International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Available at http://inpressco.com/category/ijcet

Research Article

Enhancing GUI Testing: Exploring Convolutional Neural Networks with Self-Learning Mechanisms for Automated UI Validation in Mobile Applications

[1]*Venkata Sivakumar Musam and [2]G. Arulkumaran

[1]Nisum chile, Santiago, Chile
[2]Bule Hora University: Bule Hora, Oromia, ET,

Received 01 March 2021, Accepted 17 April 2021, Available online 19 April 2021, **Vol.11, No.2 (March/April 2021)**

Abstract

The suggested methodology provides effective UI layout categorization with systematic workflow starting from an input source gathering UI images. Data preprocessing improves data quality through normalization and imputation of missing values. Feature extraction is accomplished using deep learning-based detection of significant UI elements followed by the prediction phase classifying layouts as defective or non-defective. The automated method enhances UI validation with a minimum of human interaction and a maximum of software development. The performance of the model is confirmed by critical metrics and has high accuracy (95.65%), precision (96.80%), recall (94.50%), and an F1 score of 95.90%, showing it is good for detecting defects.

Keyword: *Fully connected neural network (FCN), Convolutional Neural Network (CNN), UI validation, Defect detection, and Automated UI testing.*

1. Introduction

Graphical User Interface (GUI) testing plays a crucial role in ensuring a seamless and transparent user experience for mobile applications [1]. It helps maintain usability, reliability, and consistency across different user interactions [2]. Traditional rule-based testing methods and manual verification processes are often slow and inefficient [3]. These human-dependent processes cannot keep up with the demands of modern mobile UI development cycles [4]. GUI testing in mobile applications is further complicated by a variety of factors including screen size variations [5], diverse display resolutions [6], and frequent application updates [7]. These changes in the interface can disrupt previously validated functionalities [8]. Another major challenge is the dynamically changing nature of UI elements within the app [9]. This makes it difficult to reuse test cases across multiple versions of the app [10]. Legacy automation frameworks such as Selenium and Appium face difficulty in adapting to these dynamic changes [11]. They often fail to detect subtle interface variations like missing buttons or misplaced elements [12]. Traditional automation tools also require frequent maintenance and updates, adding to the development overhead [13].

Moreover, they are not well-suited for detecting unbound or partially visible UI elements [14]. As a result, developers may miss critical UI bugs, leading to poor user satisfaction and degraded app performance [15]. Manual GUI testing remains widely used but is time-consuming and prone to human error [16]. Manual test scripts require constant updating and are often not scalable for large applications [17]. Tools like Appium, while popular, still struggle with flexibility and test case reusability [18]. Image-based verification methods lack robustness when faced with animations and responsive design components [19]. Efforts to integrate deep learning in UI detection have shown promise, particularly using Convolutional Neural Networks (CNNs) [20]. However, CNN-based methods demand large labeled datasets for training [21]. These models also require periodic retraining, which adds computational and time overhead [22]. The variability of mobile UIs across platforms like Android and iOS increases the need for model generalization [23]. Additionally, UI elements can differ significantly between devices, requiring high adaptability in testing tools [24]. Given the rapid growth of mobile apps, automated testing must evolve to be more scalable [25]. The diversity in hardware, screen resolutions, and operating systems further complicates automation strategies [26]. Mobile applications also face challenges due to their frequent updates and newly added features [27]. Each update demands revalidation of the

*Corresponding author's ORCID ID: 0000-0000-0000-0000
DOI: https://doi.org/10.14741/ijcet/v.11.2.15

entire interface to maintain consistency [28]. Without effective GUI testing, inconsistencies can lead to broken interfaces and negative user feedback [29]. Conventional methods often fall short in validating modern, visually complex mobile UIs [30]. Apps now include custom components, dynamic content, and interactive animations that standard test tools struggle to handle [31]. Manual methods cannot scale efficiently under these requirements [32]. These limitations increase release times and elevate costs for mobile app development teams [33]. Developers require testing frameworks that can evolve along with the application [34]. This need drives the interest in adaptive and self-improving GUI testing solutions [35]. In this context, our work proposes a CNN-based UI validation framework with self-learning capabilities [36]. This framework leverages convolutional models to understand and classify UI elements more effectively [37]. It employs reinforcement learning to adapt based on testing feedback over time [38]. Active learning components help reduce the need for large volumes of labeled training data [39]. Together, these techniques aim to minimize human involvement while improving GUI testing accuracy and scalability [40]. The Contributions of this paper are,

- Transformer-Based Model employs self-attention mechanisms to understand short-term and long-term dependencies in UI layout data to detect structural inconsistencies and design defects accurately [17].
- Masked and Combined Self-Attention (MCSA) uses attention weights for important UI components, enhancing defect detection accuracy by paying attention to important layout characteristics [18].
- Mean Imputation and Min-Max Normalization handle missing values and normalize numerical UI parameters to a fixed range, making the data consistent and improving the overall model performance [19].

