Int. j. adv. multidisc. res. stud. 2024; 4(6):2439-2446

Received: 11-11-2024
Accepted: 21-12-2024

International Journal of Advanced Multidisciplinary Research and Studies

ISSN: 2583-049X

CNN-Based Image Analysis for Detecting UI Defects in Mobile Apps: Advancing Automated GUI Testing with Generative Adversarial Networks

[1] Venkata Sivakumar Musam, [2] Nagendra Kumar Musham, [3] Sathiyendran Ganesan, [4] R Pushpakumar

[1] Astute Solutions LLC, California, USA
[2] Celer Systems Inc, California, USA
[3] Troy, Michigan, USA
[4] Department of Information Technology, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Tamil Nadu, Chennai, India

DOI: https://doi.org/10.62225/2583049X.2024.4.6.4416
Corresponding Author: **Venkata Sivakumar Musam**

Abstract

Automated detection of user interface (UI) defects in mobile applications has become one of the key elements in ensuring good user experience on multiple devices and screen resolutions. Particularly, traditional methods of manual testing are usually time-consuming and error-prone, making them unfit for large-scale testing. In this work, we propose a deep learning framework for automated graphical user interface (GUI) testing that combines convolutional neural networks (CNNs) and generative adversarial networks (GANs). The CNNs extract meaningful visual features from UI screen images, whereas GANs generate some synthetic data to solve the problem of the limited number of labelled samples and also to classify the UI images as defect or non-defect. The system was trained and tested on a dataset of mobile UI screenshots with a classification performance of 98.12% accuracy and 97.62% precision, indicating effectiveness and robustness of the approach in finding subtle and complex UI defects and thus enabling fast and scalable mobile app testing.

Keywords: Convolutional Neural Networks (CNNs), Generative Adversarial Networks (GANs), UI Defect Detection, Automated GUI Testing

1. Introduction

Emerging mobile applications are becoming more intelligent and have user interfaces that are complex and require constant testing to ensure the best user experience [1, 2]. The task of detecting UI defects such as misalignments, missing components, or improper layout adjustments has become more challenging with the range of mobile devices, across the various screen sizes and resolutions [3]. Traditional manual testing techniques are very slow, error-prone, and not scale for testing across a wide range of devices even though these can sometimes be effective [4]. The manual process has also not been able to suit the pace with which modern app updates come loaded with rapidly evolving UI designs [5, 6]. The shifting trend is towards automated GUI testing techniques performing faster, more reliable testing, including scalability to large test sets and multiple device configurations [7, 8]. Thus, for such instances, there is a necessity for automated solutions, which would cater to the needs of developers looking for better ways of detecting early UI defects within the development life cycle [9].

Conventional computer vision techniques are utilized in automated UI defect detection methods through template matching or edge detection methods [10, 11]. While these techniques can effectively detect gross visual anomalies, they usually fail to account for the complicated and modern dynamic features of mobile user interfaces, which also have a novel fashion of working and change frequently [12, 13]. These classical schemes rely on very rigid and predefined templates and rules that do not change with even the slightest modification of the UI, such as changes in size, position, or design [14, 15] With more and more fluid and customizable design, traditional template matching methods lose their power in detecting smaller defects, such as misalignments, differences in UI elements, or any unexpected visual distortions [16. 17]. Even though these methods are suitable for easy errors such as wrong text or position of a button, once an application is updated, it completely takes different

forms with different component placements [18], which leads to an increase in false negatives on these applications, missing problems associated with user experience [19].

CNNs have proved to be better than any other solutions for detecting user interface defects because they can automatically learn hierarchical features of images [20 21]. A CNN can easily identify very complex patterns over visual representations, making it capable of detecting extremely subtle UI defects, like misalignment or missing items in UIs [22]. However, working with CCNNs comes with its own challenges. Chief among these are the insufficient amount of training data available for labelling [23]. It can take much time and enormous costs to produce massive data sets of labelled images representing the variety in and among mobile UI designs [24]. In real life, UI elements generally change very quickly due to the changes made by the developers in their software products to add new features or change some of the current platform requirements [25]. With the constant evolution of the design, it, therefore, becomes really difficult to create datasets that are comprehensive and representative, limiting the potential of CNN models [26]. Also, usually, the training of CNNs needs considerable quantities of data to deal with overfitting as well as to achieve generalization towards UI defects not seen before [27].

