tradeoffs can be paid back in a few quarters in the initial investment in hundreds of suites, the lost developer time, and the drop in blocked releases. Quantitative models examine the tradeoff between the cost of, and the value of, a higher rate of provisioning compared to the flaky failure rate and the length of build blockage.

Table 4 demonstrates the trade-offs of stability, speed, and cost in test automation. It contrasts the effects of various strategies like longer reruns, locators healing themselves, reducing test maintenance, and quantitative models. The table points out that, whereas reruns bring about increased stability by minimizing the rate of spurious failures, they also slow down the pace of release and raise the infrastructure expenses. Self-healing locators are more stable and faster, but cost more in initial engineering and licensing, and maintenance cost savings should be appreciated over the long term. Reduction of test maintenance leads to high-speed application and reduced expense in terms of automation of updates, as much as 35%. The quantitative models assist in balancing the cost allocation, utilizing resources, and minimizing the flaky test rates. These trade-offs play a critical role in sensible decision making as regards test optimization.

Table 4: An overview of stability, speed, and cost trade-offs in test automation within the context of rerun effects, self-healing locator effects, and reduction of maintenance.

Trade-Offs	Impact on Stability	Impact on Speed	Impact on Cost
Additional Reruns	Reduces spurious failure rates, improving stability but increases rerun overhead.	Slows release cycles due to increased reruns and feedback latency.	Increases infrastructure cost due to more reruns but provides more reliable results.
Self-Healing Locators	Improves stability by reducing false alarms and brittle selectors but adds initial cost.	Improves test speed by reducing diagnostic time and rerun frequency.	Higher engineering and licensing cost but yields long-term savings through automation.
Test Maintenance Reduction	Reduces maintenance effort, improves stability by minimizing manual updates.	Faster execution as it reduces the need for manual intervention.	Saves 30-35% on maintenance costs by automating locator and synchronization rule updates.
Quantitative Models	Helps quantify the relationship between infrastructure costs, failure rate, and build blockage time.	Allows faster releases by optimizing the allocation of resources and reducing downtime.	Helps to optimize cost allocation by balancing provisioning with flaky test rate and blockage.

5.3. Organizational and Process Implications for Enterprise Test Automation

Since flakiness is also socio-technical, it cannot be controlled only through governance; it needs tooling. A practical mechanism is to establish explicit rules, such as an explicit set of flaky test budgets, such as less than 1% flaky tests per critical service, and a strict limit on quarantined tests to the current sprint. In [29], the author numeric policies would result in a clear ownership of such policies and enable the teams to handle test stability as a controlled risk rather than the background noise. Flakiness triage may be built into pull request workflows and CI gates, such as in change based testing implementations where predictive test selection and historical data on failures are surfaced during review time so that it may be used to determine which suites can and should be run, or which suites can be quarantined [30]. This motivates developers, software development engineers in test, reliability engineers, and platform teams to coordinate at the early stages to deal with the root cause. Role positions need to be clear, as there should be SDETs in charge of test design and guardrails, cross cutting overlays of observability and incident responses in the reliability engineers, platform teams in charge of hermetic environments, and orchestrator level controls preventing flakiness in the environment.

Figure 5 demonstrates the main points in effective enterprise test automation, such as the importance of automation in relation to business objectives, the necessity to focus on high-impact test cases, and the use of AI and ML to gain smarter testing. The process includes the selection of the right tool that can be incorporated in the tech stack, the construction of the scalable and reusable surroundings, and the optimization of the testing with real-time analytics. These measures assist in developing a distinct framework of how to deal with flakiness and enhance test stability. Predictive test selection and flakiness triage allow integrating across CI workflows such that teams ensure coordination earlier on to find and resolve the underlying causes of instability and have clear responsibilities between SDETs, reliability engineers, and platform teams [31].

5.4. Implications for Automation Maturity and Pipeline Health

The flaky tests measure the maturity of the automation strategy of the organization. Low maturity is met not with ad hoc reruns, manual retries in CI dashboards, and a culture of playing blind to problematic known suites, which leads to a lack of confidence in automation [32]. More mature pipelines are also predictive, self-scheduling, and under continuous monitoring: structure failures caused by flaky failures are automatically made known and give rise to a set of classifications, quarantined or sent to owners depending on the diagnostics context.

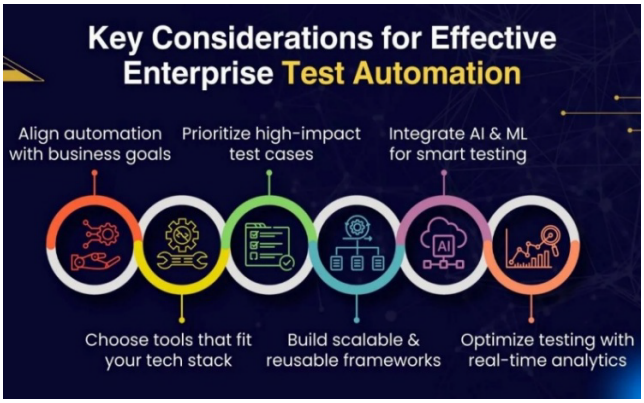

Figure 5: The primary factors leading to successful enterprise test automation put the emphasis on business objectives, prioritizing case tests and integrating AI/ML, and optimizing with real-time analytics.

The pipeline's health could be gauged by the percentage of pipeline failures that could be specified by the flakiness and throughput of the average build blockages happening due to unstable suites, and the fraction of the tests that could be covered by self-instrumented health control or telemetry control. This feedback is like pre silicon design for test feedback loops deployed to enhance the process of productizing a given GPU, with the test coverage data and defect patterns fed back into the design loop to minimize the escape rates and shorten the bring uptime period [33]. Periodic appraisal of pipeline condition measurements at the engineering leadership level allows constant enhancing of stability.

5.5. Limitations and Threats to Validity

The conclusions have several limitations that limit the generalizability of the findings. A large portion of quantitative data regarding flakiness is based on case reports publicly released by organizations like Google, Meta, and CQSE, and on academic codebase datasets with a bias towards Java, Android, and Python [34]. Different root cause distributions and cost structures may be found in businesses whose technology stack has very different core technologies, including a mainframe based financial institution or a real time embedded gear part vendor. CI datasets can be biased: logging fidelity, rerun policy, and naming tests can be systematically different in most systems and thus cause the under exploitation or over reporting of flaky behavior.

Measurements also may be distorted owing to tool specific behavior, to different CI orchestrators, and to different device farms, which process retries, timeouts, and environment configuration in a systematic manner, which has a systematic impact on the observed flaky rates. Threats to domain factors include the ability of the mobile environment to be device and network fragmented, the web environment being sensitive to browser and tooling

churn, and the data pipeline becoming mixed up with outside providers and schedulers. The multi-layer structure and metrics can be widely used, but local calibration of them with the telemetry of an organization is critical.

6. Future Research Recommendations

6.1. Advanced AI Models for Flakiness Prediction and Root Cause Localizations

The further development should be related to the more sophisticated AI models of predicting and locating flaky tests that go beyond basic heuristics or superficial learning. Achieving approximately 82% flaky test auto root cause localization on rule-based pipelines and classical machine learning models is the state of the art in the modern industrial practice; a natural research goal is to achieve above 90% using deep learning models and with the help of large language models capable of doing joint reasoning based on looking at logs, code, test artefacts, and configuration history [10]. Those models might incorporate stack traces, code differences, and environment metadata into some common representation space and be trained to rank probable individual root causes of each clustered flaky failure. A challenge lies in designing the training regimes, which are hard to label as noise because the labels formed based on CI data are still probabilistic.

6.2. Cross-Layer and Systemic Flakiness Modelling

The other research direction is cross-layer modelling of systemic flakiness of microservices, serverless, and event driven architecture. The work done reveals that about 75% of the flaky tests tend to be in clusters about 13.5 in size, but many analyses have been restricted to monolithic or single repo systems [14]. Future research should employ cluster analysis across inter service boundaries by comparing the interests of failure co-occurrence with call graphs, message topics, and deployment topology. Methods of distribution systems benefit, like scalable leader selection and consensus with incomplete information, can be inspirational, and algorithms that determine global flakiness hotspots using limited coordination overhead [35]. Long-term longitudinal studies over a period of 12 to 24 months also demonstrate the stability, drifting, or episodic nature of systemic clusters and the influence of architectural changes on them.

6.3. Standardized Metrics and Benchmarks for Flaky Test Rate and Pipeline Health

Standardized measures and open benchmarks, which include UI, API, mobile, and data pipelines, are also in urgent need. Practical research records unstable test rate, unstable failure count, as well as rerun overhead via inconsistent definitions, resulting in the inability to make cross-tool or cross organization comparisons [4]. Future studies can suggest a set of indicators that are minimal, such as: flaky test rate per suite and per component; ratio of pipeline failures due to flakiness; percentage of rerun time as per percentage of all executions; and mean flaky test diagnosis and repair time.

Public benchmark datasets, which will involve identified CI logs, requiring 10,000 or more tests, 100,000 or more job executions, and classified classifications of flaky results, would allow reasonable evaluation of detection intellectual property and self-laboratory techniques. Benchmark scenarios desired must be reduced across a variety of 4 domains: web UI, microservice API, Android or iOS mobile applications, and a batch or streaming data pipeline [36]. The researchers can also report similar results, such as a reduction of flaky failure share by 2x or a reduction of the time to diagnosis by 30%, often under well-defined conditions.

Table 5 summarizes essential standard measures of assessing the rate of flaky tests and the health of a pipeline. It highlights such metrics as the number of flaky tests per suite and component, pipeline failures because of flakiness, a percentage ratio of reschedule to total execution, the mean time of diagnosis and fixing flaky tests. These metrics help in quantifying instability in automatic tests, the cost of rerunning, and the effort in maintaining a flaky test. The table provides an overview of the organization against which the stability of tests, the optimality of CI/CD pipelines, and improvement should be made based on themselves, and specifically in accordance with specific, actionable indicators.

Table 5: An overview of automated or standardized measures of flaky test rates, pipeline health, rerun overhead, and diagnosis time to benchmark and test automated test performance.

Metric	Definition	Purpose
Flaky Test Rate	The rate of flaky tests per suite and per component.	To quantify the extent of flaky tests in a test suite.
Pipeline Failure Due to Flakiness	The ratio of pipeline failures caused by flaky tests.	To identify the impact of flakiness on pipeline failures.
Rerun Time as Percentage of Executions	The percentage of time spent on reruns compared to all test executions.	To assess the cost of rerun overhead in the CI/CD pipeline.
Mean Flaky Test Diagnosis and Repair Time	The average time spent diagnosing and repairing flaky tests.	To measure the effort involved in fixing flaky tests and reducing downtime.

6.4. Human and Socio-Technical Dimensions

The human and socio technical concern of flakiness should be addressed in the future. The qualitative and mixed method research is required to comprehend the perception of creating flaky tests by developers and its effects on the behavior, including either unawareness of failing builds, excess retries, or disabled checks [37]. The relationship between flaky failure rate and the metrics of burnout should be measured in a survey and telemetry-based analysis, such as afterhours debugging or middle place revert rate.

Mobile or UI-heavy teams demand special consideration because flakiness is sometimes perceived to be more intense, based on the devices being able to fracture and their visual instability. Organization experiments could compare those that have and do not have formal flakiness budgets, triage rotations, or stability-focused OKRs, and compare the results in terms of differences in flaky test rate, blockage time in a build and release cadence. The findings of these studies inform the structure of interventions combining technical controls with process and cultural change to ensure that stability engineering is an integrated and shared responsibility and not an isolated and enclosed quality activity [38].