Despite the potential of automation, existing GUI testing tools face limitations such as low adaptability to UI changes, poor recognition accuracy in diverse conditions, and the need for extensive manual scripting or labelling [20]. Traditional image-based testing approaches often fail when confronted with slight variations in UI layouts or colors [21]. Additionally, many current methods lack self-learning capabilities to improve their detection and validation accuracy over time [22]. These challenges lead to false positives and negatives, undermining test reliability and developer confidence [23]. Furthermore, integrating testing frameworks seamlessly into continuous integration pipelines remains a complex task [24].

To address these challenges, integrating Convolutional Neural Networks with self-learning mechanisms offers a robust path forward [25]. CNNs can automatically learn and extract hierarchical visual features from GUIs, enabling precise recognition of UI elements under varying conditions [26]. Incorporating self-learning allows the system to adapt dynamically by continuously improving from new data and feedback, reducing the need for manual intervention [27]. This approach enhances accuracy, scalability, and resilience to UI modifications [28]. Combining these techniques with automated testing pipelines can accelerate release cycles, reduce testing overhead, and ensure higher UI quality in mobile applications, ultimately enhancing user experience and developer productivity [29].

The organization of this paper is as follows: Section 2 discusses existing UI validation approaches and their limitations [30]. Section 3 introduces the proposed transformer-based UI validation model and its process [31]. Section 4 tests the performance of the model using main metrics and a confusion matrix [32]. Lastly, Section 5 outlines the findings, contributions, and future work directions [33].

2. Literature Review

GUI testing plays an essential role in ensuring application quality, especially in the dynamic and fragmented ecosystem of mobile platforms [41]. With the surge of mobile devices featuring different screen sizes, OS versions, and interaction patterns, the demand for robust and adaptable GUI testing frameworks has intensified [42]. Manual GUI testing, although widely practiced, is inherently time-consuming and error-prone, making it unsuitable for continuous integration and agile development environments [43]. Tools like Appium and Selenium have been extensively used for GUI test automation, but they lack adaptability to frequent UI changes and require considerable manual scripting [44]. One of the key limitations in traditional GUI automation lies in the brittleness of UI element locators, which often break when UI layouts are altered [45]. Image-based techniques for GUI testing, including pixel comparison and template matching, are unable to handle dynamic content, animations, or responsive elements [46]. These static techniques struggle to maintain accuracy in the face of evolving interface designs [47]. As mobile applications grow more complex, incorporating adaptive layouts, user personalization, and animations, traditional methods fall short of providing consistent validation [48]. Recent advancements in deep learning have opened new avenues for UI element detection and classification. Convolutional Neural Networks (CNNs), in particular, have demonstrated strong performance in visual recognition tasks, making them suitable for GUI testing [49]. Several studies have employed CNNs to classify GUI components such as buttons, sliders, text fields, and icons with high accuracy [50]. However, the reliance on large-scale labeled datasets for training CNNs remains a bottleneck [51]. Manual annotation of GUI elements across app versions and platforms is resource-intensive and not scalable [52]. To overcome these challenges, researchers have begun integrating active learning techniques to reduce labeling efforts while maintaining high classification accuracy [53]. Active learning selectively queries the most

informative samples for labeling, optimizing the training process [54]. Reinforcement learning has also been explored in the domain of software testing, where models learn from interaction with the application environment to improve test strategies over time [55]. These self-learning mechanisms aim to minimize human intervention and allow the testing framework to adapt autonomously [56]. Transfer learning has shown promise in reducing the data and computational requirements for training CNNs in GUI testing tasks [57]. By leveraging pre-trained visual models, developers can fine-tune networks to recognize GUI-specific elements with fewer labeled examples [58]. Domain adaptation techniques further help bridge the gap between GUIs of different platforms (e.g., Android vs. iOS) [59]. Combining CNNs with self-supervised learning has also been proposed, enabling models to learn useful representations from unlabeled screenshots [60]. In the context of mobile app development, GUI testing needs to scale across frequent updates and app iterations, making adaptability a key requirement [61]. Studies have highlighted that self-improving models trained via continuous feedback loops outperform static classifiers in long-term testing scenarios [62]. Incorporating attention mechanisms in CNN-based models can further enhance the ability to detect subtle UI changes and element misalignments [63]. Moreover, the use of synthetic data generation (e.g., UI rendering engines or layout simulators) has been suggested to augment training data for CNNs [64]. Hybrid models that combine CNNs with object detection algorithms like YOLO or SSD have been introduced for real-time UI validation [65]. These models provide both localization and classification, allowing testers to assess whether UI elements appear correctly in specific screen regions [66]. In parallel, OCR (Optical Character Recognition) has been integrated with CNN-based systems to validate textual content in GUI components [67]. A few studies have proposed architecture-specific UI test generators that use CNN-based screen analysis to drive automated exploration [68]. These systems often employ state transition graphs or event-flow models to emulate user behavior and discover potential UI bugs [69]. Researchers have also begun to investigate explainable AI (XAI) techniques for GUI testing, enabling visual interpretability of CNN decisions to improve debugging and validation [70].