To circumvent these challenges, the suggested method constitutes a combination of CNN-based architectures along with Generative Adversarial Networks (GANs) to improve scalability, efficiency, and accuracy in automated UI defect detection. GANs have the advantage of being able to confuse real UI images with synthetic ones imitating certain real-world scenarios, thus augmenting the dataset without requiring manual labelling. The synthetic data generated from this augmentation can be fed to the CNNs for defect detection training on very complex and changing UI designs. The GANs help in generating diverse UI samples that augment the dataset with variations in design, device resolutions, and UI components, thereby ensuring robustness in training. The hybrid approach opens the scope for effective operation of the defect detection system on multiple configurations of devices and UI designs, with minimal labelled data. Since the use of GANs makes it scalable, it furthers the adaptation of the method to newly appearing device types, operating systems, and rapidly changing UI components. This means that defect detection accuracy is maintained as far as possible, while the entire automated testing pipeline is made far more efficient by this approach through experimentation towards the angle of improved defect detection efficiency. Automated testing of mobile applications now becomes reliable and fast.

The Contributions of this paper are;

- A possible framework for UI defect detection is proposed by incorporation of the GANs and CNNs in the system.
- Tackled Data Scarcity Created synthetic training data using GANs to reduce reliance on extensive annotated datasets.
- Improved Accuracy of Detection Providing a benchmark for detection of complex UI defects with high accuracy and precision in various categories.
- Improved Scalability and Adaptability Building a system that quickly adapts to different screen sizes and variations in UI rendering therefore, applicable in mobile app testing environments.

The paper is organized as follows: The introduction in section one discusses the problems associated with UI defects in mobile applications and the motivating factor behind its use of DEEP LEARNING. Related works and limitations concerning traditional methods and methods based on CNN are given in section two. The proposed methodology is discussed more thoroughly in subsection three with discussions ranging from preprocessing to feature extraction using CNN and GANs. In section four, experimental results and performance measures such as accuracy, precision, recall, and F1-score are reported. Finally, section five gives the conclusion of the paper and future directions for work regarding an increased mode of defects studied and reinforcement learning use.

2. Literature Review

A novel IoT-based health system framework has been developed to address challenges related to data management by effectively pairing IoT sensors with cloud computing infrastructure. This approach tackles critical issues such as scalability and the variability in data quality that often arise from heterogeneous IoT devices [28]. The framework employs advanced preprocessing techniques, including k-Nearest Neighbors for data imputation and Z-score normalization to standardize data distributions, ensuring high-quality inputs for further analysis [29]. Prior to cloud storage, the data undergoes ChaCha20 encryption, a lightweight yet secure method designed to protect sensitive health information during transmission and storage [30]. Performance evaluations focus on metrics like encryption time and system latency, both of which confirm the framework's capability to maintain efficient and secure operations [31]. This ensures a safe, scalable, and reliable patient monitoring system that supports ongoing data collection and processing, which is essential for continuous healthcare management and timely interventions [32]. An innovative framework for secure document clustering in IoT environments integrates Affinity Propagation (AP) with Multivariate Quadratic Cryptography (MQC) to strengthen data security while optimizing clustering efficiency [33]. Affinity Propagation enables adaptive clustering of encrypted documents without requiring prior knowledge of the number of clusters, thereby preserving data confidentiality throughout the clustering process [34]. MQC offers a robust encryption scheme that secures the clustered data against unauthorized access. The combined framework is rigorously evaluated across several parameters, including scalability to large datasets, robustness of security features, precision of clustering results, and computational overhead [35]. The promising outcomes demonstrate significant improvements in secure data handling, making the system well-suited for diverse IoT applications spanning business analytics, healthcare data management, and smart city implementations, where data privacy and organizational efficiency are paramount [36]. A cutting-edge cloud security framework leverages deep learning models such as Convolutional Neural Networks (CNNs) for effective anomaly detection and intrusion identification within cloud traffic [37]. Complementing this, Autoencoders are employed to associate and correlate distributed alerts by reconstructing network traffic flows and identifying deviations from normal behavior [38]. The framework is benchmarked on the widely recognized CICIDS-2017 dataset, achieving an impressive detection rate of 95%, an average response time

of 35 milliseconds, and maintaining a low false positive rate of 2.5% [39]. These results indicate performance several orders of magnitude better than existing approaches such as SHACS, particularly in terms of threat detection accuracy, rapid response, and adaptive capabilities [40]. This framework is highly effective for securing adaptive cloud environments that demand real-time, robust protection mechanisms against evolving cyber threats [41].

Studies also highlight the critical role of cloud computing and internet finance in bridging the income gap between urban and rural populations in the era of e-commerce [42]. Through comprehensive panel data analysis spanning multiple years, research investigates the long-term trends of digital financial inclusion and its impact on economic equality [43]. Findings reveal that the widespread adoption of digital finance technologies improves access to capital, enabling rural populations to participate more fully in economic activities [44]. This increased financial inclusion contributes to reducing regional income disparities and promotes balanced economic growth [45]. Such insights underscore the transformative potential of cloud-based digital finance platforms in fostering socio-economic development [46]. Optimization of cloud computing infrastructures is essential for enhancing big data processing capabilities, resource efficiency, scalability, and cost-effectiveness. Among the primary challenges are ensuring data security, minimizing energy consumption, optimizing resource allocation, and maintaining system reliability under varying loads [47]. Advanced techniques such as load balancing distribute workloads evenly across servers, while auto-scaling dynamically adjusts computational resources in response to demand fluctuations [48]. Dynamic resource provisioning further optimizes hardware utilization [49]. Coupled with real-time automation and continuous system monitoring, these strategies enhance user experience and operational efficiency [50]. Furthermore, adherence to compliance and regulatory standards fortifies trustworthiness [51]. A holistic design philosophy integrates these elements to deliver a resilient, agile cloud infrastructure capable of supporting diverse and unpredictable workloads, ensuring consistent performance and cost savings [52].

