World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(REVIEW ARTICLE)

Check for updates

Enhancing agricultural health with AI: Drone-based machine learning for mango tree disease detection

Ali Husnain [1,*], Aftab Ahmad [2] and Ayesha Saeed [3]

[1] *Department of Computer Science, University of Chicago, USA.*
[2] *Department of Computer Science, American National University, USA.*
[3] *Department of Computer Science, University of Lahore, Lahore, Pakistan.*

World Journal of Advanced Research and Reviews, 2024, 23(02), 1267–1276

Publication history: Received on 02 July 2024; revised on 13 August 2024; accepted on 15 August 2024

Article DOI: https://doi.org/10.30574/wjarr.2024.23.2.2455

Abstract

In the agriculture sector, timely detection of diseases in fruit trees is a significant challenge, leading to substantial economic losses. Automated detection of diseases in fruit trees, particularly mango trees, is crucial to minimize these losses by enabling early intervention. This research explores the use of drone-captured multispectral images combined with deep learning and computer vision techniques to detect diseases in mango trees. The proposed system leverages various pre-trained Convolutional Neural Network (CNN) models, such as YOLOv5, Detectron2, and Faster R-CNN, to achieve optimal accuracy. Data augmentation techniques are employed to address data skewness and overfitting, while Generative Adversarial Networks (GANs) enhance image quality. The system aims to provide a scalable solution for early disease detection, thereby reducing economic losses and supporting the agricultural sector's growth.

Keywords: YOLOv5; Drone; Convolutional Neural Network (CNN); Generative Adversarial Networks (GANs); WebODM

1. Introduction

Agriculture plays a pivotal role in the economies of many countries, particularly in Pakistan, where it contributes significantly to the GDP. Among various agricultural products, mangoes hold a substantial market value due to their high demand globally. However, the cultivation of mangoes is plagued by various diseases that can severely impact yield and quality. Traditional methods of disease detection, which rely on human expertise, are often slow, labor-intensive, and inadequate for large-scale farms. This research addresses these challenges by developing an automated system for detecting diseases in mango trees using drone images and advanced machine learning techniques.

The objective of this research is to improve the efficiency and accuracy of disease detection in mango trees. By integrating artificial intelligence (AI) and deep learning technologies, the proposed system seeks to enhance the agricultural sector's productivity and reduce the economic impact of crop diseases. The study employs a DJI Air 2s drone to capture multispectral images of mango trees, which are then processed using state-of-the-art CNN models to detect diseases.

2. Literature Review

In recent years, the integration of drone technology with advanced image processing techniques has significantly enhanced the detection and monitoring of plant diseases. Drones equipped with high-resolution cameras and various sensors enable the rapid identification of plant pathogens at early stages, which is crucial for minimizing crop damage

* Corresponding author: Ali Husnain

and reducing pesticide costs. The flexibility, low cost, and high efficiency of drones make them an ideal tool for large-scale agricultural monitoring, although their limited flight duration and payload capacity remain challenges. Despite these limitations, the adoption of drones in agriculture represents a critical advancement in the field, offering precise and timely data for effective crop management. [1]

A disease detection technique utilizing the YOLOv3 deep learning model, which is extensively used for semantic segmentation and object detection, was proposed. The researchers collected a dataset of tomato images from planting greenhouses using digital and smartphone cameras. These images were annotated using the bounding box technique, which effectively reduces the search range for object features, thereby lowering computational resource demands. The model achieved an accuracy of 92.39% with a detection time of 20.39 milliseconds. [2]

In another study, six different Convolutional Neural Network (CNN) models were combined and fine-tuned using an ensemble technique. The performance of various model combinations was then evaluated using Support Vector Machine (SVM). The study employed the Turkey-Plant Dataset, which includes images of 15 different plant diseases. These images were captured using a Nikon 7200d camera with a resolution of 4000x6000 pixels. The majority voting ensemble model used in this study managed to achieve an impressive accuracy of 97.56%. [3]