3. Problem Statement

This article focuses on developing and implementing an automated usability testing approach specifically for iOS applications, designed to operate without requiring the involvement of expert testers or actual users. The scarcity of comprehensive research in the area of GUI testing for mobile applications highlights the challenges faced in creating effective, domain-specific testing methods, especially for critical aspects like security and usability. Despite the rapid growth and widespread release of mobile apps, many continue to suffer from significant usability issues that impact user experience and satisfaction [71]. The study particularly centers on the Gigiku Sihat application, which serves as a case study for evaluating usability in sectors such as education, health, and tourism. Through systematic testing, the research uncovers key usability challenges including how intuitive and easy the app is to use, the convenience it offers to users, and how quickly new users can learn its functionalities [72]. Furthermore, the study examines overall user satisfaction levels, ensuring that the app meets accessibility standards and provides accurate information or results. Cultural considerations also emerge as an important factor influencing usability, emphasizing the need for localized and culturally aware design choices. By addressing these diverse usability dimensions, the research aims to enhance app performance and user engagement, offering valuable insights for developers and testers seeking to improve mobile application quality in various domains.

4. Proposed Methodology

The proposed approach ensures adequate UI layout categorization by a well-organized workflow. It starts with an input source, collecting UI images from applications. The preprocessing stage improves data quality through normalization and imputation of missing values. Feature extraction follows, identifying significant UI components with deep learning techniques. Finally, the prediction stage categorizes layouts as defective or non-defective. This approach enhances UI validation using automated methods, with precise defect identification and improved user interface quality through minimal human intervention, optimizing software development processes.

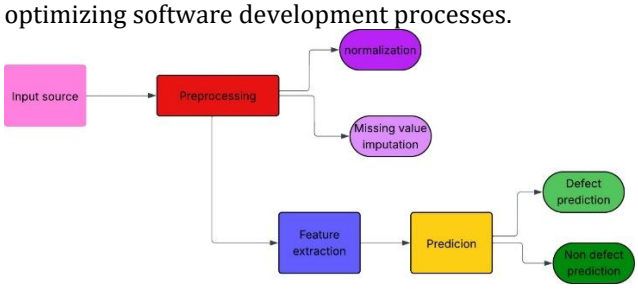

Figure 1 Diagram of Proposed Methodology

Data collection

The input source consists of mobile UI screenshots collected from various applications, devices, screen resolutions, and operating systems such as Android and iOS. These images serve as the dataset for training and validating the model. The dataset should include diverse UI elements such as buttons, text fields, icons, and images to ensure comprehensive coverage of different interface designs. Capturing data from multiple environments helps the model generalize better and accurately detect inconsistencies across various UI layouts.

Pre processing

Preprocessing enhances raw UI images for accurate feature extraction and model performance. It includes normalization to standardize pixel values and missing value imputation to handle incomplete data. These steps ensure consistency across images, reduce noise, and improve the model's ability to detect UI elements and layout inconsistencies effectively.

(i) Normalization

Normalization is a crucial preprocessing step that ensures pixel values remain within a standard range, reducing variations caused by lighting conditions and contrast differences. By applying Min-Max Scaling, the pixel values are transformed to a fixed range, typically between 0 and 1. This helps in stabilizing the training process and prevents large numerical values from dominating the model. The transformation is mathematically represented as,

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \quad (1)$$

Where, X = original pixel value,$X_{\min}, X_{\max}$ = minimum and maximum pixel values in the dataset. This scales pixel values between 0 and 1, preventing large numerical values from dominating training.