7. Conclusions

This study has explored the issue of automation flakiness as an enterprise scale reliability risk within CI/CD, not a mere trivia, and shown the impact of automation flakiness to be both technical and financial. In large ecosystem studies, up to 16% of test suites have some flakiness, and particular programming languages and platforms, like Python or Android, have up to 170 rerun failures or specialized exploration to be able to tell between flaky and failure revealing failures. In the enterprise level organizations, such as Google and CQSE indicated that 2-16% of test infrastructure and $\geqslant$2.5% of developer effort is occupied by rerun, diagnosis, and repair, which validates that uncontrollable flakiness is a direct time drain on pipeline trust and release rate, and engineering productivity.

Multi-layer root causes characterized by the synthesis of the work in this context reduce across concurrent UI behavior, dynamic version of the DOM, test data unstable, shared test data, environment variability, API latency burst, are cooperating defects, as well as absent / fallacious synchronization. The proposed multi-layer stability framework within every category has a list of suggested interventions: deterministic locators and intelligent wits on UI and mobile, resilient contracts and idempotent operations on APIs, snapshot based and synthetic datasets on pipelines, and hermetic and reproducible environments on infrastructure. Together with AI-based telemetry and probabilistic flakiness scoring and heatmap style analytics, these controls provide a way to bring the flaky test rates down to levels of 5-10% in an immature organization and below 1% in a mature organization, as well as to reduce the Mean Time to Diagnose and Mean Time to Repair by 30-50%.

Based on these findings, a working enterprise checklist can be identified. Quantitative measures of flakiness, such as but not limited to the flaky test rate, the share of flaky failures, rerun overhead, and developer time share, should be baselined by service and layer. Technical standards enforced by engineers with the support of a code review and CI policy should be deterministic locators, explicit native synchronization primitives, hardened API contracts, as well as controlled test data strategies. The organizations are proposed to launch self-healing automation and AI-assisted telemetry, with the availability of flakiness probability scores, and heatmaps to persist in triage and quarantine, and refactoring. Leadership must also establish clear SLOs, such as FTR $\leqslant$1, rerun overhead $\leqslant$5, and consist of flaky MTTR defined in hours instead of days and monitored on shared dashboards.

Non credible continuous testing and high maturity of automation in the long run necessitate the stabilization of automated tests. Well-instrumented, consistent suites of the same have been found to be able to release faster and predictably, contain defects earlier, as well as achieve safer experimentation with micro services, serverless, as well as data heavy architectures. The evidence further reviewed indicates that organizations that address the issue of flakiness as a first order stability engineering problem can regain quantifiable capacity, minimize the amounts of unaccounted work, and open CI/CD pipelines into a reliable indicator instead of a noisy bottleneck. Additional collaboration between industry and academia in datasets, benchmarks, and tools will be required so that the flakiness localization precision can be more than 90% and systemic flakiness modeling can be generalized across architectures, and stability analytics can be made part of normal engineering activity. In the absence of this kind of collaboration, flaky tests will likely remain between 2 and 3% of engineering capacity and about 5-15% of test infrastructure, limiting the quantifiable payback that many companies hope big automation programs will give over time.

The future of flakiness management is largely in further automation, further telemetry, and closer connection to reality in reliability engineering. During the next 5-10 years, enterprises will shift toward more predictive systems in which many flaky failures are automatically identified, classified, and rerouted with little human intervention. With the maturing of AI

models and the expansion of training data into the 10^6–10^7 range of the test executing input, the accuracy of flakiness localization may be expected to be above 95% so that near real time advice may be given on code changes, configuration changes, or environment remedies. Such regulatory and economic pressures will prompt organizations to institutionalize numeric stability objectives, such as a flaky test rate of less than 1%, a rerun overhead of less than 5% and a flaky MTTR of at most a few hours of test instead of days. Test stability will become the first class SRE concern, just like the other SLOs: availability and latency.

Conflict of Interest

The authors declare no conflict of interest.

Acknowledgment

The authors gratefully acknowledge the technical discussions and peer feedback that contributed to refining the proposed methodology.

References

[1] S. Habchi, G. Haben, M. Papadakis, M. Cordy, and Y. Le Traon, "A qualitative study on the sources, impacts, and mitigation strategies of flaky tests," *arXiv* preprint arXiv:2112.04919, 2021. [Online]. Available: https://arxiv.org/pdf/2112.04919.

[2] A. Tahir, S. Rasheed, J. Dietrich, N. Hashemi, and L. Zhang, "Test flakiness causes, detection, impact and responses: A multi-vocal review," *Journal of Systems and Software*, vol. 206, Art. no. 111837, 2023, doi:10.1016/j.jss.2023.111837.

[3] F. Leinen, D. Elsner, A. Pretschner, A. Stahlbauer, M. Sailer, and E. Jürgens, "Cost of flaky tests in continuous integration: An industrial case study," in *Proceedings of the 2024 IEEE International Conference on Software Testing, Verification and Validation*, 2024, pp. 329–340, doi:10.1109/ICST60714.2024.00037.

[4] O. Parry, G. M. Kapfhammer, M. Hilton, and P. McMinn, "A survey of flaky tests," *ACM Transactions on Software Engineering and Methodology*, vol. 31, no. 1, pp. 1–74, 2021, doi:10.1145/3476105.

[5] Q. Luo, F. Hariri, L. Eloussi, and D. Marinov, "An empirical analysis of flaky tests," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on the Foundations of Software Engineering*, 2014, pp. 643–653, doi:10.1145/2635868.2635920.

[6] M. Gruber, S. Lukasczyk, F. Kroiß, and G. Fraser, "An empirical study of flaky tests in Python," in *Proceedings of the 2021 IEEE International Conference on Software Testing, Verification and Validation*, 2021, pp. 148–158, doi:10.1109/ICST49551.2021.00026.

[7] A. Romano, Z. Song, S. Grandhi, W. Yang, and W. Wang, "An empirical analysis of UI-based flaky tests," in *Proceedings of the 2021 IEEE/ACM International Conference on Software Engineering*, 2021, pp. 1585–1597, doi:10.1109/ICSE43902.2021.00141.

[8] Z. Dong, A. Tiwari, X. L. Yu, and A. Roychoudhury, "Flaky test detection in Android via event order exploration," in *Proceedings of the 2021 ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021, pp. 367–378, doi:10.1145/3468264.3468584.

[9] G. Haben, S. Habchi, M. Papadakis, M. Cordy, and Y. Le Traon, "The importance of discerning flaky from fault-triggering test failures: A case study on the Chromium continuous integration," *arXiv* preprint arXiv:2302.10594, 2023. [Online]. Available: https://arxiv.org/abs/2302.10594.

[10] C. Ziftci and D. Cavalcanti, "De-flake your tests: Automatically locating root causes of flaky tests in code at Google," in *Proceedings of the 2020 IEEE International Conference on Software Maintenance and Evolution*, 2020, pp. 736–745, doi:10.1109/ICSME46990.2020.00075.

[11] M. Machalica, A. Samylkin, M. Porth, and S. Chandra, "Predictive test selection," in *Proceedings of the 2019 IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice*, 2019, pp. 91–100, doi:10.1109/ICSE-SEIP.2019.00018.

[12] P. R. Vennamaneni, "Building compliance-driven AI systems: Navigating IEC 62304 and PCI-DSS constraints," *International Journal of Network Security*, 2025.

[13] R. Khankhoje, "Strategies for mitigating flaky tests in automated environments," 2025. International Journal of Science and Research, vol. 8, no. 3, 1950-1954, 2019

[14] O. Parry, G. Kapfhammer, M. Hilton, and P. McMinn, "Systemic flakiness: An empirical analysis of co-occurring flaky test failures," *arXiv* preprint arXiv:2504.16777, 2025. [Online]. Available: https://arxiv.org/abs/2504.16777.

[15] V. K. Enugala, "BIM-to-field inspection workflows for zero paper sites," *Utilitas Mathematica*, vol. 122, no. 2, pp. 372–404, 2025.

[16] A. Gajera Jr., "Comparative analysis of Jenkins, GitLab CI, and GitHub Actions: Performance evaluation in CI/CD pipelines," Bachelor's thesis, Metropolia University of Applied Sciences, Helsinki, Finland, 2025. [Online]. Available: https://urn.fi/URN:NBN:fi:amk-202504309434.

[17] A. C. Jha, "AI-optimized spine–leaf fabrics: NVIDIA Quantum-2 vs. Cisco Nexus," *Journal of Information Systems Engineering and Management*, vol. 10, no. 60s, pp. 1209–1234, 2025.

[18] M. Jonson and S. Törnqvist, "Analyzing root causes and smells of test flakiness by simulating resource usage," unpublished manuscript, 2025.

[19] S. R. Rouholamini, M. Mirabi, R. Farazkish, and A. Sahafi, "Proactive self-healing techniques for cloud computing: A systematic review," *Concurrency and Computation: Practice and Experience*, vol. 36, no. 24, Art. no. e8246, 2024, doi:10.1002/cpe.8246.

[20] A. Khan, "What is a flaky test in software testing, and how to fix it," *Currents.dev*, Oct. 30, 2025. [Online]. Available: https://currents.dev/posts/what-is-a-flaky-test-and-how-to-fix-it.

[21] P. Gannavarapu, "Cloud infrastructure management and automation," *AJT Journal*, 2025.

[22] A. Chandrachood, "Optimizing resource allocation through telemetry-based performance monitoring," *North American Journal of Engineering Research*, vol. 4, no. 4, 2023.

[23] S. Singh, "Early-warning prediction for machine failures in automated industries using advanced machine learning techniques," unpublished manuscript, 2023.

[24] M. Y. H. Yeow, C. Y. Chong, M. K. Lim, and Y. Y. Yee, "Predicting software reuse using machine learning techniques—A case study on open-source Java software systems," *PLOS ONE*, vol. 20, no. 2, e0314512, 2025, doi:10.1371/journal.pone.0314512.

[25] O. Patlak, "Strategies to ensure software quality in existing Java applications," Doctoral dissertation, University of Applied Sciences, 2023.

[26] C. Saastamoinen, "Evaluation of machine learning models in predicting software flakiness," Master's thesis, 2024.

[27] S. Thorve, C. Sreshtha, and N. Meng, "An empirical study of flaky tests in Android apps," in *Proceedings of the 2018 IEEE International Conference on Software Maintenance and Evolution*, 2018, pp. 534–538, doi:10.1109/ICSME.2018.00062.

[28] M. R. Dhanagari, "Choosing the right NoSQL database: MongoDB vs. Aerospike for enterprise applications," SciPubHouse, 2025.

[29] G. Oliveira, "The hidden costs of flaky tests: A deep dive into test reliability," *StickyMinds*, May 5, 2025. [Online]. Available: https://www.stickyminds.com/article/hidden-costs-flaky-tests-deep-dive-test-reliability-0.

[30] M. Machalica, W. Chmiel, S. Swierc, and R. Sakevych, "Probabilistic flakiness: How do you test your tests?" *Engineering at Meta*, Dec. 10, 2020. [Online]. Available: https://engineering.fb.com/2020/12/10/developer-tools/probabilistic-flakiness/.

[31] Y. Priya, "AI meets CI/CD: Supercharging test automation for speed and reliability," 2024.

[32] S. Hashem, "Exploring confidence challenges in integrating third-party binaries in a CI/CD pipeline with limited transparency," 2025.

[33] K. Lulla, "Cross-border compliance and quality assurance in semiconductor manufacturing," *Journal of Electrical Systems*, vol. 21, no. 1s, pp. 493–511, 2025. [Online]. Available: https://journal.esrgroups.org/jes/article/view/9196.

[34] H. N. Zhu, R. M. Furth, M. Pradel, and C. Rubio-González, "From bugs to benchmarks: A comprehensive survey of software defect datasets," *arXiv* preprint arXiv:2504.17977, 2025. [Online]. Available: https://arxiv.org/abs/2504.17977

[35] Z. Sayyed, "Application-level scalable leader selection algorithm for distributed systems," *International Journal of Computational and Experimental Science and Engineering*, 2025.