A dataset of RGB images showing symptoms of apple foliar diseases was collected using a Canon Rebel T5i DSLR camera. The images were manually annotated with the assistance of an expert pathologist. This dataset contained four types of apple plant diseases. The images were captured under varying illumination conditions, angles, and surfaces, which introduced significant noise into the dataset. For the classification task, the researchers fine-tuned the weights of the pre-trained deep learning model ResNet50, which is well-suited for computer vision tasks such as object detection and image localization. The study achieved an accuracy of 97%. [4]

An ensemble model combining the pre-trained models DenseNet121, EfficientNetB7, and EfficientNet NoisyStudent was proposed to detect diseases in apple leaves. The researchers collected 3,651 high-quality images of apple leaves with various foliar diseases and applied several image augmentation techniques to expand the dataset size. The ensemble model achieved an accuracy of 96.25%, while DenseNet121, EfficientNetB7, and NoisyStudent individually achieved accuracies of 95.26%, 95.62%, and 91.24%, respectively. [5]

A study leveraging aerial images of crops and an intuitive fuzzy set (IFS) approach significantly enhanced the accuracy of field-based disease diagnosis in plants. The researchers focused on isolating diseased regions from UAV-captured images. The proposed method was tested on numerous aerial images of crops, and the results indicated that the segmented images produced using the IFS technique were more distinct and better clustered compared to those generated by CNN, KNN, and FCN techniques. The segmentation process achieved an accuracy of 94%, proving effective in detecting the spread of diseases across crop bodies. [6]

This paper provided a comprehensive review of state-of-the-art CNN models for diagnosing and categorizing plant diseases. The focus was on identifying and classifying diseases in eleven economically significant plant species. The review covered papers published from 2016 to 2021, examining various CNN models used for disease detection, image preprocessing and augmentation techniques, data sources, and performance metrics. Preprocessing techniques discussed include cropping, resizing, contrast enhancement, normalization, affine transformation, and perspective transformation. The CNN architectures reviewed include AlexNet, SVM, RF, DCNN, VGG-S, VGG-VD-19, Inception v3, GoogLeNet, DenseNet 169, SqueezeNet 1.1, and VGG 13. The paper also highlighted challenges and issues related to plant disease detection. [7]

A classifier based on the pre-trained AlexNet deep learning model was developed to identify diseases on mango and grape leaves. The model was trained on a dataset comprising 1,266 mango leaf images, which were self-acquired from production fields, and 7,222 grape leaf images obtained from the publicly available Plant Village dataset. The proposed model demonstrated its effectiveness by achieving an accuracy of approximately 99.03% when classifying previously unseen grape leaves.[8]

In a study utilizing high-fidelity Datagen, an adversarial network was employed to augment a small dataset of apple plant leaves. The augmented dataset was then passed through Low-Fidelity Datagen to generate synthetic images of crop fields. The dataset, consisting of 1,821 labeled images, was classified into four classes using the EfficientNet architecture, a high-performing industry standard model, and EfficientDet. The study achieved a 70% accuracy, with the potential for improvement by refining the identifiers and using more accurate labeling techniques.[9]

3. Research Methodology

The research methodology employed in this study is structured around five key phases: Data Acquisition, Data Preprocessing, Model Selection and Training, Model Validation, and Model Testing. Each phase is carefully designed to ensure the accuracy, reliability, and generalizability of the disease detection system.

3.1. Data Acquisition

Data acquisition is the foundational step in this research, where high-quality, multispectral images of mango trees are captured. The following steps outline the data acquisition process:

Drone Selection: A DJI Air 2s drone is selected for its ability to capture high-resolution multispectral images, which include Infrared, Grayscale, and RGB channels. These images are crucial for detecting subtle differences in the health of mango trees that might not be visible in standard RGB images.

Image Capture Protocol: The drone is flown at an altitude of 100 feet over mango orchards, ensuring that the images have sufficient detail while covering a large area. The flight path is programmed to achieve a 60% overlap between consecutive images, which is necessary for accurate image stitching in later stages. The dataset comprises 8,905 images covering three distinct land areas, ensuring diversity in the captured data.

Environmental Conditions: Image capture is conducted under various lighting conditions and at different times of the day to ensure that the dataset includes a range of scenarios that the model might encounter in real-world applications. This variation is essential for training a robust model that can generalize well.