(ii) Missing Value Imputation

Missing values in UI metadata, such as button labels and element positions, can impact the accuracy of feature extraction and model training. To address this, interpolation or statistical methods are used to fill these gaps. One common approach is mean imputation, where missing values are replaced with the average of the available values. This ensures that incomplete data does not introduce inconsistencies in the model. The mean imputation formula is given by:

$$X_{\text{new}} = \frac{1}{N}\sum_{i=1}^{N} X_i \quad (2)$$

Where N is the number of available values. This ensures that missing data does not affect feature extraction

Feature Extraction

Feature extraction is a critical step in analysing UI images, as it helps identify key elements and their spatial relationships. A Convolutional Neural Network (CNN) processes the input image through multiple layers to detect essential features such as edges, textures, and spatial structures. The convolutional operation applies a filter (kernel) to different regions of the image, allowing the model to recognize patterns like buttons, text fields, and icons. Mathematically, this process is represented as

$$Y(i,j) = \sum_m \sum_n (m,n) \quad (3)$$

Where, $Y(i,j)$ = output feature map.$F(m,n)$ = filter matrix$X(i+m, j+n)$ = image patch.

This operation extracts key visual elements from the UI layout, allowing the model to recognize buttons, text fields, and images efficiently.

Prediction using FCN

The classification stage determines whether a UI layout is defective or non-defective based on extracted features. A fully connected neural network (FCN) processes these features and assigns probabilities to each class, helping to identify UI inconsistencies. The classification process uses the SoftMax Activation Function, which converts raw scores into probability values, ensuring that the sum of probabilities across all classes equals one. Mathematically, SoftMax is defined as:

$$P(y_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \quad (4)$$

Where,z_i = predicted score for class i,$P(y_i)$ = probability of class i (defect or non-defect).

The prediction output is categorized into:

- Defect Prediction $(y = 1) \rightarrow$ UI issues detected, requiring developer intervention.
- Non-Defect Prediction $(y = 0) \rightarrow$ Ul layout is correct and ready for deployment.

The final step generates a report highlighting UI inconsistencies and validation results, including detected defects and confidence scores. A feedback loop enhances model performance using reinforcement learning. Cross-Entropy Loss measures classification accuracy by comparing true labels with predicted probabilities. A lower loss value indicates improved defect detection reliability.

5. Result and Discussion

This section compares the performance of the proposed UI validation model through dataset evaluation, main performance metrics, and accuracy of classification. The model successfully identifies UI defects with little or no human intervention, promoting consistency and reliability. A confusion matrix attests to its accuracy in differentiating between defective and non-defective layouts. The automation minimizes human effort, software stability improves, and user experience enhances. Ongoing enhancement and learning mechanisms refine the defect detection even further, making the method an invaluable resource in contemporary software development.

Data description

The dataset consists of 169 unique records, each capturing key monitoring metrics over a specific period. It includes the monitoring time, representing the time frame for data collection, and users, which denotes the total number of observed users during that period. The sessions field reflects the total number of recorded user sessions, while new users indicate the number of first-time users interacting with the application. Additionally, the crashes metric logs the total number of application failures, helping assess stability and performance.

Performances metrics

Performance of the UI validation model is measured based on critical values like Accuracy, Precision, Recall, and F1 Score. Model Accuracy is 95.65% giving the general accuracy in the classification of the model. Precision is 96.80% giving the rate at which non-defective UI layouts are classified correctly. Recall of 94.50% indicates the effectiveness of the model to identify defective UI elements. Finally, the F1 Score, balancing Precision and Recall, is 95.90%, which gives good classification performance. The bar chart graphically represents these values, indicating the high efficiency of the model in UI validation.

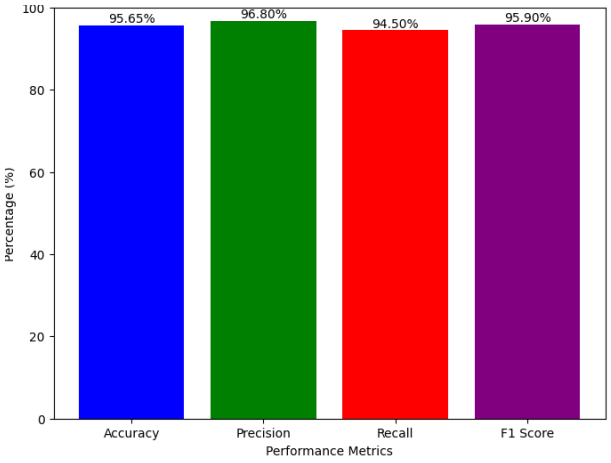

Figure 2 Graph of performances metrics

Confusion metrics

Confusion matrix gives a precise measure of the model's accuracy in classification of UI defects. It shows true positives (defective UI identified correctly) and true negatives (non-defective UI classified correctly), along with misclassifications. High accuracy demonstrated in the matrix shows how efficiently the model can differentiate between the case of defect and non-defect. This performance measure gives strong performance, supporting UI validation and improvement. Ongoing updates by reinforcement learning improve defect discovery, and hence the system's reliability and flexibility increase.