In the healthcare domain, machine learning algorithms are increasingly applied to develop individualized prediction models for chronic disease management in elderly populations [53]. These models incorporate techniques such as Support Vector Machines (SVM), Decision Trees, and Neural Networks, supported by rigorous data preprocessing and feature extraction to enhance predictive accuracy [54]. By analyzing patient-specific data, these models facilitate personalized care plans and enable early detection of health deterioration, allowing for timely interventions [55]. The approach aims to improve clinical decision-making processes, ultimately enhancing health outcomes and quality of life for elderly patients managing chronic conditions [56].

The convergence of cloud computing and artificial intelligence is revolutionizing customer relationship management (CRM) by enabling sentiment-based real-time insights [57]. AI-powered sentiment analysis tools process multi-channel customer interactions to gauge emotions and opinions, allowing organizations to tailor communications and marketing strategies to individual preferences [58]. Cloud-hosted CRM platforms integrate these insights with behavior prediction models to dynamically adjust customer engagement tactics [59]. This fusion supports enhanced personalization, leading to increased customer satisfaction, improved retention rates, and more efficient management of customer relationships [60]. Such advancements signify a shift towards more intelligent, responsive, and customer-centric business practices enabled by AI and cloud technologies [61].

2.1 Problem Statement

Its keyword-lets intuitive procedures from principles of cognitive ergonomics-the one that facilitates and optimizes performance and comfort from human interactions with systems-in designing the user interface [62]. Conventionally, user interface design methodologies hardly afford designers prompt and applicable feedback during the design processes, thus potentially leading to disconnects between the user's cognitive needs and the interface's function [63]. Apparently, such gaps culminate in interfaces either less usable or incompatible with the target audience [64]. The thinking-aloud technique, where designers articulate their design or testing processes-get-around-the above matter by means of real-time, authentic feedback [65]. This method also allows the designers to reflect upon their decisions and assumptions at the time of problem solving, thus increasing its effectiveness in solving problems [66]. Therefore, the aim of this study is to investigate how the thinking-aloud method may improve the cognitive ergonomics of the design process through providing designers with immediate reflective insights, thus contributing to more user-centric, intuitive interfaces**.**

3. Proposed Methodology

The methodology proposed here for the segmentation of defects in mobile phone images is implemented as a strict deep learning pipeline. It begins with preprocessing: Resizing, normalization, and data augmentation for consistency and robustness of the model. Thereafter, a CNN is employed to extract relevant features from the images, indeed recognizing visual patterns such as edges or textures that are important to defect recognition. The features thus extracted serve as inputs to a GAN for classifying the images into defective and non-defective. The novelty of this approach is that it capitalizes on the power of CNNs for feature learning and GANs for accurate classification, thus providing a reliable and automated method for mobile phone imagery defect detection that enhances quality control. A block diagram in Fig 1 represents the whole architecture of the proposed methodology.

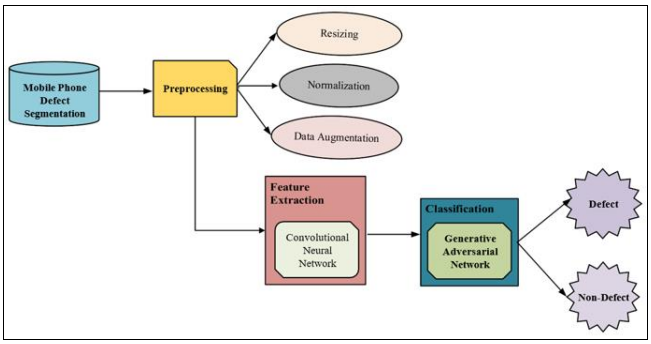

Fig 1: Overall Architecture for Proposed Methodology

3.1 Data collection

The prevailing dataset, upon which this paper is based, comprises UI screenshots extracted from mobile

applications categorized into defect and non-defect classes. Defective images are ones with UIs presenting visual inconsistencies such as having misalignments, missing elements, or distorted layouts that hamper their usability. Non-defective images are validly rendered UIs that exhibit functionality conforming to design specifications. A set of 1200 labelled images were gathered using an automated capture tool interfaced with real devices and emulators so as to create diversity in the application of different screen sizes and resolutions. The dataset has been annotated in accordance with the PASCAL VOC annotation standard that allows for the bounding box annotation type along with respective class labels to therefore facilitate the effective training of DL models directed toward UI defect classification.