[36] D. Drofa, "Integrating advanced API solutions into full-stack web and mobile applications to optimise user experience," *International Journal of Current Science Research and Review*, vol. 8, no. 5, pp. 2086–2100, 2025.

[37] V. Pontillo, F. Palomba, and F. Ferrucci, "Test code flakiness in mobile apps: The developer's perspective," *Information and Software Technology*, vol. 168, Art. no. 107394, 2024, doi:10.1016/j.infsof.2023.107394.

[38] S. Grover, S. Yadav, S. K. Tiwari, and S. Ramachandran, "Engineering robust AI products through continuous quality assurance: A framework for testing, monitoring, and validation of adaptive live learning AI/ML systems in dynamic production environments," *International Journal of Applied Mathematics*, vol. 38, no. 2s, 2025, doi:10.12732/ijam.v38i2s.710.

Received: 28 December 2025, Revised: 24 January 2026, Accepted: 01 February 2026, Online: 24 February 2026
DOI: https://dx.doi.org/10.55708/js0502003

Impact of Brainwave Entrainment using VR to Improve Attentional Learning in Children with ADHD, ASD and Comorbidity

Manasa Mandapati*, Prabhat Ranjan
DY Patil Pratishthan, School of Interdisciplinary Studies and Reasearch, DYPIU, Pune, 411044, India
Email(s): prof.prabhat.ranjan@gmail.com (P. Ranjan)
*Corresponding author: Manasa Mandapati, DYPIU, Contact No & Email: manasa.mandapati@gmail.com

ABSTRACT: Various neurological disorders (NDs) across the globe are prevalent among children, affecting their quality of life. Among them are Attention Deficit Hyperactivity Disorder (ADHD), Autism Spectrum Disorder (ASD), and comorbid conditions, which are defined by proximity of symptoms of inattention, hyperactivity, and impulsivity that are accompanied by impairment in several functional domains. The overall aim of this research is to improve attentional learning in children with NDs using brainwave entrainment techniques with cutting-edge technology like virtual reality and artificial intelligence. For photic stimulation, pulses of light (virtual reality device) were used. For audio stimulation, binaural beats (noise-cancellation headphones) were used. We identified 40 children in the age group of 6 to 12 years, diagnosed by clinicians as ADHD or ASD, or comorbid conditions. Among 40, Audio Visual Entrainment (AVE) at 10 Hz frequency was administered to 23 subjects for 20 days, with 15 minutes daily. Electroencephalogram (EEG) signals were recorded before and after stimulation using the Emotiv Epoc X device. Initially, the data was pre-processed using the Welch method and independent component analysis (programming done in Python and MATLAB). The power spectral density and the relative band power ratios were calculated. The data was analyzed using quantitative analysis, which included comparative and statistical analysis. This analysis revealed a reduction in distraction and an improvement in cognition and attention among subjects with ADHD and comorbid conditions. No improvement was observed in ASD subjects. A positive result is observed in 72% of the ADHD subjects and 66% of the comorbidity subjects, with only 16% of ASD subjects.

KEYWORDS: Electroencephalography, virtual reality, Independent component analysis, brainwave entrainment, photic entrainment, ADHD, Autism Spectrum Disorder, Comorbid

1. Introduction

Lack of attention due to neural disorders (NDs) impacts children's learning ability, communication, and behavior. Attention Deficit hyperactivity disorder (ADHD), Autism Spectrum Disorder (ASD), and comorbid conditions are the most common NDs among children [1]. According to the Diagnostic and Statistical Manual of Mental Disorders (DSM-5) (2013) [2], the children with ADHD

- Suffer from lack of attention,
- Regularly does not follow the instructions,
- Gets distracted very easily by extraneous stimuli in the envithe environmentd
- Usually forgets the daily activities.

They avoid tasks that need to sustain mental efforts and have difficulty completing problem solving tasks that require attention. A study by PilaNemutSandani [3] concluded that participants with ADHD had poor organizational and planning skills. The difficulties with executive functioning and complex reasoning is one of the reasons to believe that people with ADHD lack problem-solving skills [4].

A substantial minority of children with ADHD demonstrate ASD symptoms [5, 6]. ADHD is the most common comorbidity in children with ASD at a range of 40-70% [7, 8, 9, 10]. Consistent with the data, some researchers assert that although ADHD symptoms occur without ASD symptoms, ASD symptoms do not occur without ADHD symptoms. Most children with ASD had high levels of inattention and hyperactivity, thus introducing bias into ADHD diagnosis. The co-occurrence of the symptoms [6] made comorbidity research important in this field.

ASD is a behavioral syndrome with impairments in several areas of development including social interaction, communication, and stereotypical interests and activities (American Psychiatric Association, 2013 [2]). Autism can affect either the intelligence quotient (IQ) continuum, or the level of language function by age six [11]. These are strong predictors of clinical outcomes. Autism is invariably accompanied by language delay. People with Asperger syndrome (AS) and autism have common symptoms like social impairments and restricted behaviors. However, their verbal and nonverbal IQ will be average or above, without serious language impairment [12]. The most common treatment for ADHD is through psychostimulants such as Ritalin and Adderall which have short-term positive effects and/or severe long-term side effects [13, 14]. An

alternative to ADHD medication is brainwave entrainment, a method that naturally synchronizes the brainwaves to external periodic stimuli.

The brain can change under the influence of experience, as well as restore lost connections after damage or as a response to external influences [15] as per the phenomenon of neuroplasticity. This phenomenon can be used to develop memory, learning, attention and restore damaged cells [15]. Studies illustrate that the central nervous system was developed in a predictable sequence [15]. First is the development of senses, second is the movement/motor, and third is the mental/cognitive ability. To better understand the nature of sound/light of the first language, it would be simple to consider first language expression as music and light. Appreciating the potential of deeply rooted sensory communications that do not rely on higher cognition characteristics of mental processing, the signal approaches can be considered positive [16]. This signaling can be presented in light and sound compositional designs that permit the brain to shift in and out of degrees of stable/unstable behaviors. Studies also illustrate that light and sound brain engagement can guide the brain into beautiful positive neuroplasticity experiences [17].

There are different methods to acquire brainwave entrainment. In this research audio visual entrainment using the pulses of light and the binaural beats were chosen. Visual entrainment was done using periodic sinusoidal light pulses in a Virtual Reality (VR) environment [18, 19, 20, 21]. Photic stimulation of 10 Hz was well known to entrain the EEG signals in human beings [22] as it helps to relax the brain and improve the rewiring of cognitive styles and communicative patterns. Alpha rhythm exerts beneficial effects on cognition and behavior [23, 24, 25, 26, 27, 28, 29]. Therefore, the alpha frequency was used in the present study. Furthermore, findings from studies state that differences in working memory load have been linked to modulations in theta peaks, along with the theoretical and empirical support of the role of theta-gamma phase coupling in STM/WM capacity [30]. This suggests that the individual differences in theta frequencies (and gamma frequencies) could potentially be linked to individual differences in WM or STM capacity [31]. Binaural beats were provided using a lower frequency tone in one ear and a higher frequency tone in another ear using headphones, the superposition of these two waves of different frequencies produces a beat wave to which the brain gets synchronized [32, 33].

A single cell recording studies and neurological studies state that social intelligence was a function of three regions: the amygdala, orbitofrontal, and medial frontal cortices in which abnormalities in autism have been found. Studies in neuroimaging have implicated the Prefrontal cortex in attentional functions, in ADHD [14]. Understanding the cognitive behavior of the brain can be done using the signals or images from the brain [34]. Thus, EEG signals were captured using a consumer-grade portable 14-channel EEG device, Emotiv Epoc X, in this research to study the changes that occurred before and after the brainwave entrainment. The overall aim of this research is to improve attentional learning in children with NDs using brainwave entrainment.

2. Related Work

The change in the neural structure and function in response to experience or environmental stimuli is called neuroplasticity. Considerable changes in the brain's anatomical structure and connectivity occur during childhood [35, 36, 37]. Studies indicate that functional parcellation of cerebral cortical areas into discrete processing centers involves a wide range of signals with tangled interaction, between mechanisms both intrinsic to cortical progenitors and neurons and environmental stimuli extrinsic to the cortex which require neural activity such as light stimulation to the developing eye [38].

ADHD is one of the most heritable neuropsychiatric disorders [39, 40]. EEG recordings reveal that ADHD children produce more theta (4-7 Hz) wave activity in the brains frontal and central cortical regions [41]. ADHD children also produce less beta (13-21 Hz) wave activity [42]. Changing the cerebral electrical activity associated with ADHD has improved ADHD children's symptoms [42, 43, 44]. As an alternative treatment approach to ADHD, neurofeedback has been used to increase cerebral activity [42, 45, 46, 47, 48].

Before the DSM-5 [49] in 2013, clinicians were unable to make an ADHD diagnosis in the context of ASD. It was presumed that any symptoms of inattention and/or hyperactivity-impulsivity were secondary to ASD and not due to an additional ADHD diagnosis [50]. ASD and ADHD often co-occur [51]. Hyperactive, impulsive symptoms correlate strongly with restricted and repetitive behaviors in ASD [52] and family members of individuals with ASD have elevated rates of ADHD diagnoses [53, 54, 55] and vice versa [56].

Many researchers are testing the therapeutic effectiveness of Audio-Visual Entrainment (AVE) as a treatment for a variety of low-arousal brain disorders like ADHD. AVE is the repetitive and intermittent presentation of light and sound. AVE affects the EEG output [23, 57, 58, 59, 60, 61, 62, 63]. It has been suggested that many clinicians are using AVE informally for ADHD with anecdotal reports of successful treatment but with few published results [64, 65]. Researchers [66] conducted a pilot study using AVE to treat learning and behavioral disorders. Russell states that AVE achieves the same results as EEG biofeedback but at less cost and in less time [44]. Due to tremendous changes in technology, hardware, and software, there is a need to expand the research. A study on augmented cognition via brainwave entrainment in virtual reality by Argento E, Papagiannakis G, and his team proved a concept that the theoretical idea of fast learning in VR under certain conditions is indeed feasible and has important practical potential [67]. Nowadays, all kinds of technologies in the VR field are developing rapidly. Recently VR has been used for brainwave entrainment in many researches but they were used only in place of tasks to be done during entrainment and EEG data was not considered [68, 69, 70, 71].

Brief, intense flashes produce harmonic activity in the brain [21]; whereas, sine wave stimulation produces a sine-like response (insignificant harmonic activity). Researchers [18, 19, 20, 21] agree that sine-wave modulated light eliminates the problem of light intensity from a Xenon strobe

increasing with frequency and the harmonics generated within the neo-cortex at frequency multiples much higher than the fundamental at times. The raw EEGs shown in the studies, show neither signs of epileptiform activity nor any discussion about it. To address the concerns of eliciting a photic-induced seizure, the VR device video used had a slowed turn-on and -off time of about 15 msec. The light emitted from the VR device was white.

The purpose of this research was to expand the research on learning disorders and to further determine the effectiveness of AVE in reducing the symptoms associated with learning disorders by introducing new technologies and software for entrainment and data collection. The research also contributes in finding tremendous potential and usability of VR in brainwave entrainment.

3. Materials and methods

3.1. Ethical clearance

The methodology used in this research was approved and performed following the guidelines and regulations approved by the institutional ethics committee. The parents/legal guardians of all the participants signed the consent forms. The study took place from November 2022 to November 2023 in a separate room in a children's school environment under the observation of the researcher. Consumables like sanitizers, masks, and tissues were used as per COVID protocols. The study was accomplished using protocols established by the foremost clinicians and researchers in the field.