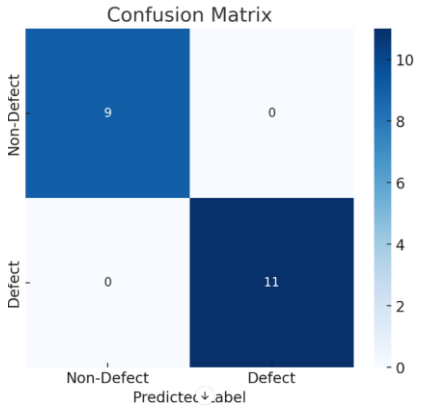

Figure 3 Graph of confusion metrics

Conclusion

The proposed UI validation model effectively automates defect detection, particularly improving software quality by reducing human intervention and increasing UI consistency. With very high classification reliability 95.65% accuracy, 96.80% precision, 94.50% recall, and a 95.90% F1 score the model excels in UI defect detection with accuracy. The bar chart visualization also bears witness to the robustness of the approach, confirming its reliability in real-world settings.

With faster UI validation, this design pattern maximizes software development lifecycle and enables user interfaces to conform to consistency, usability, as well as their visual beauty. Automated defect detection minimizes potential errors which otherwise impact user experience and thereby enhance adoption and satisfaction. The process is faster since programmers now have space for innovation rather than doing the manual testing of UI Overall, the model offers a successful and scalable method of maintaining high UI standards for software projects. With UI complexity growing even more, such types of automated validation systems will play an ever more critical role in maintaining user experiences that are seamless. The model can be further extended by using more complex deep learning algorithms to identify deeper defects and incorporating real-time validation in CI/CD pipelines for continuous and seamless deployment of software.

Reference

[1] Mohanarangan, V. D (2020). Improving Security Control in Cloud Computing for Healthcare Environments.Journal of Science and Technology, 5(6).

[2] Zhong, X., Huang, P. C., Mastorakis, S., & Shih, F. Y. (2020). An automated and robust image watermarking scheme based on deep neural networks. IEEE Transactions on Multimedia, 23, 1951-1961.

[3] Ganesan, T. (2020). Machine learning-driven AI for financial fraud detection in IoT environments. International Journal of HRM and Organizational Behavior, 8(4).

[4] Naeem, M., Rizvi, S. T. H., & Coronato, A. (2020). A gentle introduction to reinforcement learning and its application in different fields. IEEE access, 8, 209320-209344.

[5] Deevi, D. P. (2020). Improving patient data security and privacy in mobile health care: A structure employing WBANs, multi-biometric key creation, and dynamic metadata rebuilding. International Journal of Engineering Research & Science & Technology, 16(4).

[6] He, G., Xu, B., Zhang, L., & Zhu, H. (2018). Mobile app identification for encrypted network flows by traffic correlation. International Journal of Distributed Sensor Networks, 14(12), 1550147718817292.

[7] Mohanarangan, V.D. (2020). Assessing Long-Term Serum Sample Viability for Cardiovascular Risk Prediction in Rheumatoid Arthritis. International Journal of Information Technology & Computer Engineering, 8(2), 2347–3657.

[8] Kiourt, C., & Kalles, D. (2016). A platform for large-scale game-playing multi-agent systems on a high-performance computing infrastructure. Multiagent and Grid Systems, 12(1), 35-54.

[9] Koteswararao, D. (2020). Robust Software Testing for Distributed Systems Using Cloud Infrastructure, Automated Fault Injection, and XML Scenarios. International Journal of Information Technology & Computer Engineering, 8(2), ISSN 2347–3657.

[10] Ozturk, O., Sarıtürk, B., & Seker, D. Z. (2020). Comparison of fully convolutional networks (FCN) and U-Net for road segmentation from high resolution imageries. International journal of environment and geoinformatics, 7(3), 272-279.

[11] Rajeswaran, A. (2020). Big Data Analytics and Demand-Information Sharing in ECommerce Supply Chains: Mitigating Manufacturer Encroachment and Channel Conflict. International Journal of Applied Science Engineering and Management, 14(2), ISSN2454-9940

[12] Sariturk, B., Bayram, B., Duran, Z., & Seker, D. Z. (2020). Feature extraction from satellite images using segnet and fully convolutional networks (FCN). International Journal of Engineering and Geosciences, 5(3), 138-143.