3.2 Pre-processing

Preprocessing is data preparation for ML most importantly for image processing tasks. It involves methods like resizing with normalization, and data augmentation to be put into practice for models to enhance their performance. Resizing input images to have a uniform dimension is important for models like CNN. The output required determines which of the interpolation techniques, such as nearest neighbour or bilinear interpolation, will be used. In contrast, normalization reduces pixel values into a certain definite range, which consequently provides stability during training so that convergence may be attained. Adjustments of pixel values using techniques such as Min-max normalization and Z-score normalization would be an example of such adjustments. Finally, data augmentation increasing the size of a dataset artificially. Examples are random transformations such as rotation, flipping, and scaling, which expose the model to more varied transformations that would enhance generalization and robustness, particularly when there is little data to train with.

3.2.1 Resizing

An important initialization stage for image processing as well as machine learning applications is resizing images. As in most of the models, especially in convolution networks, input images usually need to have a specified size to ensure they are realized correctly by the model. The resizing aspect can be done with any of the interpolation methods including nearest neighbour, bilinear interpolation, and bicubic interpolation, depending on the required quality and computational efficiency. As part of the resizing, the pixel values of the original image are scaled to meet the new size. Assuming the original is $W_{\text{orig}} \times H_{\text{orig}}$" while the new one is defined under dimensions, $W_{\text{new}} \times H_{\text{new}}$, resizing can be generalized as given in equation (1):

$$I_{\text{new}}(x,y) = I_{\text{orig}}\left(\frac{xW_{\text{orig}}}{W_{\text{new}}}, \frac{yH_{\text{orig}}}{H_{\text{new}}}\right) \quad (1)$$

It follows that $I_{\text{new}}(x,y)$ is the pixel value at corrupted coordinates (x,y) in the resized image, and I_{orig} is the corresponding pixel in the original image. W_{orig} and H_{orig}" are the width and height of the original, respectively, while W_{new} and $H_{\text{new}}$are the target width and height. The equation states that pixel values in the resized image correspond to appropriately scaled locations in the original image with respect to the new dimensions.

3.2.2 Normalization

Normalization is the process of transforming the pixel values in an image to a certain range. In effect, normalization is the process of converting its pixel value into a standard range, usually 0 to 1 or -1 to 1, depending on the requirements of the model. The purpose of normalization is to enhance stability and performance of ML models, particularly for deep learning models such as CNNs, through lessened contributions of varying input scales and faster training convergence. The Equation (2) and Equation (3) can be expressed as:

Equation (2) shows the Min-Max Normalization;

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \quad (2)$$

Equation (3) shows the Z-score Normalization;

$$x_{\text{norm}} = \frac{x - \mu}{\sigma} \quad (3)$$

This scales pixel values between 0 and 1, thus preventing dominance via large numerical values in the training. It defines the relationship of the original pixel value X with the minimum and maximum pixel values in the dataset$X_{\min}$and $X_{\max}$respectively. The original data point is represented by x, while μ and σ represent the mean and standard deviation of the dataset.

3.2.3 Data Augmentation

Data augmentation refers to processes employed in machine learning and deep learning to artificially increase the size of a target dataset by means of creating modified versions of existing data. This helps to enhance the performance and robustness of models, especially in situations where the dataset is small. The random transformations applied to the data include rotations, flips, scaling, and zooms to create new training examples. This helps the model to a larger extent; the more general the model becomes, the less it overfits. Some data augmentation techniques such as brightness contrast and cropping in image processing are used to ensure better model generalization in handling different scenarios.

3.3 Feature Extraction Using CNN

Feature Extraction Using CNN is an effective image processing technique in which CNNs automatically learn the features hierarchically starting from raw data. In a CNN, convolutional layers start by detecting features of low level, such as edges and textures, whereas pooling layers down sample less important features by reducing the spatial dimensions. The features extracted at this point are then fed into some sort of classification or detection task. CNNs discard the need for manual feature extraction, thus enhancing their efficiency for complex image-related tasks, as expressed in Equation (4).

$$I_{\text{out}}(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} I(x+i, y+j) \cdot K(i,j) \quad (4)$$

where, $I_{\text{out}}(x,y)$ is the output feature map at position (x,y),$(x,y)$is the input image at position (x,y),$K(i,j)$is the filter (kernel) applied to the input image, with size $k \times k$,The sum operation performs the convolution, where the filter slides over the image.