3.2. Data Collection

We approached approximately 200 subjects, among which 110 parents declined or were not interested to participate in the research. 90 subjects parents/legal guardians accepted to participate but only 33 finally met the inclusion and exclusion criteria and gave the consent to participate in the research. However eventually, only 23 subjects completed all the sessions including pretest survey.
Inclusion criteria

- Subjects in the age group of 6-12 years
- Clinically diagnosed: ADHD, ASD, COMORBID
- Gender eligible for study: All
- Socioeconomic background: All
- Parents/Legal guardians and subjects willing and able to give consent for taking part in the research study

Exclusion criteria

Since visual brainwave entrainment can stimulate large areas of the brain, certain people listed below should not use this technology. Those who have

- History of seizures, (especially epileptics)
- History of brain injury
- Hearing-related disease
- Any known ocular disease in one or two eyes

EEG signals were recorded before and after sessions from the 23 subjects, and demographic information of the subjects is summarized in Table 1 and Table 2. In the pretest survey, data on the diagnosis of a disorder by clinicians (pediatricians/neurologists/psychiatrists), age, sex, left or right-handed, and verification of medical diagnosis report were obtained. Based on the pretest survey, the data was subdivided into three groups. This experiment does not require a traditional control group. In the study, we compared individual A's pre-session data with the post-session data of individual A i.e. when AVE does not influence the participant and when AVE influences the participant. Thus pre sessions data is equivalent to control group.

Table 1: Demographic data of the subjects who participated in the research

S.No.	Data	No. of subjects
1	Subjects	23
2	Male	19
3	Female	4
4	Avg. age	8 years
5	LH	2
6	RH	21

Table 2: Classifications of the subjectsbased on the clinical history

S.No.	Diagnosis	No. of Subjects
1	ADHD	11
2	ASD	6
3	Comorbid	6

3.3. Device Information

The following devices were used in this research Hardware

- Emotiv Epoc X -14 channel wireless EEG headset to record the brain activity.
- Noise cancellation headphones for audio entrainment.
- Oculus Quest 2 VR headset for photic entrainment.

Software

- Emotiv Pro Suite v3.2.1 can be downloaded from https://www.emotiv.com/products/emotivpro to obtain the raw EEG data.
- Matlab 2022b from Math Works Incorporation at https://in.mathworks.com , and Python v3.33 for data analysis.
- Audacity v3.2.1 which can be downloaded from https://www.audacityteam.org for generating binaural beats.
- Adobe Premiere Pro 2020 from https://www.adobe.com/in/products/premiere.html for generating pulses of light in VR.

The software used is installed on a personal computer with the following specifications: Intel(R) Core(TM) i3-8130U CPU @ 2.20GHz 2.20 GHz, and 8 GB in RAM. The Emotiv Epoc x is paired to the system via Bluetooth and the EEG data was monitored and recorded in Emotive Pro Suite.

3.4. EEG data recording

EEG readings were recorded in the subject's school environment using 14 Ag/AgCl electrodes (AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, and AF4) and 2 references CMS/DRL references at P3/P4. These electrodes were positioned according to the 10-20 international system with the bipolar method. This system is a standard system of positioning the electrodes for reliable data comparison across all the laboratories in the world.

Entrainment is given with eyes open in a VR environment at resting condition under 10 Hz photic stimulation (white light) with brightness intensity of 200 cd/m2 along with audio stimulation using binaural beats at 10 Hz for 20 days with 15 minutes per day. The pulses of light are imposed on a video made of pictures of birds and animals. A total of 20 min EEG signals pre and post-stimulation sessions were obtained. These EEG signals were sampled at the rate of 128sps and 14-bit resolution. In addition, the signals were recorded in ASCII format.

3.5. Methodology

The block diagram of the proposed methodology is depicted in Fig. 1. Initial screening of participants took place to identify the relevant subjects and pretest was conducted. Once subjects were identified and consent was signed, initial EEG data was recorded. From the next day, entrainment sessions took place for 20 days. On the 21st day, the final EEG data was recorded. After data collection data analysis was done and the results were discussed in the results section.

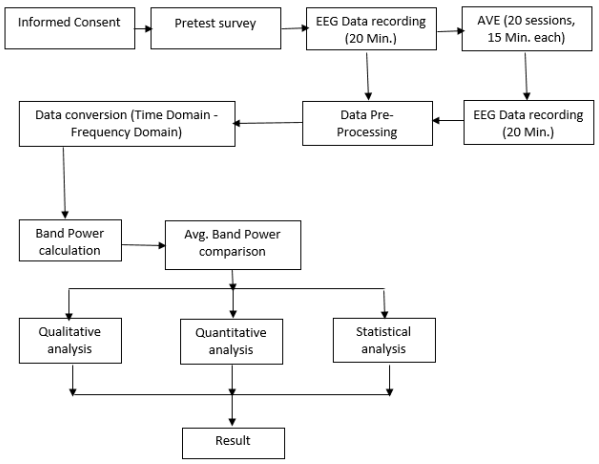

Figure 1: The block diagram of the methodology

3.5.1. AVE Sessions

Initially, EEG signals were recorded from the subjects before entrainment sessions. Then a combination of audio and photic stimulation at 10Hz (white light with intensity 200 cd/m2) with eyes open was given for 20 days (15 minutes per day). Later, EEG signals were recorded post-sessions with a sampling rate of 128sps at 14-bit resolution.

3.5.2. Data Processing

The Emotive software gives the recordings in a .csv file and edf file. Steps during data pre-processing were reading the signal stream as well as event timing information along with electrode positions, followed by filtering (0.2–12 Hz band-pass filter with Hamming window and a transition bandwidth of 4 Hz), relevant electrode selection (FC5 and FC6), discretization into epochs (from 0 until +900 msec after stimulus onset), application of Independent Component Analysis (ICA) and hybrid artifacts rejection methods (improbable data epochs of amplitudes over 25 micro V, abnormal data trends, abnormal frequency spectrum amplitudes of over ±25 dB, values higher/lower than 8 standard deviations of the data amplitude distribution, as well as visual signal inspection) followed by the calculation of grand averages across all of the epochs.

ICA has been proven to be more effective and flexible in separating EEG signals from artifacts. ICA is a linear model that decomposes the acquired sensor data into components that can subsequently be interpreted regarding the underlying physiological processes (e.g. brain, eyes, muscle, other). The effectiveness of ICA is given in a study conducted by [72]. In contrast to traditional artifact removal algorithms, various authors tested the independent component analysis (ICA) method on simulated and experimental data and showed good performance in the separation of signals from their linear mixtures and the extraction of eye information present in EEG signals [73, 74, 75]. To avoid errors introduced by manually selected components, automatic extraction and removal of eye movement artifacts were introduced after ICA analysis [76].

Band Power Separation: In EEG signal processing, separating the power spectrum density into the following bands is common. Delta (1–4 Hz), Theta (4–8 Hz), Alpha (8–12 Hz), Beta (12–25 Hz), and Gamma (25–40 Hz) using Welch method in which FFT was used with the moving Hanning window. The data is divided into K overlapping segments of length L. The Hanning window was enforced for each section. FFT was calculated in each window and PSD was computed as an average of FFTs over all windows [77]. One downside to the compactness of the headset concerning our paradigm was the lack of a Cz electrode, which would have been especially suitable for recording errors due to its proximity to the ACC, so our study focused on its two nearest available electrodes, FC5 and FC6.

The band power ratios were calculated using Eq. (1) to (9).

$$RelativeThetaPowerRT \quad {}^{n}_{i1}\Theta_i T\, xreturn \tag{1}$$

$$RelativeThetaPowerRT \quad {}^{n}_{i1}\Theta_i T \tag{2}$$

$$RelativeAlphaPowerRA \quad {}^{n}_{i1}\alpha_i T \tag{3}$$

$$RelativeBetaPowerRB \quad {}^{n}_{i1}\beta_i T \tag{4}$$

$$ThetatoBetaRatioTBR \quad {}^{n}_{i1}\Theta_i \, {}^{n}_{i1}\beta_i \tag{5}$$

$$ThetatoAlphaRatioTAR \quad {}^{n}_{i1}\Theta_i \, {}^{n}_{i1}\alpha_i \tag{6}$$

$$ThetatoAlpha\ BetaRatioTABR \quad {}^{n}_{i1}\Theta_i \, {}^{n}_{i1}\alpha_i \ \beta_i \tag{7}$$

$$BetatoAlpha\ ThetaBAT \quad {}^{n}_{i1}\beta_i \, {}^{n}_{i1}\Theta_i \ \alpha_i \tag{8}$$

$$TotalPowerT \quad {}^{n}_{i1}\Theta_i \ \alpha_i \ \beta_i \tag{9}$$

where n is the channel number,
β l = Beta low
β h = Beta high.

After calculating all the band powers and band power ratios, the values obtained before and after AVE sessions were compared and the data analysis was done. Data normalization was done using MinMax scalar and fit transform was used to fit the data and then transform the data. The amplitude spectrum of data was plotted in Fig. 2 and PSD was plotted in Fig. 3.

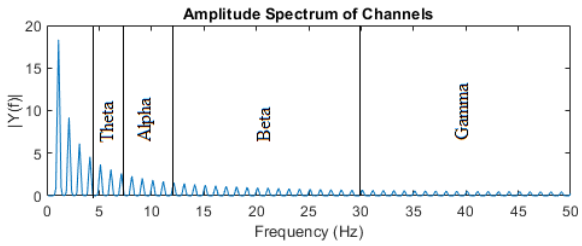

Figure 2: Amplitude spectrum of the data of ADHD subject 2

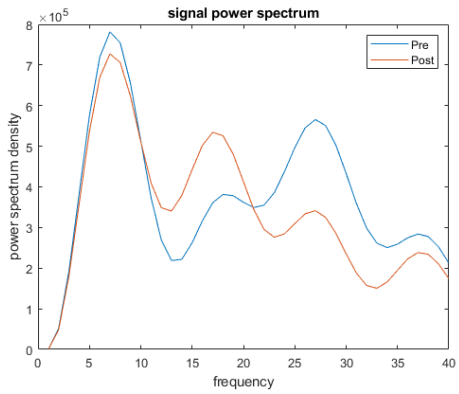

Figure 3: Power spectrum of the data of ADHD subject 2

The PSD in Fig. 3 indicates that the Theta frequency (4-8 Hz) was suppressed and Beta low frequency (12-18 Hz) was invoked. Increased beta levels to theta and alpha indicate improvement in attention levels.

4. Data Analysis

Initially based on the clinical data, the subjects were subdivided into three categories ADHD, ASD, and Comorbid. To understand the brain activity of the subject, the EEG data from front central electrodes FC5 and FC6 was computed before and after the AVE sessions and the topographical plot was plotted in Fig. 4 to Fig. 6. The analysis of the brainwave pattern is done using Python v3.11 and Matlab 2023b, The Math Works Incorporation.

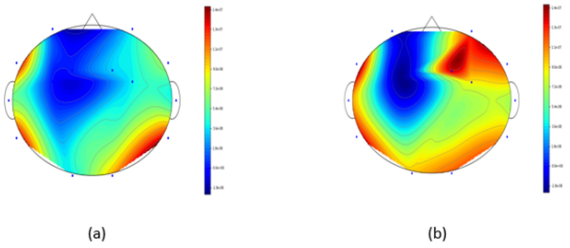

Figure 4: The topographical plot of ADHD subject 2 (a) Pre and (b) Post entrainment sessions.The Δ shows the frontal side of the brain .

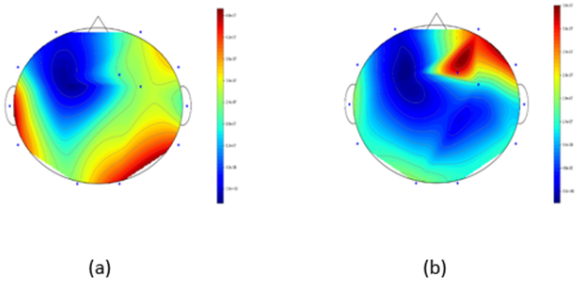

Figure 5: The topographical plot of ASD subject (a) Pre and (b) Post entrainment sessions.The Δ shows the frontal side of the brain.