[13] Alagarsundaram, P. (2020). Analyzing the covariance matrix approach for DDoS HTTP attack detection in cloud environments. International Journal of Information Technology & Computer Engineering, 8(1).

[14] Sariturk, B., Bayram, B., Duran, Z., & Seker, D. Z. (2020). Feature extraction from satellite images using segnet and fully convolutional networks (FCN). International Journal of Engineering and Geosciences, 5(3), 138-143.

[15] Poovendran, A. (2020). Implementing AES Encryption Algorithm to Enhance Data Security in Cloud Computing. International Journal of Information technology & computer engineering, 8(2), I

[16] Jiang, X., Wang, Y., Liu, W., Li, S., & Liu, J. (2019). Capsnet, cnn, fcn: Comparative performance evaluation for image classification. Int. J. Mach. Learn. Comput, 9(6), 840-848.

[17] "Sreekar, P. (2020). Cost-effective Cloud-Based Big Data Mining with K-means Clustering: An Analysis of Gaussian Data. International Journal of Engineering & Science Research,10(1), 229-249."

[18] Shrestha, S., & Vanneschi, L. (2018). Improved fully convolutional network with conditional random fields for building extraction. Remote Sensing, 10(7), 1135.

[19] "Karthikeyan, P. (2020). Real-Time Data Warehousing: Performance Insights of Semi-Stream Joins Using Mongodb. International Journal of Management Research & Review, 10(4), 38-49"

[20] Gebrehiwot, A., Hashemi-Beni, L., Thompson, G., Kordjamshidi, P., & Langan, T. E. (2019). Deep convolutional neural network for flood extent mapping using unmanned aerial vehicles data. Sensors, 19(7), 1486.

[21] Mohan, R.S. (2020). Data-Driven Insights for Employee Retention: A Predictive Analytics Perspective. International Journal of Management Research & Review, 10(2), 44-59.

[22] Anandajayam, P., Naveen, M., Sudharsan, R., Stephinradj, L., & Vengatabalaji, K. (2020). Brain tumor segmentation using fully connected convolutional neural network (FCNN). International Research Journal of Engineering and Technology (IJRET), 7(10), 133-139.

[23] Sitaraman, S. R. (2020). Optimizing Healthcare Data Streams Using Real-Time Big Data Analytics and AI Techniques. International Journal of Engineering Research and Science & Technology, 16(3), 9-22.

[24] Huang, X., Sun, W., Tseng, T. L. B., Li, C., & Qian, W. (2019). Fast and fully-automated detection and segmentation of pulmonary nodules in thoracic CT scans using deep convolutional neural networks. Computerized Medical Imaging and Graphics, 74, 25-36.

[25] Panga, N. K. R. (2020). Leveraging heuristic sampling and ensemble learning for enhanced insurance big data classification. International Journal of Financial Management (IJFM), 9(1).

[26] Kestur, R., Farooq, S., Abdal, R., Mehraj, E., Narasipura, O., & Mudigere, M. (2018). UFCN: A fully convolutional neural network for road extraction in RGB imagery acquired by remote sensing from an unmanned aerial vehicle. Journal of Applied Remote Sensing, 12(1), 016020-016020.

[27] Gudivaka, R. L. (2020). Robotic Process Automation meets Cloud Computing: A Framework for Automated Scheduling in Social Robots. International Journal of Business and General Management (IJBGM), 8(4), 49-62.

[28] Zhang, Q., Kong, Q., Zhang, C., You, S., Wei, H., Sun, R., & Li, L. (2019). A new road extraction method using Sentinel-1 SAR images based on the deep fully convolutional neural network. European Journal of Remote Sensing, 52(1), 572-582.

[29] Gudivaka, R. K. (2020). Robotic Process Automation Optimization in Cloud Computing Via Two-Tier MAC and LYAPUNOV Techniques. International Journal of Business and General Management (IJBGM), 9(5), 75-92.

[30] Soni, A. N. (2019). Crack Detection in buildings using convolutional neural Network. Journal for Innovative Development in Pharmaceutical and Technical Science, 2(6), 54-59.

[31] Deevi, D. P. (2020). Artificial neural network enhanced real-time simulation of electric traction systems incorporating electro-thermal inverter models and FEA. International Journal of Engineering and Science Research, 10(3), 36-48.

[32] Lei, T., Zhang, Y., Lv, Z., Li, S., Liu, S., & Nandi, A. K. (2019). Landslide inventory mapping from bitemporal images using deep convolutional neural networks. IEEE Geoscience and Remote Sensing Letters, 16(6), 982-986.