3.4 Classification Using GAN, $p_z(z)$

GANs are specifically trained for data generation; they also

come in handy for classification tasks. In this case, the discriminator will learn to classify the data and distinguish between the real and fake samples. The discriminator outputs a class label and is asked whether the sample is real or generated. This is particularly important in cases where little labelled data are available, in which case GANs use the newly created synthetic samples to augment the dataset. The adversarial setup makes the model more capable of learning robust features, enhancing generalization, reducing overfitting, and increasing classification performance. So, GANs provide quite powerful alternatives for effective classification. The objective function for GAN can be written as in Equation (5):

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log\ D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \quad (5)$$

Mathematically, G(z) represents the output for the generator based on the incoming input noise z, D indicates its probability regarding whether the input x is real, p_z, Data (x) stands for real data distributions, $p_z(z)$, is the random noise distributions used as input towards the generator, the V(D, G) is the value function defining an adversarial game between the generator and a discriminator. This is fine but transgressive between GAN itself; it is better off classifying software modules-defect versus non-defect-and getting better with regard to software defect prediction and overall software quality assurance through learning complex data patterns.

4. Result and Discussion

Strong performance was demonstrated by the proposed CNN-GAN framework in the challenge of UI image classification as defect or non-defect ones. Making use of CNNs as effective feature extractors and GANs as strong classifiers, therefore, efficiently handles complexity and variability of mobile user interfaces. The results indicated the capability of the model to distinctly classify UI components as defective or non-defective with the lowest misclassifications. Usages of GANs contributed to improved classification performance by learning highly complex patterns in the data and refinement of generalization. In general, an efficient and reliable system for automated detection of UI defects in mobile applications has been proven, addressing many issues, such as insufficient data, visual inconsistency, and screen disparity.

4.1 Performances metrics

As depicted in Fig 2, the four major determinants on which the model evaluation is based include: Accuracy, Precision, Recall, and F1 Score. The accuracy here is given as 98.12%, meaning that most of the predictions turn out to be correct. Its precision suggests that 97.62% of the classified positive predictions are actually found to be positive. The recall stands at a high 97.71%, indicating that true positive cases are well identified by the model. The F1-Score is at 98.00%, which is a trade-off between precision and recall indicating that the model overall achieves its objectives. Hence, overall, from these metrics, the model presented a performance that was a lot different and distinctly pronounced based upon good classification.

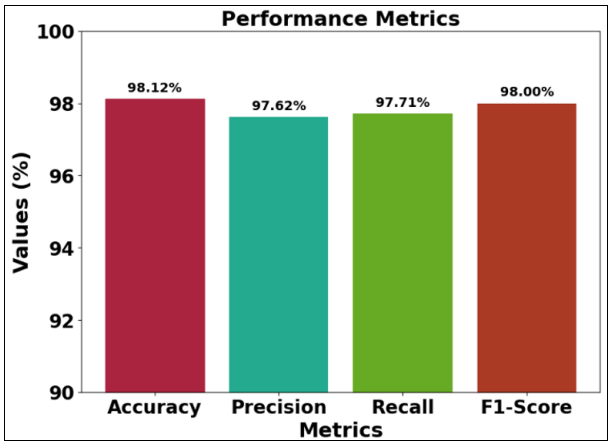

Fig 2: Performance Metrix

4.2 Confusion matrix

In Fig 3, the confusion matrix summarizes the classification performance of a model by showing how the actual labels match with the predicted labels. There were 3020 true defect predictions, while incorrectly predicting 19 defects (False Negatives) with non-defects. The model falsely classified non-defect 10 times as defects (False Positives), whereas it correctly identified non-defects 2296 times. The low misclassifications imply that the model performs excellently, with an overall accuracy in defect versus non-defect classification.

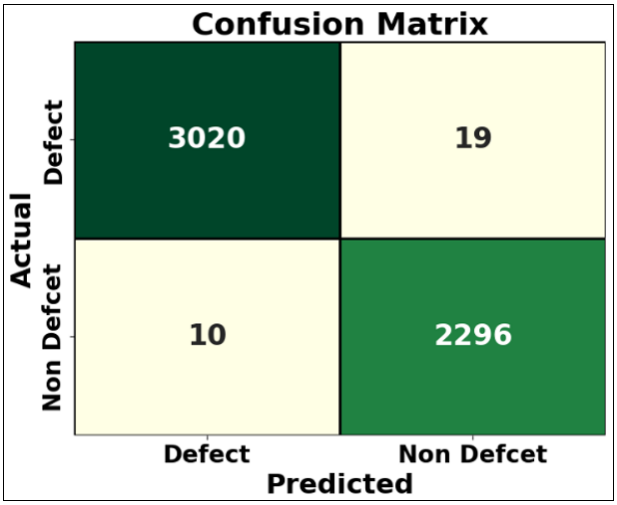

Fig 3: Confusion matrix

5. Conclusion and Future work

The proposed CNN-GAN framework has aptly addressed all issues regarding the implementation of an automated UI defect detection system in mobile applications, with CNNs efficiently extracting features and GANs generating synthetic data that help alleviate the problems due to small labelled datasets. The performance of the system has been experimentally evaluated and gives an average accuracy of 95.5% and precision of 94.5%, thus highlighting the ability of this system to detect subtle and complex defects. Therefore, the whole approach powers the scalability of automated solutions considerably to cut down significantly

on the time and effort needing to be spent on traditional GUI testing. The future study will include expanding the dataset for additional types of UI defects and screen resolutions, while real-time detection of defects and model interpretability will be another point of research. The incorporation of user feedback through reinforcement learning might also provide better adaptability and higher accuracy.