3.2. Data Preprocessing

Data preprocessing is critical for preparing the raw images for analysis. The following steps describe the preprocessing techniques applied:

Image Stitching: Given the 60% overlap between images, an image stitching process is employed to create comprehensive, high-resolution maps of the mango orchards. Software such as WebODM is used for this purpose, which aligns and blends the images seamlessly. The stitched images provide a holistic view of the orchard, essential for accurate disease detection.

Data Augmentation: To address potential issues of data imbalance and overfitting, data augmentation techniques are applied. These include:

- Cropping: Random crops are taken from the images to create additional training samples.
- Scaling and Rotation: Images are scaled and rotated to simulate different perspectives and conditions, enhancing the model's robustness.
- Flipping and Adding Noise: Horizontal and vertical flips are applied, along with Gaussian noise, to further diversify the dataset.
- Image Quality Enhancement: Generative Adversarial Networks (GANs) are employed to enhance the quality of the images. GANs generate synthetic images that resemble real-world scenarios but with enhanced clarity and detail. This step is crucial for improving the accuracy of object detection and semantic segmentation tasks.

3.3. Model Selection and Training

The selection and training of models are pivotal to the success of the disease detection system. The following models and techniques are utilized:

- Model Selection: Three state-of-the-art Convolutional Neural Network (CNN) models are selected based on their performance in object detection and classification tasks:
- YOLOv5: Known for its real-time detection capabilities and high accuracy, YOLOv5 is used for initial detection tasks. Its ability to process images quickly makes it ideal for applications requiring prompt responses.
- Detectron2: This model is chosen for its advanced capabilities in semantic segmentation, enabling the precise identification of diseased areas on the mango trees.
- Faster R-CNN: Selected for its balance between detection speed and accuracy, Faster R-CNN is used to refine the detection process, particularly in complex scenarios with overlapping leaves or fruits.

- Model Training: The training process involves feeding the preprocessed images into the selected models. Key aspects of the training process include:
- Training-Validation Split: The dataset is divided into training and validation sets, ensuring that the model is evaluated on unseen data during training. The split is done in such a way that each set contains a balanced representation of healthy and diseased trees.
- Loss Function and Optimization: Cross-entropy loss is used as the loss function, with the Adam optimizer employed to adjust the model weights during training. The learning rate is fine-tuned through experiments to achieve the best balance between convergence speed and accuracy.
- Data Augmentation during Training: The augmented dataset is fed into the models during training to prevent overfitting and improve generalization. Each model is trained for a fixed number of epochs, with early stopping implemented to avoid overfitting.

3.4. Model Validation

Model validation is a critical step that ensures the trained models meet the intended performance criteria before being deployed for real-world applications. The validation process involves:

- Purpose of Validation: Model validation serves to confirm that the model achieves its intended purpose and performs well on unseen data. It is during this phase that potential issues such as overfitting or underfitting are identified.
- Validation Techniques: The model's performance is validated using a separate validation dataset, distinct from the training data. Metrics such as accuracy, precision, recall, and F1-score are used to assess the model's performance. Additionally, cross-validation techniques may be employed to ensure the model's robustness across different subsets of the data.
- Identification of Issues: During validation, the model is monitored for signs of overfitting (where the model performs well on training data but poorly on unseen data) or underfitting (where the model fails to capture the underlying patterns in the data). Adjustments to the model architecture, hyperparameters, or training data may be made based on the validation results.

3.5. Model Testing

The final phase of the research methodology is the rigorous testing of the validated model to ensure its readiness for deployment. The testing process involves multiple approaches and techniques:

- Manual Testing: In manual testing, the model is tested by manually inputting data and observing the outputs. This approach helps identify any obvious errors or bugs in the system. Manual testing is useful for validating the model's basic functionality before moving on to more automated testing methods.
- Automated Testing: Automated testing involves the use of software tools to execute test cases and evaluate the model's performance. Automated testing is essential for efficiently testing large datasets and performing complex tests such as load testing and regression testing. This approach ensures that the model performs reliably under different conditions and workloads. As automation testing becomes increasingly necessary due to the complexity and shorter development cycles of current software projects, it is crucial to plan and implement these tests properly to avoid increased costs and less effective outcomes. [10]

3.6. Testing Approaches

The system is subjected to both functional and structural testing:

- Functional Testing: This approach verifies that the model's outputs meet the expected results based on the given inputs, without considering the internal workings of the model.
- Structural Testing: This approach examines the internal structure and implementation of the model to ensure that it operates correctly and efficiently. Structural testing includes techniques such as white-box testing, where the tester has access to the internal workings of the model.