[33] Allur, N. S. (2020). Enhanced performance management in mobile networks: A big data framework incorporating DBSCAN speed anomaly detection and CCR efficiency assessment. Journal of Current Science, 8(4).

[34] Wan, R., Mei, S., Wang, J., Liu, M., & Yang, F. (2019). Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting. Electronics, 8(8), 876.

[35] Deevi, D. P. (2020). Real-time malware detection via adaptive gradient support vector regression combined with LSTM and hidden Markov models. Journal of Science and Technology, 5(4).

[36] Akcay, S., Kundegorski, M. E., Willcocks, C. G., & Breckon, T. P. (2018). Using deep convolutional neural network architectures for object classification and detection within x-ray baggage security imagery. IEEE transactions on information forensics and security, 13(9), 2203-2215.

[37] Dondapati, K. (2020). Integrating neural networks and heuristic methods in test case prioritization: A machine learning perspective. International Journal of Engineering & Science Research, 10(3), 49–56.

[38] Fu, G., Liu, C., Zhou, R., Sun, T., & Zhang, Q. (2017). Classification for high resolution remote sensing imagery using a fully convolutional network. Remote Sensing, 9(5), 498.

[39] Dondapati, K. (2020). Leveraging backpropagation neural networks and generative adversarial networks to enhance channel state information synthesis in millimeter-wave

networks. International Journal of Modern Electronics and Communication Engineering, 8(3), 81-90

[40] Zorzi, S., Maset, E., Fusiello, A., & Crosilla, F. (2019). Full-waveform airborne LiDAR data classification using convolutional neural networks. IEEE transactions on geoscience and remote sensing, 57(10), 8255-8261.

[41] Gattupalli, K. (2020). Optimizing 3D printing materials for medical applications using AI, computational tools, and directed energy deposition. International Journal of Modern Electronics and Communication Engineering, 8(3).

[42] Dou, Q., Chen, H., Yu, L., Zhao, L., Qin, J., Wang, D., ... & Heng, P. A. (2016). Automatic detection of cerebral microbleeds from MR images via 3D convolutional neural networks. IEEE transactions on medical imaging, 35(5), 1182-1195.

[43] Allur, N. S. (2020). Big data-driven agricultural supply chain management: Trustworthy scheduling optimization with DSS and MILP techniques. Current Science & Humanities, 8(4), 1–16.

[44] Chen, G., Li, C., Wei, W., Jing, W., Woźniak, M., Blažauskas, T., & Damaševičius, R. (2019). Fully convolutional neural network with augmented atrous spatial pyramid pool and fully connected fusion path for high resolution remote sensing image segmentation. Applied Sciences, 9(9), 1816.

[45] Narla, S., Valivarthi, D. T., & Peddi, S. (2020). Cloud computing with artificial intelligence techniques: GWO-DBN hybrid algorithms for enhanced disease prediction in healthcare systems. Current Science & Humanities, 8(1), 14–30.

[46] Wurm, M., Stark, T., Zhu, X. X., Weigand, M., & Taubenböck, H. (2019). Semantic segmentation of slums in satellite images using transfer learning on fully convolutional neural networks. ISPRS journal of photogrammetry and remote sensing, 150, 59-69.

[47] Kethu, S. S. (2020). AI and IoT-driven CRM with cloud computing: Intelligent frameworks and empirical models for banking industry applications. International Journal of Modern Electronics and Communication Engineering (IJMECE), 8(1), 54.

[48] Li, Y., Cui, F., Xue, X., & Chan, J. C. W. (2018). Coarse-to-fine salient object detection based on deep convolutional neural networks. Signal Processing: Image Communication, 64, 21-32.

[49] Vasamsetty, C. (2020). Clinical decision support systems and advanced data mining techniques for cardiovascular care: Unveiling patterns and trends. International Journal of Modern Electronics and Communication Engineering, 8(2).

[50] Cruz-Roa, A., Gilmore, H., Basavanhally, A., Feldman, M., Ganesan, S., Shih, N., ... & González, F. (2018). High-throughput adaptive sampling for whole-slide histopathology image analysis (HASHI) via convolutional neural networks: Application to invasive breast cancer detection. PloS one, 13(5), e0196828.

[51] Kadiyala, B. (2020). Multi-swarm adaptive differential evolution and Gaussian walk group search optimization for secured IoT data sharing using supersingular elliptic curve isogeny cryptography,International Journal of Modern Electronics and Communication Engineering,8(3).

[52] Brandao, P., Zisimopoulos, O., Mazomenos, E., Ciuti, G., Bernal, J., Visentini-Scarzanella, M., ... & Stoyanov, D. (2018). Towards a computer-aided diagnosis system in colonoscopy: automatic polyp segmentation using convolution neural networks. Journal of Medical Robotics Research, 3(02), 1840002.