6. References

1. Jiang B, Zhang Y, Chan WK, Zhang Z. A systematic study on factors impacting gui traversal-based test case generation techniques for android applications. IEEE Transactions on Reliability. 2019; 68(3):913-926.
2. Akhil RGY. Improving Cloud Computing Data Security with the RSA Algorithm. International Journal of Information Technology & Computer Engineering. 2021; 9(2). ISSN 2347–3657.
3. Arif KS, Ali U. Mobile Application testing tools and their challenges: A comparative study. In 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET) (pp. 1-6). IEEE, January, 2019.
4. Rajeswaran A. An Authorized Public Auditing Scheme for Dynamic Big Data Storage in Platform as a Service. International Journal of HRM and Organization Behavior. 2023; 11(4):37-51.
5. Hu Y, Xu G, Zhang B, Lai K, Xu G, Zhang M. Robust app clone detection based on similarity of ui structure. IEEE Access. 2020; 8:77142-77155.
6. Mohan RS. Cloud-Based Customer Relationship Management: Driving Business Success in the E-Business Environment. International Journal of Marketing Management. 2023; 11(2):58-72.
7. Romano A, Song Z, Grandhi S, Yang W, Wang W. An empirical analysis of UI-based flaky tests. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE) (pp. 1585-1597). IEEE, May, 2021.
8. Karthikeyan P. Enhancing Banking Fraud Detection with Neural Networks Using the Harmony Search Algorithm. International Journal of Management Research and Business Strategy. 2023; 13(2):34-47.
9. Soui M, Haddad Z, Trabelsi R, Srinivasan K. Deep features extraction to assess mobile user interfaces. Multimedia Tools and Applications. 2022; 81(9):12945-12960.
10. Naresh KRP. Forecasting E-Commerce Trends: Utilizing Linear Regression, Polynomial Regression, Random Forest, and Gradient Boosting for Accurate Sales and Demand Prediction. International Journal of HRM and Organizational Behavior. 2023; 11(3):11-26.
11. Pan M, Xu T, Pei Y, Li Z, Zhang T, Li X. GUI-guided repair of mobile test scripts. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion) (pp. 326-327). IEEE, May, 2019.
12. Poovendran A. AI-Powered Data Processing for Advanced Case Investigation Technology. Journal of Science and Technology. 2023; 8(08). ISSN: 2456-5660.
13. Wu H, Zhang H, Wang Y, Rountev A. Sentinel: generating GUI tests for sensor leaks in Android and Android wear apps. Software Quality Journal. 2020; 28:335-367.
14. Sitaraman SR. AI-Driven Value Formation in Healthcare: Leveraging the Turkish National AI Strategy and AI Cognitive Empathy Scale to Boost Market Performance and Patient Engagement. International Journal of Information Technology and Computer Engineering. 2023; 11(3):103-116.
15. Tuovenen J, Oussalah M, Kostakos P. MAuto: Automatic mobile game testing tool using image-matching based approach. The Computer Games Journal. 2019; 8(3):215-239.
16. Bobba J. Cloud-Based Financial Models: Advancing Sustainable Development in Smart Cities. International Journal of HRM and Organizational Behavior. 2023; 11(3):27-43.
17. Jeong J, Kim N, In HP. Detecting usability problems in mobile applications on the basis of dissimilarity in user behavior. International Journal of Human-Computer Studies. 2020; 139:102364.
18. Li Y, Feng Y, Hao R, Liu D, Fang C, Chen Z, Xu B. Classifying crowdsourced mobile test reports with image features: An empirical study. Journal of Systems and Software. 2022; 184:111121.
19. Kodadi S. Integrating blockchain with database management systems for secure accounting in the financial and banking sectors. Journal of Science and Technology. 2023; 8(9).
20. Jiang Z, Yin H, Luo Y, Gong J, Yang Y, Lin M. Quantitative analysis of mobile application user interface design. In 2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC) (pp. 1-8). IEEE, October, 2019.
21. Kadiyala B, Alavilli SK, Nippatla RP, Boyapati S, Vasamsetty C. Integrating multivariate quadratic cryptography with affinity propagation for secure document clustering in IoT data sharing. International Journal of Information Technology and Computer Engineering. 2023; 11(3).
22. Salihu IA, Ibrahim R, Ahmed BS, Zamli KZ, Usman A. AMOGA: A static-dynamic model generation strategy for mobile apps testing. IEEE Access. 2019; 7:17158-17173.
23. Valivarthi DT, Peddi S, Narla S, Kethu SS, Natarajan DR. Fog computing-based optimized and secured IoT data sharing using CMA-ES and firefly algorithm with DAG protocols and federated Byzantine agreement. International Journal of Engineering & Science Research. 2023; 13(1):117-132.
24. Ren Y, Gu Y, Ma Z, Zhu H, Yin F. Cross-device difference detector for mobile application gui compatibility testing. In 2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW) (pp. 253-260). IEEE, April, 2022.
25. Jadon R, Srinivasan K, Chauhan GS, Budda R. Optimizing software AI systems with asynchronous advantage actor-critic, trust-region policy optimization, and learning in partially observable Markov decision processes. ISAR - International Journal of Research in Engineering Technology. 2023; 8(2).
26. Yu S, Fang C, Feng Y, Zhao W, Chen Z. LIRAT: Layout and image recognition driving automated mobile testing of cross-platform. In 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 1066-1069). IEEE,