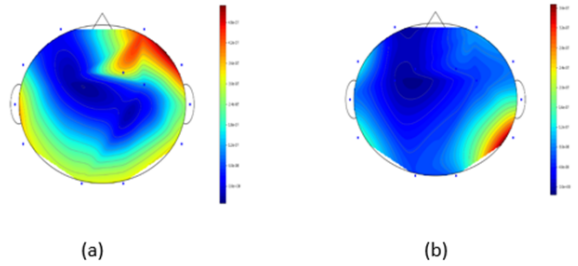

Figure 6: The topographical plot of Comorbid subject (a) Pre and (b) Post entrainment sessions. The Δ shows the frontal side of the brain

The above topographical plot in Fig. 4 shows that the intensity at the frontal region that is responsible for spatial learning is active post AVE sessions. Similar kind of result can be seen in ASD plot in Fig. 5 as well but in Comorbid condition in Fig. 6 the intensity level was reduced post AVE sessions representing the calmness in the brain region. This indicates that the attention and spatial learning in case of subjects with comorbid conditions did not improve as for those subjects with ADHD.

4.1. Quantitative (Comparative) Analysis of Data

Relative band power at Theta, Alpha, and Beta frequency bands was computed using the formulae given in Eq. (2) to Eq. (9) and the comparative analysis is done between pre and post session data. The band power ratios were considered as features that describe the change in the

brain pattern before and after the AVE sessions. The calculated values of Relative Theta (RT), Relative Alpha (RA), Theta to Beta (TBR), Theta to Alpha Beta (TABR) values was expected to reduce after brainwave entrainment sessions and at the same time Relative Beta (RB) and Beta to Theta Alpha ratio (BTAR) values were expected to increase after the entrainment sessions for ADHD, ASD and Comorbid conditions. The values that indicate the change between pre and post session data are given below in Table 3, Table 4 and Table 5. The calculations were done using Python version 3.11. The values in the grey colors indicate that the change was as expected as per the hypothesis of the research while the values in white color represent that the change was not as expected.

Table 3: Change in feature parameters before and after AVE sessions in ADHD subjects

ADHD	RT	RA	RB	TB	TAB	BTA
1	-0.11	0.38	0.07	-0.17	-0.25	0.08
2	-0.02	0.02	0.04	-0.06	0.03	0.05
3	-0.07	-0.02	0.33	-0.30	-0.11	0.41
4	0.24	-0.15	-0.35	0.91	0.75	-0.41
5	-0.03	0.07	-0.05	0.03	-0.03	-0.06
6	-0.03	0.04	0.07	-0.09	0.00	0.08
7	0.05	-0.07	-0.02	0.07	0.23	-0.03
8	0.31	-0.26	-0.47	1.46	1.18	-0.53
9	-0.04	0.01	0.14	-0.16	-0.06	0.16
10	0.01	-0.03	0.01	0.01	0.14	0.01
11	-0.03	-0.20	0.73	-0.44	-0.07	0.90

Table 4: Change in feature parameters before and after AVE sessions in Comorbid (ADHD+ASD) subjects.

Com	RT	RA	RB	TB	TAB	BTA
12	-0.07	0.08	0.24	-0.25	-0.07	0.29
13	-0.08	0.12	0.32	-0.30	-0.17	0.36
14	0.03	0.01	-0.18	0.26	0.18	-0.20
15	0.04	-0.08	-0.17	0.25	0.23	-0.18
16	-0.20	0.16	0.59	-0.50	-0.34	0.75
17	-0.04	0.08	0.02	-0.06	-0.01	0.02

Table 5: Change in feature parameters before and after AVE sessions in ASD subjects

ASD	RT	RA	RB	TB	TAB	BTA
18	0.21	-0.20	-0.46	1.23	0.77	-0.50
19	0.02	0.05	-0.18	0.24	0.09	-0.20
20	0.19	-0.03	-0.35	0.83	0.44	-0.40
21	0.00	-0.01	0.05	-0.05	0.04	0.05
22	0.56	-0.29	-0.61	3.00	1.90	-0.67
23	0.11	-0.24	-0.15	0.32	0.51	-0.17

The above ADHD subject data in Table 3 represents that the relative theta power is suppressed in 8 subjects 1,2,3,4,5,6,9 and 11 and beta power is invoked to a little extent in 7 subjects 1,2,3,6,9,10,11. The Theta to Beta ratio reduced in 6 subjects 1,2,3,6,9,11 out of 11 ADHD subjects which is represented in graphical form in Fig. 7. The pre and post theta to beta ratio data is given in Fig. 10.The ratio of Beta to Theta and Alpha increased in 7 subjects 1,2,3,6,9,10,11. This data represents a positive change in 8 subjects and a negative change in 3 subjects in case of ADHD. Table 4 represents the comorbid subject data in which the relative theta power is suppressed in 4 subjects 12,13,16,17 and relative beta power is invoked in 4 subjects 12,13,16,17. The same is depicted in bar plot in Fig. 8. The pre and post theta to beta ratio data is given in Fig. 11. Table 5 contains the ASD subject data in which relative theta is not reduced and relative beta value is invoked in 1 subject 21 . The graphical representation of the data is given in Fig. 9. The pre and post theta to beta ratio data is given in Fig. 12.

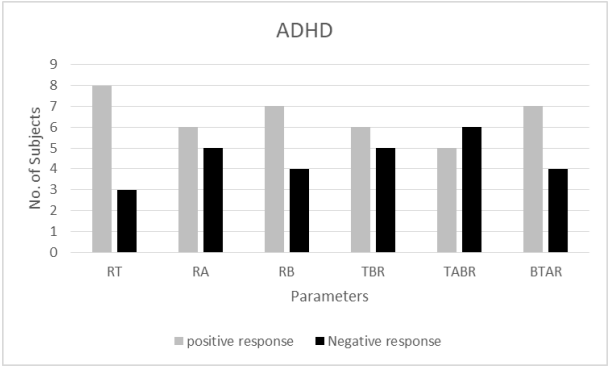

Figure 7: Graphical representation of the number of subjects who responded positively and negatively to the AVE sessions

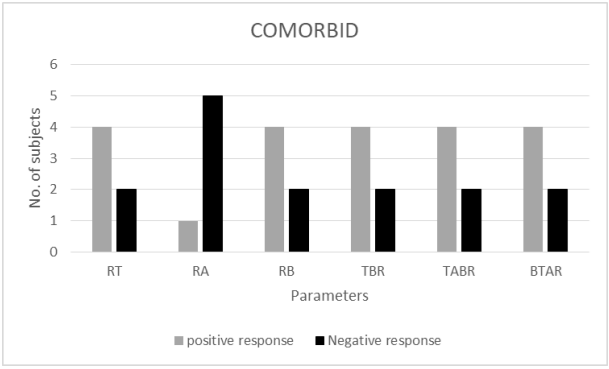

Figure 8: Graphical representation of the number of subjects who responded positively and negatively to the AVE sessions.

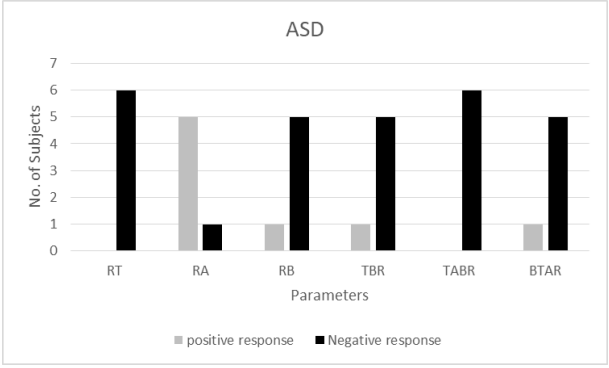

Figure 9: Graphical representation of the number of subjects who responded positively and negatively to the AVE sessions.

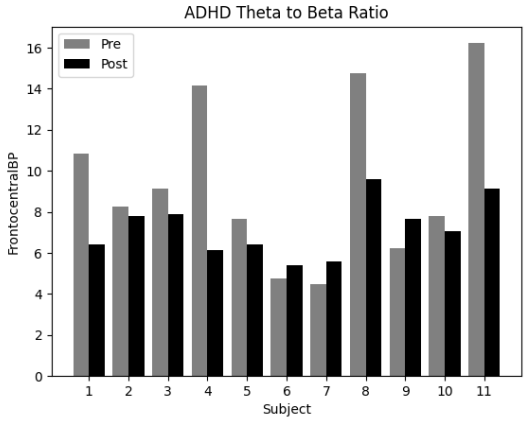

Figure 10: Bar plot of Theta to Beta Ratio before (Pre) and after (Post) the AVE sessions in ADHD subjects.

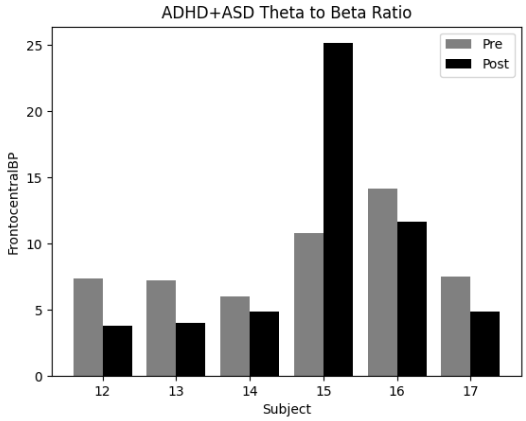

Figure 11: Bar plot of Theta to Beta Ratio before (Pre) and after (Post) the AVE sessions in Comorbidity subjects.

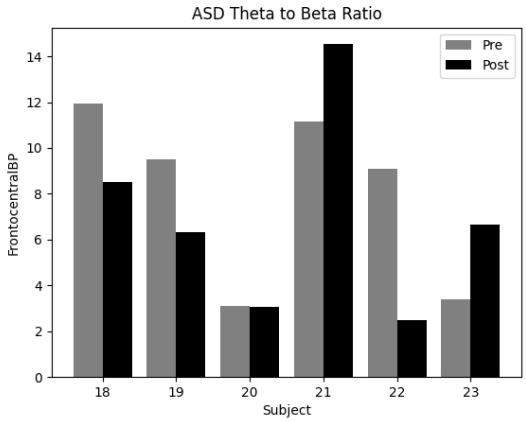

Figure 12: Bar plot of Theta to Beta Ratio before (Pre) and after (Post) the AVE sessions in ASD subjects.

This analysis reveals that 72% of subjects showed a positive change in the case of ADHD, 66% of subjects showed a positive change in the case of Comorbid conditions, and 16% of subjects showed a positive change in the case of ASD.

This experiment does not require a control group in the traditional sense. In the study, we compared individual A's pre-session data with the post-session data of individual A i.e. when the participant is not influenced by AVE and when the participant is influenced by AVE. Thus pre session data is equivalent to the control group.

4.2. Statistical analysis

The outliers in the data were screened using outlier detection algorithm [78]. The test scores were obtained by calculating the standard deviation. The test scores 2.5 times the standard deviation were considered outliers and exempted from further analysis. The box plot shows the outliers in Fig. 13. The results indicate two outliers. One from ADHD subgroup and one from ASD subgroup. The total number of participants for statistical analysis included 21 participants. Participants with good improvement in scores were contacted to check for the possible effects.