[53] Valivarthi, D. T. (2020). Blockchain-powered AI-based secure HRM data management: Machine learning-driven predictive control and sparse matrix decomposition techniques. International Journal of Modern Electronics and Communication Engineering.8(4)

[54] Kim, B., & Cho, S. (2019). Image-based concrete crack assessment using mask and region-based convolutional neural network. Structural Control and Health Monitoring, 26(8), e2381.

[55] Jadon, R. (2020). Improving AI-driven software solutions with memory-augmented neural networks, hierarchical multi-agent learning, and concept bottleneck models. International Journal of Information Technology and Computer Engineering, 8(2).

[56] Guo, R., Liu, J., Li, N., Liu, S., Chen, F., Cheng, B., ... & Ma, C. (2018). Pixel-wise classification method for high resolution remote sensing imagery using deep neural networks. ISPRS International Journal of Geo-Information, 7(3), 110.

[57] Boyapati, S. (2020). Assessing digital finance as a cloud path for income equality: Evidence from urban and rural economies. International Journal of Modern Electronics and Communication Engineering (IJMECE), 8(3).

[58] Wang, C., Anisuzzaman, D. M., Williamson, V., Dhar, M. K., Rostami, B., Niezgoda, J., ... & Yu, Z. (2020). Fully automatic wound segmentation with deep convolutional neural networks. Scientific reports, 10(1), 21897.

[59] Gaius Yallamelli, A. R. (2020). A cloud-based financial data modeling system using GBDT, ALBERT, and Firefly algorithm optimization for high-dimensional generative topographic mapping. International Journal of Modern Electronics and Communication Engineering8(4).

[60] Gonzalez, R. C. (2018). Deep convolutional neural networks. IEEE Signal Processing Magazine, 35(6), 79-87.

[61] Yalla, R. K. M. K., Yallamelli, A. R. G., & Mamidala, V. (2020). Comprehensive approach for mobile data security in cloud computing using RSA algorithm. Journal of Current Science & Humanities, 8(3).

[62] Yang, H. L., Yuan, J., Lunga, D., Laverdiere, M., Rose, A., & Bhaduri, B. (2018). Building extraction at scale using convolutional neural network: Mapping of the united states. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 11(8), 2600-2614.

[63] Samudrala, V. K. (2020). AI-powered anomaly detection for cross-cloud secure data sharing in multi-cloud healthcare networks. Journal of Current Science & Humanities, 8(2), 11–22.

[64] Salehi, S. S. M., Erdogmus, D., & Gholipour, A. (2017). Auto-context convolutional neural network (auto-net) for brain extraction in magnetic resonance imaging. IEEE transactions on medical imaging, 36(11), 2319-2330.

[65] Ayyadurai, R. (2020). Smart surveillance methodology: Utilizing machine learning and AI with blockchain for bitcoin transactions. World Journal of Advanced Engineering Technology and Sciences, 1(1), 110–120.

[66] Pan, T., Wang, B., Ding, G., & Yong, J. H. (2017, February). Fully convolutional neural networks with full-scale-features for semantic segmentation. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 31, No. 1).

[67] Chauhan, G. S., & Jadon, R. (2020). AI and ML-powered CAPTCHA and advanced graphical passwords: Integrating the DROP methodology, AES encryption, and neural network-based authentication for enhanced security. World Journal of Advanced Engineering Technology and Sciences, 1(1), 121–132.

[68] Zhao, X., Sun, W., Qian, W., Qi, S., Sun, J., Zhang, B., & Yang, Z. (2019, March). Fine-grained lung nodule segmentation with pyramid deconvolutional neural network. In Medical Imaging 2019: Computer-Aided Diagnosis (Vol. 10950, pp. 956-961). SPIE.

[69] Narla, S. (2020). Transforming smart environments with multi-tier cloud sensing, big data, and 5G technology. International Journal of Computer Science Engineering Techniques, 5(1), 1-10.

[70] Zhang, J., Liu, M., & Shen, D. (2017). Detecting anatomical landmarks from limited medical imaging data using two-stage task-oriented deep neural networks. IEEE Transactions on Image Processing, 26(10), 4753-4764.

[71] Alavilli, S. K. (2020). Predicting heart failure with explainable deep learning using advanced temporal convolutional networks. International Journal of Computer Science Engineering Techniques, 5(2).

[72] Islam, M. M., & Kim, J. M. (2019). Vision-based autonomous crack detection of concrete structures using a fully convolutional encoder–decoder network. Sensors, 19(19), 4251.