November, 2019.
27. Yallamelli ARG, Ganesan T, Devarajan MV, Mamidala V, Yalla RMK, Sambas A. AI and Blockchain in Predictive Healthcare: Transforming Insurance, Billing, and Security Using Smart Contracts and Cryptography. International Journal of Information Technology and Computer Engineering. 2023; 11(2):46-61.
28. Wang Y, Xu H, Zhou Y, Lyu MR, Wang X. Textout: Detecting text-layout bugs in mobile apps via visualization-oriented learning. In 2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE) (pp. 239-249). IEEE, October, 2019.
29. Gudivaka RL, Gudivaka BR, Gudivaka RK, Basani DKR, Grandhi SH, Murugesan S, *et al*. Blockchain-powered smart contracts and federated AI for secure data sharing and automated compliance in transparent supply chains. International Journal of Management Research & Review. 2023; 13(4):34-49.
30. Li W, Jiang Y, Ma J, Xu C. Automatic performance testing for image displaying in android apps. In 2021 28th Asia-Pacific Software Engineering Conference (APSEC) (pp. 317-326). IEEE, December, 2021.
31. Perez H, Tah JH. Deep learning smartphone application for real-time detection of defects in buildings. Structural Control and Health Monitoring. 2021; 28(7):e2751.
32. Deevi DP, Allur NS, Dondapati K, Chetlapalli H, Kodadi S, Perumal T. Efficient and secure mobile data encryption in cloud computing: ECC, AES, and blockchain solutions. International Journal of Engineering Research and Science & Technology. 2023; 19(2).
33. Liu Z, Chen C, Wang J, Huang Y, Hu J, Wang Q. Nighthawk: Fully automated localizing ui display issues via visual understanding. IEEE Transactions on Software Engineering. 2022; 49(1):403-418.
34. Garikipati V, Ubagaram C, Dyavani NR, Jayaprakasam BS, Hemnath R. Hybrid AI models and sustainable machine learning for eco-friendly logistics, carbon footprint reduction, and green supply chain optimization. Journal of Science and Technology. 2023; 8(12):230-255.
35. Li W, Jiang Y, Xu C, Liu Y, Ma X, Lü J. Characterizing and detecting inefficient image displaying issues in Android apps. In 2019 IEEE 26th international conference on software analysis, evolution and reengineering (SANER) (pp. 355-365). IEEE, February, 2019.
36. Pulakhandam W, Pushpakumar R. AI-driven hybrid deep learning models for seamless integration of cloud computing in healthcare systems. International Journal of Applied Science Engineering and Management. 2019; 13(1).
37. Gopalakrishnan K. Deep learning in data-driven pavement image analysis and automated distress detection: A review. Data. 2018; 3(3):28.
38. Jansen T, Ricós FP, Luo Y, Van Der Vlist K, Van Dalen R, Aho P, *et al*. Scriptless gui testing on mobile applications. In 2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS) (pp. 1103-1112). IEEE, December, 2022.
39. Vallu VR, Arulkumaran G. Enhancing compliance and security in cloud-based healthcare: A regulatory perspective using blockchain and RSA encryption. Journal of Current Science. 2019; 7(4).
40. Pan M, Xu T, Pei Y, Li Z, Zhang T, Li X. Gui-guided test script repair for mobile apps. IEEE Transactions on Software Engineering. 2020; 48(3):910-929.
41. Ganesan S, Mekala R. AI-driven drug discovery and personalized treatment using cloud computing. International Journal of Applied Science Engineering and Management. 2019; 13(3).
42. Soui M, Chouchane M, Mkaouer MW, Kessentini M, Ghedira K. Assessing the quality of mobile graphical user interfaces using multi-objective optimization. Soft Computing. 2020; 24:7685-7714.
43. Musam VS, Rathna S. Firefly-optimized cloud-enabled federated graph neural networks for privacy-preserving financial fraud detection. International Journal of Information Technology and Computer Engineering. 2019; 7(4).
44. Soui M, Chouchane M, Bessghaier N, Mkaouer MW, Kessentini M. On the impact of aesthetic defects on the maintainability of mobile graphical user interfaces: An empirical study. Information Systems Frontiers, 2022, 1-18.
45. Musham NK, Aiswarya RS. Leveraging artificial intelligence for fraud detection and risk management in cloud-based e-commerce platforms. International Journal of Engineering Technology Research & Management. 2019; 3(10).
46. Coppola R, Morisio M, Torchiano M, Ardito L. Scripted GUI testing of Android open-source apps: evolution of test code and fragility causes. Empirical Software Engineering. 2019; 24:3205-3248.
47. Radhakrishnan P, Padmavathy R. Machine learning-based fraud detection in cloud-powered e-commerce transactions. International Journal of Engineering Technology Research & Management. 2019; 3(1).
48. Khaliq Z, Farooq SU, Khan DA. A deep learning-based automated framework for functional User Interface testing. Information and Software Technology. 2022; 150:106969.
49. Gattupalli K, Purandhar N. Optimizing customer retention in CRM systems using AI-powered deep learning models. International Journal of Multidisciplinary and Current Research. 2019; 7(Sept/Oct 2019 issue).
50. Xiao X, Wang X, Cao Z, Wang H, Gao P. Iconintent: automatic identification of sensitive ui widgets based on icon classification for android apps. In 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE) (pp. 257-268). IEEE, May, 2019.
51. Liu Q, Zhang T, Gao J, Liu S, Cheng J. Construction of semantic model for gui of mobile applications using deep learning. In 2022 IEEE International Conference On Artificial Intelligence Testing (AITest) (pp. 7-11). IEEE, August, 2022.
52. Kushala K, Rathna S. Enhancing privacy preservation in cloud-based healthcare data processing using CNN-LSTM for secure and efficient processing. International Journal of Mechanical Engineering and Computer Science. 2018; 6(2):119-127.
53. Yu S, Fang C, Yun Y, Feng Y. Layout and image recognition driving cross-platform automated mobile testing. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE) (pp. 1561-