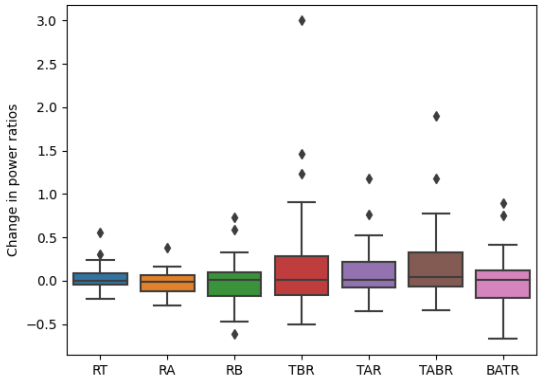

Figure 13: Box plot to detect outliers.

4.2.1. Brainwave Parameters

Since the null correlation is zero, we used the t distribution to test the correlation. The correlation's distribution is not symmetrical when r is not equal to 0, hence we used the Z distribution over Fisher transformation to create the confidence interval. the scatter plots for the parameters are given in Fig. 14 to Fig. 20

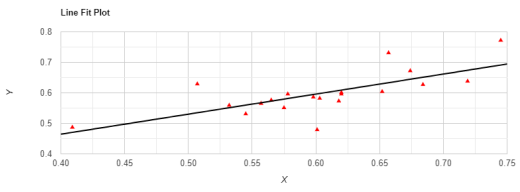

Figure 14: Line fit scatter plot for Relative Theta

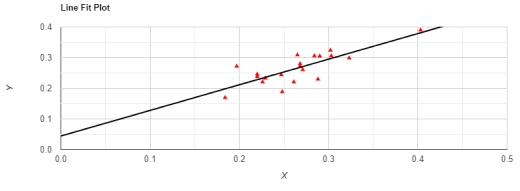

Figure 15: Line fit scatter plot for Relative Alpha

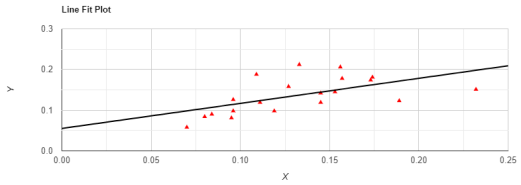

Figure 16: Line fit scatter plot for Relative Beta

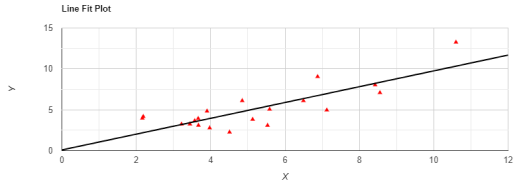

Figure 17: Line fit scatter plot for Theta to Beta ratio

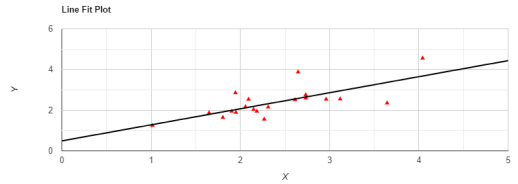

Figure 18: Line fit scatter plot for Theta to Alpha Ratio

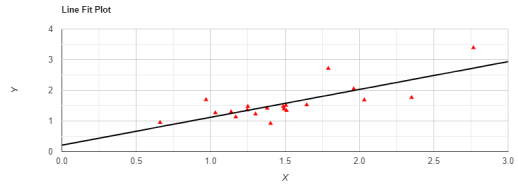

Figure 19: Line fit scatter plot for Theta to Beta and Alpha Ratio

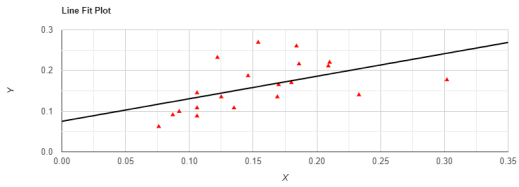

Figure 20: Line fit scatter plot for Beta to Theta and Alpha ratio

The Pearson correlation results in Table 6 indicated that there is a significant positive relationship between X (pre-session data) and Y (post-session data) in Fig. 11 to Fig. 17. The P value state that the difference between the sample correlation and the expected correlation is good enough to be statistically significant.

Table 6: Correlation between Brainwave ratios. * Indicates p less than 0.01

S. No	Parameter (Pre and Post sessions)	Pearson correlation coefficient (r)	Statistic	P-value
1	**RT**	0.7149	4.3381	0.0004*
2	**RA**	0.7971	5.6012	0.00003*
3	**RB**	0.5779	3.0043	0.00762*
4	**TBR**	0.821	6.1005	0.00001*
5	**TAR**	0.7204	4.4073	0.00034*
6	**TABR**	0.7684	5.0942	0.00008*
7	**BTAR**	0.5238	2.6091	0.01775*

5. Results and Discussion

The power spectral density is one of the important factors that indicated the changes that occurred in the brainwave frequency before and after entrainment sessions. The quantitative analysis revealed a positive result in 7 subjects out of 11 subjects based on EEG data which subjects to nearly 72% of the positive result. In the Comorbid category, the feedback from guardians revealed a positive result in 5 subjects out of 6 subjects. In contrast, quantitative analysis revealed that positive results were observed in 4 subjects out of 6 subjects based on EEG data which resulted in positive results in nearly 66% of the subjects. In the case of ASD, the guardian's feedback revealed a positive result in 2 subjects out of 6 subjects whereas the quantitative analysis revealed that positive results were observed in 1 subject out of 6 subjects based on EEG data that subjects to positive results in 16% of subjects. The same is depicted in Fig. 21. The feedback from the guardians of the participants was taken in different forms like messages, emails, word of mouth, etc. An in-depth case study approach and feedback using constructive tools for the clinical participants would have comprehensively elaborated the association of brainwaves with the clinical symptoms of NDs.

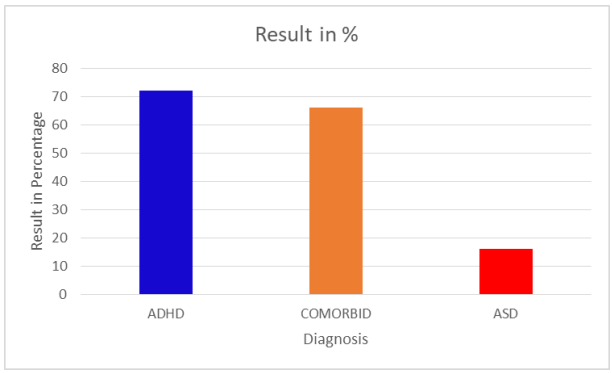

Figure 21: Comparison of effectiveness of AVE of the 3 subgroups.

Overall considerations reveal that 17 out of 23 subjects observed positive results which is 73.9% in the case of behavioral analysis and 12 out of 23 subjects observed positive results which is 52.1% in the case of quantitative analysis. The statistical analysis also shows a positive correlation between pre and post-session data supporting the research. This research also found that there is a reduction in distraction and an improvement in cognitive power which is one

of the most important factors to improve learning but still needs more investigation.

There was a lack of significant improvement in ASD. The reason behind this is the differences in neuroanatomy, functioning, and connectivity within the wider neural systems that probably mediate autistic symptoms and traits. However, the brain development in ASD is complex and is mediated by many genetic and environmental factors and their interactions [79]. Future work can explore the potential strategies to improve the results.

The differences in brainwave band power were analyzed to explore the role of change of theta, alpha, and beta bands post-entrainment sessions. The role of alpha-band oscillations was emphasized as a mechanism for selective attention due to their ability to process potential distracting stimuli [80], recent studies have challenged this theory [81]. Another study highlighted the interactions between theta and alpha/beta responses for visual selective attention among healthy individuals [82]. Further, [83] found the role of the beta frequency band's rhythmic activity in the neuronal correlates of an individual's attentional processes. High theta, low beta ADHD profile is well established in literature [84, 85, 86]. This suggests that an increase in TBR values will result in decreased attention.

Furthermore, in the current study, the individual's theta and alpha PSD affected the alpha activity during the alpha entrainment sessions. Previous studies stated the role of alpha resting state PSD in determining the changes in central alpha asymmetry during alpha and beta entrainment sessions [72]. The current study supports the role of theta and alpha baseline PSD in determining the effects of alpha entrainment. Hence, an individual's baseline frequencies could moderate the entrainment effects leading to individual differences.

Certain limitations need to be considered. The results were based on a small dataset, a larger dataset will give more accurate results. The results may vary when subjects with different dietary and lifestyle habits are considered. The observation time was limited to 20 days, and changes in the long run were unknown. Another limitation is that the scalp EEG is the result of the joint activity from multiple neurons, therefore, separating the source signals when the number of sources is greater than the number of sensors is promising. Another limitation is EEG quality degrades if the subject is not steady. So only the activities that can be done in a sitting position were given during EEG data recording.

6. Conclusion

This research provided evidence not only in the reduction of symptoms of the NDs but also an improvement in attention using audio photic brainwave entrainment. Though several studies show that AVE is a useful tool for treating attentional disorders, the introduction of VR for photic stimulation and portable commercial EEG devices, made it usable out of the lab, within the comfort zone of the subject. Food habits and gut health may also contribute to the improvement in the subject's attention and learning which needs to be considered in further research work. Based on individual psychological health, personalized programmable devices can be developed further. This research also affects the symptoms of Stuttering, Dyslexia, and Neuro developmental delay but needs more investigation. Further research can also explore different approaches or therapies that may be more effective for ASD group.

The study supports the use of technologically assistance to help children with ADHD maintain focus and manage transitions between activities. This can enable educators to personalize instruction and pacing without disrupting overall classroom. The results have important implications for psychosocial interventions for children with neural disorders by emphasizing the need for individualized, developmentally appropriate, and engagement driven approaches. The findings suggest that technology assisted therapies can improve accessibility and scalability of mental health services. The study highlights the shift from traditional, clinical centric models towards more adaptive and patient catered care.Future studies can involve in increasing the data set with diverse participant samples to improve the generalizability of findings across different age groups. Longitudinal study designs would help determine the stability and long-term impact of the observed outcomes.

Acknowledgments

This work was supported by the D. Y. Patil International University (DYPIU). We thank DYPIU, Akurdi, Pune, Maharashtra, India for providing financial assistance to conduct this research. We thank all the subjects who participated in this research.

Consent

The research is approved by the Ethics Committee of DYPIU, Akurdi, Pune, Maharashtra, India. All the participants in the research signed an informed consent form before they participated in the research.

Data Availability

The raw data is the primary data and cannot be provided publicly to protect the privacy and preservation of the anonymity of the participant, but the data and study analysis code are available from the corresponding author on a reasonable request. The band power data are provided in the supplementary information file.

References

[1] L. Ghirardi, Q. Chen, Z. Chang, R. Kuja-Halkola, C. Skoglund, P. D. Quinn, B. M. D'Onofrio, H. Larsson, "Use of medication for attention-deficit/hyperactivity disorder and risk of unintentional injuries in children and adolescents with co-occurring neurodevelopmental disorders", *Journal of Child Psychology and Psychiatry*, vol. 61, no. 2, pp. 140–147, 2020, doi:10.1111/jcpp.13136.

[2] A. Lolk, "Neurokognitive lidelser", "Diagnostic and Statistical Manual of Mental Disorders", American Psychiatric Association, 2013.

[3] R. G. Pila-Nemutandani, A. Meyer, "Behaviour planning and problem solving deficiencies in children with symptoms of attention deficit hyperactivity disorder from the balobedu culture, limpopo province, south africa", *Journal of Child & Adolescent Mental Health*, vol. 28, no. 2, pp. 109–121, 2016, doi:10.2989/17280583.2016.1200582.

[4] R. N. Elisa, B. A. Parris, "The relationship between core symptoms of ADHD and the cognitive reflection test in a non-clinical sample", *Cognitive Neuropsychiatry*, vol. 20, no. 5, pp. 416–423, 2015, doi:10.1080/13546805.2015.1068687.

[5] A. Kotte, G. Joshi, R. Fried, M. Uchida, A. Spencer, K. Y. Woodworth, J. Biederman, "Autistic traits in children with and without ADHD", *Pediatrics*, vol. 132, no. 3, pp. e612–e622, 2013, doi:10.1542/peds.2012-3947.

[6] M. Cooper, J. Martin, K. Langley, M. Hamshere, A. Thapar, "Autistic traits in children with ADHD index clinical and cognitive problems", *European Child & Adolescent Psychiatry*, vol. 23, pp. 23–34, 2014, doi:10.1007/s00787-013-0398-6.

[7] G. Joshi, S. V. Faraone, J. Wozniak, L. Tarko, R. Fried, M. Galdo, J. Biederman, "Symptom profile of ADHD in youth with high-functioning autism spectrum disorder: a comparative study in psychiatrically referred populations", *Journal of Attention Disorders*, vol. 21, no. 10, pp. 846–855, 2017, doi:10.1177/1087054714543368.

[8] G. Joshi, S. V. Faraone, J. Wozniak, C. Petty, R. Fried, M. Galdo, J. Biederman, "Examining the clinical correlates of autism spectrum disorder in youth by ascertainment source", *Journal of Autism and Developmental Disorders*, vol. 44, pp. 2117–2126, 2014, doi:10.1007/s10803-014-2063-4.

[9] F. Salazar, G. Baird, S. Chandler, E. Tseng, T. O'sullivan, P. Howlin, E. Simonoff, "Co-occurring psychiatric disorders in preschool and elementary school-aged children with autism spectrum disorder", *Journal of Autism and Developmental Disorders*, vol. 45, pp. 2283–2294, 2015, doi:10.1007/s10803-015-2361-5.

[10] A. J. Kaat, K. D. Gadow, L. Lecavalier, "Psychiatric symptom impairment in children with autism spectrum disorders", *Journal of Abnormal Child Psychology*, vol. 41, pp. 959–969, 2013, doi:10.1007/s10802-013-9739-7.

[11] P. Szatmari, S. E. Bryson, M. H. Boyle, D. L. Streiner, E. Duku, "Predictors of outcome among high functioning children with autism and asperger syndrome", *Journal of Child Psychology and Psychiatry*, vol. 44, no. 4, pp. 520–528, 2003, doi:10.1111/1469-7610.00141.

[12] L. Wing, "Asperger's syndrome: a clinical account", *Psychological Medicine*, vol. 11, no. 1, pp. 115–129, 1981, doi:10.1017/S0033291700053332.

[13] L. Brothers, B. Ring, A. Kling, "Response of neurons in the macaque amygdala to complex social stimuli", *Behavioural Brain Research*, vol. 41, no. 3, pp. 199–213, 1990, doi:10.1016/0166-4328(90)90108-Q.

[14] S. Tafazoli, J. O'Neill, A. Bejjani, R. Ly, N. Salamon, J. T. McCracken, J. G. Levitt, "1h MRSI of middle frontal gyrus in pediatric ADHD", *Journal of Psychiatric Research*, vol. 47, no. 4, pp. 505–512, 2013, doi:10.1016/j.jpsychires.2012.11.011.

[15] J. Shaffer, "Neuroplasticity and clinical practice: Building brain power for health", *Frontiers in Psychology*, vol. 7, p. 1118, 2016, doi:10.3389/fpsyg.2016.01118.

[16] V. F. da Silva, A. P. Ribeiro, V. A. Dos Santos, A. E. Nardi, A. L. King, M. R. Calomeni, "Stimulation by light and sound: Therapeutics effects in humans", *Clinical Practice & Epidemiology in Mental Health*, vol. 11, pp. 150–154, 2015, doi:10.2174/1745017901511010150.

[17] N. Jirakittayakorn, Y. Wongsawat, "A novel insight of effects of a 3-hz binaural beat on sleep stages during sleep", *Frontiers in Human Neuroscience*, vol. 12, p. 387, 2018, doi:10.3389/fnhum.2018.00387.

[18] D. Donker, L. Njio, W. Storm Van Leeuwen, G. Wieneke, "Interhemispheric relationships of responses to sine wave modulated light in normal subjects and patients", *EEG and Clinical Neurophysiology*, vol. 44, pp. 479–489, 1978.

[19] D. Regan, "Some characteristics of average steady state and transient responses evoked by modulated light", *EEG and Clinical Neurophysiology*, vol. 20, pp. 238–248, 1965.

[20] R. Townsend, "A device for generation and presentation of modulated light stimuli", *Electroencephalography and Clinical Neurophysiology*, vol. 34, pp. 97–99, 1973.

[21] L. Van Der Tweel, H. Lunel, "Human visual responses to sinusoidally modulated light", *Encephalography and Clinical Neurophysiology*, vol. 18, pp. 587–598, 1965.

[22] R. Phogat, P. Parmananda, A. Prasad, "Intensity dependence of sub-harmonics in cortical response to photic stimulation", *Journal of Neural Engineering*, vol. 19, no. 4, p. 046026, 2022, doi:10.1088/1741-2552/ac817f.

[23] W. G. Walter, "Colour illusions and aberrations during stimulation by flickering light", *Nature*, vol. 177, p. 710, 1956, doi:10.1038/177710A0.

[24] T. L. Huang, C. Charyton, "A comprehensive review of the psychological effects of brainwave entrainment", *Alternative Therapies in Health and Medicine*, vol. 14, no. 5, pp. 38–50, 2008.

[25] T. Nomura, K. Higuchi, H. Yu, S.-i. Sasaki, S. Kimura, H. Itoh, *et al.*, "Slow-wave photic stimulation relieves patient discomfort during esophagogastroduodenoscopy", *Journal of Gastroenterology and Hepatology*, vol. 21, no. 1, pp. 54–58, 2006, doi:10.1111/j.1440-1746.2005.04204.x.

[26] H. C. Ossebaard, "Stress reduction by technology? an experimental study into the effects of brainmachines on burnout and state anxiety", *Applied Psychophysiology and Biofeedback*, vol. 25, no. 2, pp. 93–101, 2000, doi:10.1023/A:1009514824951.

[27] J. Williams, D. Ramaswamy, A. Oulhaj, "10 hz flicker improves recognition memory in older people", *BMC Neuroscience*, vol. 7, p. 21, 2006, doi:10.1186/1471-2202-7-21.

[28] J. P. Rosenfeld, A. M. Reinhart, S. Srivastava, "The effects of alpha (10-hz) and beta (22-hz) "entrainment" stimulation on the alpha and beta EEG bands: individual differences are critical to prediction of effects", *Applied Psychophysiology and Biofeedback*, vol. 22, no. 1, pp. 3–20, 1997, doi:10.1023/A:1026233624772.

[29] J. H. Williams, "Frequency-specific effects of flicker on recognition memory", *Neuroscience*, vol. 104, no. 2, pp. 283–286, 2001, doi:10.1016/S0306-4522(00)00579-0.

[30] J. E. Lisman, M. A. P. Idiart, "Storage of 7 +/- 2 short-term memories in oscillatory subcycles", *Science*, vol. 267, no. 5203, pp. 1512–1515, 1995, doi:10.1126/science.7878473.

[31] P. Sauseng, W. Klimesch, K. Heise, "Brain oscillatory substrates of visual short-term memory capacity", *Current Biology*, vol. 19, pp. 1846–1852, 2009, doi:10.1016/j.cub.2009.08.062.

[32] G. Oster, "Auditory beats in the brain", *Scientific American*, vol. 229, no. 4, pp. 94–102, 1973, doi:10.1038/scientificamerican1073-94.

[33] M. Mandapati, P. Ranjan, "Virtual reality based audio visual brainwave entrainment to improve learning in children with attention deficit hyperactive disorder", *Applied Neuropsychology: Child*, pp. 1–15, 2025, doi:10.1080/21622965.2025.2455102.

[34] M. Mandapati, P. Ranjan, "Re-engineering of brainwaves to increase resistance to distractions in young adults and promote optimal learning", "2024 6th International Conference on Electrical, Control and Instrumentation Engineering (ICECIE)", pp. 1–7, IEEE, 2024, doi:10.1109/ICECIE63774.2024.10815674.

[35] P. R. Huttenlocher, "Synaptic density in human frontal cortex—developmental changes and effects of aging", *Brain Research*, vol. 163, no. 2, pp. 195–205, 1979, doi:10.1016/0006-8993(79)90349-4.

[36] d. C. C. Huttenlocher PR, "The development of synapses in striate cortex of man", *Hum Neurobiol*, vol. 6, pp. 1–9, 1987.

[37] T. K. Casey BJ, Giedd JN, "Structural and functional brain development and its relation to cognitive development", *Biol Psychol*, vol. 54, pp. 241–257, 2000.

[38] R. J. Sur M, "Patterning and plasticity of the cerebral cortex", *Science*, vol. 310, pp. 805–810, 2005.

[39] F. LEVY, D. A. HAY, M. McSTEPHEN, C. WOOD, I. WALDMAN, "Attention-deficit hyperactivity disorder: A category or a continuum? genetic analysis of a large-scale twin study", *Journal of the American Academy of Child & Adolescent Psychiatry*, vol. 36, no. 6, pp. 737–744, 1997, doi:https://doi.org/10.1097/00004583-199706000-00009.

[40] F. SV, "Genetics of childhood disorders: Xx", *ADHD, Part 4: is ADHD genetically heterogeneous? J Am Acad Child Adolesc Psychiatry*, vol. 39, pp. 1455–1457, 2000.

[41] C. A. Mann, J. F. Lubar, A. W. Zimmerman, C. A. Miller, R. A. Muenchen, "Quantitative analysis of EEG in boys with attention-deficit-hyperactivity disorder: Controlled study with clinical implications", *Pediatric Neurology*, vol. 8, pp. 30–36, 1992.

[42] J. F. Lubar, "Discourse on the development of EEG diagnostics and biofeedback for attention-deficit/hyperactivity disorder", *Biofeedback and Self-Regulation*, vol. 16, pp. 201–225, 1991.

[43] C. P. Utter, "A controlled study of the effects of neurofeedback training on iq and eeg patterns for add subjects", 1996.

[44] H. L. Russell, "Intellectual, auditory and photic stimulation and changes in functioning in children and adults", *Biofeedback, 25(1), 16-17*, vol. 23, p. 24, 1997.

[45] J. F. Lubar, W. M. Deering, *Behavioral Approaches to Neurology*, Academic Press, New York, 1981.

[46] J. O. Lubar, J. F. Lubar, "Electroencephalographic biofeedback of SMR and beta for treatment of attention deficit disorders in a clinical setting", *Biofeedback and Self-Regulation*, vol. 9, pp. 1–23, 1984.

[47] J. F. Lubar, M. N. Shouse, "Use of biofeedback in the treatment of seizure disorders and hyperactivity", B. B. Lahey, A. E. Kazdin, eds., "Advances in Child Clinical Psychology", Plenum Press, New York, 1977.

[48] M. A. Tansey, "Righting the rhythms of reason, EEG biofeedback training as a therapeutic modality in a clinical office setting", *Medical Psychotherapy*, vol. 3, pp. 57–68, 1990.

[49] A. P. Association, *Diagnostic and Statistical Manual of Mental Disorders*, American Psychiatric Association, Washington, DC, 2000.

[50] A. P. Association, *Diagnostic and Statistical Manual of Mental Disorders*, American Psychiatric Association, Washington, DC, 1994.

[51] L. Brookman-Frazee, N. Stadnick, C. Chlebowski, M. Baker-Ericzén, W. Ganger, "Characterizing psychiatric comorbidity in children with autism spectrum disorder receiving publicly funded mental health services", *Autism*, vol. 22, no. 8, pp. 938–952, 2018, doi:10.1177/1362361317712650.