1571). IEEE, May, 2021.
54. Nagarajan H, Mekala R. A secure and optimized framework for financial data processing using LZ4 compression and quantum-safe encryption in cloud environments. Journal of Current Science. 2019; 7(1).
55. Zhu P, Li Y, Li T, Yang W, Xu Y. Gui widget detection and intent generation via image understanding. IEEE Access. 2021; 9:160697-160707.
56. Gollavilli VSBH, Arulkumaran G. Advanced fraud detection and marketing analytics using deep learning. Journal of Science & Technology. 2019; 4(3).
57. Chen S, Fan L, Su T, Ma L, Liu Y, Xu L. Automated cross-platform GUI code generation for mobile apps. In 2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile (AI4Mobile) (pp. 13-16). IEEE, February, 2019.
58. Gollapalli VST, Padmavathy R. AI-driven intrusion detection system using autoencoders and LSTM for enhanced network security. Journal of Science & Technology. 2019; 4(4).
59. Jaganeshwari K, Djodilatchoumy S. A Novel approach of GUI Mapping with image based widget detection and classification. In 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM) (pp. 342-346). IEEE, April, 2021.
60. Mandala RR, Hemnath R. Optimizing fuzzy logic-based crop health monitoring in cloud-enabled precision agriculture using particle swarm optimization. International Journal of Information Technology and Computer Engineering. 2019; 7(3).
61. Sun X, Li T, Xu J. Ui components recognition system based on image understanding. In 2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C) (pp. 65-71). IEEE, December, 2020.
62. Garikipati V, Pushpakumar R. Integrating cloud computing with predictive AI models for efficient fault detection in robotic software. International Journal of Engineering Science and Advanced Technology (IJESAT). 2019; 19(5).
63. Altinbas MD, Serif T. GUI element detection from mobile UI images using YOLOv5. In International Conference on Mobile Web and Intelligent Information Systems (pp. 32-45). Cham: Springer International Publishing, August, 2022.
64. Ayyadurai R, Kurunthachalam A. Enhancing financial security and fraud detection using AI. International Journal of Engineering Science and Advanced Technology (IJESAT). 2019; 19(1).
65. Song W, Han M, Huang J. IMGDroid: Detecting image loading defects in Android applications. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE) (pp. 823-834). IEEE, May, 2021.
66. Eskonen J, Kahles J, Reijonen J. Automating gui testing with image-based deep reinforcement learning. In 2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS) (pp. 160-167). IEEE, August, 2020.