[52] L. Ghirardi, I. Brikell, R. Kuja-Halkola, C. M. Freitag, B. Franke, P. Asherson, H. Larsson, "The familial co-aggregation of ASD and ADHD: a register-based cohort study", *Molecular psychiatry*, vol. 23, no. 2, pp. 257–262, 2018, doi:10.1038/mp.2017.17.

[53] L. Ghirardi, E. Pettersson, M. J. Taylor, C. M. Freitag, B. Franke, P. Asherson, R. Kuja-Halkola, "Genetic and environmental contribution to the overlap between ADHD and ASD trait dimensions in young adults: a twin study", *Psychological Medicine*, vol. 49, no. 10, pp. 1713–1721, 2019, doi:10.1017/S003329171800243X.

[54] E. Magallon-Neri, D. Vila, K. Santiago, P. Garcia, G. Canino, "The prevalence of psychiatric disorders and mental health services utilization by parents and relatives living with individuals with autism spectrum disorders in puerto rico", *The Journal of Nervous and Mental Disease*, vol. 206, no. 4, pp. 226–230, 2018, doi:10.1097/NMD.0000000000000760.

[55] M. Septier, H. Peyre, F. Amsellem, A. Beggiato, A. Maruani, M. Poumeyreau, R. Delorme, "Increased risk of ADHD in families with ASD", *European Child & Adolescent Psychiatry*, vol. 28, pp. 281–288, 2019, doi:10.1007/s00787-018-1206-0.

[56] E. Jokiranta-Olkoniemi, K. Cheslack-Postava, P. Joelsson, A. Suominen, A. S. Brown, A. Sourander, "Attention-deficit/hyperactivity disorder and risk for psychiatric and neurodevelopmental disorders in siblings", *Psychological medicine*, vol. 49, no. 1, pp. 84–91, 2019, doi:10.1017/S0033291718000521.

[57] J. Toman, "Flicker potentials and the alpha rhythm in man", *Journal of Neurophysiology*, vol. 4, pp. 51–61, 1940.

[58] J. S. Barlow, "Rhythmic activity induced by photic stimulation in relation to intrinsical activity of the brain in man", *Electroencephalography and Clinical Neurophysiology*, vol. 12, pp. 317–326, 1960.

[59] T. Inouye, N. Sumitsuji, K. Matsumoto, "EEG changes induced by light stimuli modulated with the subject's alpha rhythm", *Electroencephalography and Clinical Neurophysiology*, vol. 49, pp. 135–142, 1979.

[60] J. Kinney, C. McKay, A. Mensch, S. Luria, "Visual evoked responses elicited by rapid stimulation", *EEG and Clinical Neurophysiology*, vol. 34, pp. 7–13, 1972.

[61] T. Nogawa, K. Katayama, Y. Tabata, T. Ohshio, T. Kawahara, "Changes in amplitude of the EEG induced by a photic stimulus", *Electroencephalography and Clinical Neurophysiology*, vol. 40, pp. 78–88, 1976.

[62] R. P. Lesser, H. Lüders, G. Klem, D. S. Dinner, "Visual potentials evoked by light-emitting diodes mounted in goggles", *Cleveland Clinic Quarterly*, vol. 52, pp. 223–228, 1986.

[63] J. Frederick, J. L. Lubar, H. Rasey, S. Brim, J. Blackburn, "Effects of 18.5 hz auditory and visual stimulation on EEG amplitude at the vertex", *Journal of Neurotherapy*, vol. 3, no. 3, pp. 23–27, 1999.

[64] D. Noton, "Pms, EEG, and photic stimulation", 1996.

[65] R. Meppelink, E. I. de Bruin, S. M. Bögels, "Meditation or medication? mindfulness training versus medication in the treatment of childhood ADHD: a randomized controlled trial", *BMC psychiatry*, vol. 16, pp. 1–16, 2016, doi:10.1186/s12888-016-0978-3.

[66] J. L. Carter, H. L. Russell, "A pilot investigation of auditory and visual entrainment of brain wave activity in learning disabled boys", *Texas Researcher, Journal of the Texas Center for Educational Research*, vol. 4, pp. 65–73, 1993.

[67] E. Argento, G. Papagiannakis, E. Baka, "Augmented cognition via brainwave entrainment in virtual reality: An open, integrated brain augmentation in a neuroscience system approach", *Augmented Human Research*, vol. 2, p. 3, 2017, doi:10.1007/s41133-017-0005-3.

[68] S. J. Halpin, N. K. Tang, A. J. Casson, A. K. Jones, R. J. O'Connor, M. Sivan, "User experiences of pre-sleep sensory alpha brainwave entrainment for people with chronic pain and sleep disturbance", *Pain Management*, vol. 13, no. 5, pp. 259–270, 2023, doi:10.2217/pmt-2022-0083.

[69] R. J. Addante, M. Yousif, R. Valencia, C. Greenwood, R. Marino, "Boosting brain waves improves memory", *Frontiers for Young Minds*, vol. 9, 2021, doi:10.3389/frym.2021.605677.

[70] H. N. Locke, J. Brooks, L. J. Arendsen, N. K. Jacob, A. Casson, A. K. Jones, M. Sivan, "Acceptability and usability of smartphone-based brainwave entrainment technology used by individuals with chronic pain in a home setting", *British journal of pain*, vol. 14, no. 3, pp. 161–170, 2020, doi:10.1177/2049463720908798.

[71] N. C. Lecavalier, B. Boller, S. Belleville, "Use of immersive virtual reality to assess episodic memory: A validation study in older adults", *Neuropsychological Rehabilitation*, 2018, doi:10.1080/09602011.2018.1477684.

[72] J. P. Rosenfeld, E. Baehr, R. Baehr, I. H. Gotlib, C. Ranganath, "Preliminary evidence that daily changes in frontal alpha asymmetry correlate with changes in affect in therapy sessions", *International Journal of Psychophysiology*, vol. 23, no. 1-2, pp. 137–141, 1996.

[73] M. Klug, T. Berg, K. Gramann, "Optimizing EEG ICA decomposition with data cleaning in stationary and mobile experiments", *Scientific Reports*, vol. 14, no. 1, p. 14119, 2024, doi:10.1038/s41598-024-64919-3.

[74] R. N. Vigário, "Extraction of ocular artefacts from EEG using independent component analysis", *Electroencephalography and Clinical Neurophysiology*, vol. 103, no. 3, pp. 395–404, 1997, doi:10.1016/S0013-4694(97)00042-8.

[75] R. Vigário, J. Sarela, V. Jousmiki, M. Hamalainen, E. Oja, "Independent component approach to the analysis of EEG and MEG recordings", *IEEE Transactions on Biomedical Engineering*, vol. 47, no. 5, pp. 589–593, 2000, doi:10.1109/10.841330.

[76] C. A. Joyce, I. F. Gorodnitsky, M. Kutas, "Automatic removal of eye movement and blink artifacts from EEG data using blind component separation", *Psychophysiology*, vol. 41, no. 2, pp. 313–325, 2004, doi:10.1111/j.1469-8986.2003.00141.x.

[77] C. Mateo, J. A. Talavera, "Bridging the gap between the short-time fourier transform (STFT), wavelets, the constant-q transform and multi-resolution STFT", *Signal, Image and Video Processing*, vol. 14, no. 8, pp. 1535–1543, 2020, doi:10.1007/s11760-020-01701-8.

[78] J. T. Gregg, J. H. Moore, "Star outliers: a python package that separates univariate outliers from non-normal distributions", *BioData Mining*, vol. 16, no. 1, p. 25, 2023, doi:10.1186/s13040-023-00342-0.

[79] C. Ecker, S. Y. Bookheimer, D. G. Murphy, "Neuroimaging in autism spectrum disorder: brain structure and function across the lifespan", *The Lancet Neurology*, vol. 14, no. 11, pp. 1121–1134, 2015, doi:10.1016/S1474-4422(15)00050-2.

[80] J. J. Foxe, A. C. Snyder, "The role of alpha-band brain oscillations as a sensory suppression mechanism during selective attention", *Frontiers in Psychology*, vol. 2, p. 154, 2011, doi:10.3389/fpsyg.2011.00154.

[81] P. A. Antonov, R. Chakravarthi, S. K. Andersen, "Too little, too late, and in the wrong place: Alpha band activity does not reflect an active mechanism of selective attention", *NeuroImage*, vol. 219, p. 117006, 2020, doi:10.1016/j.neuroimage.2020.117006.

[82] B. K. Taylor, J. A. Eastman, M. R. Frenzel, C. M. Embury, Y. P. Wang, V. D. Calhoun, T. W. Wilson, "Neural oscillations underlying selective attention follow sexually divergent developmental trajectories during adolescence", *Developmental Cognitive Neuroscience*, vol. 49, p. 100961, 2021, doi:10.1016/j.dcn.2021.100961.

[83] J. H. Lee, M. A. Whittington, N. J. Kopell, "Top-down beta rhythms support selective attention via interlaminar interaction: a model", *PLoS Computational Biology*, vol. 9, no. 8, p. e1003164, 2013, doi:10.1371/journal.pcbi.1003164.

[84] A. R. Clarke, R. J. Barry, R. McCarthy, M. Selikowitz, C. A. Magee, S. J. Johnstone, R. J. Croft, "Quantitative EEG in low-iq children with attention-deficit/hyperactivity disorder", *Clinical Neurophysiology*, vol. 117, no. 8, pp. 1708–1714, 2006, doi:10.1016/j.clinph.2006.04.015.

[85] M. M. Lansbergen, M. Arns, M. van Dongen-Boomsma, D. Spronk, J. K. Buitelaar, "The increase in theta/beta ratio on resting-state EEG in boys with attention-deficit/hyperactivity disorder is mediated by slow alpha peak frequency", *Progress in Neuro-Psychopharmacology and Biological Psychiatry*, vol. 35, no. 1, pp. 47–52, 2011, doi:10.1016/j.pnpbp.2010.08.004.

[86] E. Shephard, C. Tye, K. L. Ashwood, B. Azadi, P. Asherson, P. F. Bolton, G. McLoughlin, "Resting-state neurophysiological activity patterns in young people with ASD, ADHD, and ASD+ ADHD", *Journal of Autism and Developmental Disorders*, vol. 48, pp. 110–122, 2018, doi:10.1007/s10803-017-3300-4.

Dr. Manasa Mandapati has done her bachelor's degree from JNTUH in 2007. She has done her master's degree from NMIMS in 2013. She has completed PGDM in patent law from NALSAR university in 2018. She completed her PhD degree in computer science engineering with specialisation in brain computer interface from DYPIU in 2025.

She has a mixed experience of industry, and academia. She started her career as a project engineer at Wipro technologies from 2008 to 2010. She contributed as a Women Scientist C with TIFAC where she gained exposure to intellectual property rights and innovation driven projects. She also worked as a trainer for IP awareness under NIPAM 2.0. Her area of research is in artificial intelligence, brain computer interface. She is currently working as assistant professor. Published four research papers in reputed journals like Applied Neuropsychology Child.

Prof. Prabhat Ranjan was educated at Netarhat School(near Ranchi), IIT Kharagpur, and Delhi University. He received his Ph.D. from the University of California, Berkeley where he carried out research on "Nuclear Fusion" at Lawrence Berkeley Laboratory from 1983-86.

Prof. Prabhat Ranjan is currently Chairman(Research, Innovation and International Relations) of D Y Patil Pratishthan covering multiple institutions in Pune and Kolhapur. He is also the founder Vice Chancellor of D Y Patil International University(DYPIU), Akurdi, Pune and served in this position from April 2018 to April 2025. From April 2013 to April 2018, he was heading India's Technology Think Tank, TIFAC in Delhi. TIFAC is an autonomous body under the Dept of Science and Technology, Govt of India. Earlier he was Professor at Dhirubhai Ambani Institute for Information and Communication Technology, Gandhinagar (DA-IICT) from 2002 to 2013.