3.7. Testing Methods

Black-Box Testing: In black-box testing, the model is tested based solely on its inputs and outputs, without any knowledge of the internal implementation. This method focuses on the correctness of the output and is particularly useful for end-user testing scenarios. Black box testing treats the software as a “Black Box” – without any knowledge of internal workings, it only examines the fundamental aspects of the system. [11]

White-Box Testing: In white-box testing, the tester has access to the internal structure and logic of the model. This method is used to verify that the model's internal processes are functioning as expected and to optimize the model's overall efficiency and arrangement.

Field Validation: The final model is tested in a real-world scenario where its predictions are compared against expert assessments to validate its practical utility. This step ensures that the model is not only accurate but also practical and reliable in real-world applications.

3.8. Test Cases

3.8.1. TC01: Dataset Acquisition

Table 1 TC-01

Test case ID	Test Title	Test cases Description	Pre-conditions	Test steps	Expected result	Actual result	Status
TC-01	Check Dataset acquisition	Testing the gathering of the dataset and saving it in proper format at a secure drive	1. User must have UAV for dataset and know how to operate it and at what height 2. must have sufficient drive space for the dataset.	1. Fly the drone and gather images 2. Save them 3. Convert into digital form	1. Images are captured in TIFF or jpg or png format 2. They are saved in drive 3. Converted into digital form as an array	1. Images are captured in TIFF or jpg or png format 2. They are saved in drive 3. Converted into digital form as an array	Pass

3.8.2. TC02: Dataset Stitching

Table 2 TC-02

Test case ID	Test Title	Test cases Description	Pre-conditions	Test steps	Expected result	Actual result	Status
TC-01	Verification of Stitching of Data.	Verify if the Data is stitched without issues	1.There must be a dataset selected 2.There must of at least 60% overlapping among the captured Images 3.More System RAM will be required for high-quality stitching 4. At least 8GB RAM is preferable	1. Stitch the data using some tool such as WebODM 2. Check the Data Stitching results	1. Dataset should be stitched without any issues	1. Dataset should be stitched without any issues	Pass

3.8.3. TC03: Data Annotation

Table 3 TC-03

Test case ID	Test Title	Test cases Description	Pre-conditions	Test steps	Expected result	Actual result	Status
TC-01	Verification of Dataset annotation	Verification of right and trusted tool used for object selection and then resultant annotations are exported in the correct format	1. Must have a dataset and a tool for annotations 2. Running System, Laptop, PC	1. choose a trusted tool. E.g: labelImg, Makesense.ai, Roboflow.. 2. mention classes 3. Choose the best feasible object selection shape. 3. Check exports and their formats	1. All images are annotated 2. Annotations are saved in the drive 3. Converted into digital form as an array.	1. All images are annotated 2. Annotations are saved in the drive 3. Converted into digital form as an array	Pass

Model Selection for Object Deletion And Classification

- Yolo v5
- Yolo v7
- Detectron2
- FasterRCNN

3.8.4. TC04 Object Detection

Table 4 TC-04

Test case ID	Test Title	Test cases Description	Pre-conditions	Test steps	Expected result	Actual result	Status
TC-01	Verification of object Detection	verify if the objects in dataset images are detected using object detection algorithms	1. Must have a dataset and the algorithm selected 2. An IDE or Code Editor 3. Running System, Laptop, PC	1. Run the algorithm on the dataset for all the phases of the model 2. Check the object detection results	1. Objects are detected with high accuracy 2. Related Graphs and Other metrics Information	1. Objects are detected with high accuracy 2. Related Graphs and Other metrics Information	Pass

3.8.5. TC05: Feature Extraction

Table 5 TC-05

Test case ID	Test Title	Test cases Description	Pre-conditions	Test steps	Expected result	Actual result	Status
TC-01	Verification of Feature Extraction from the dataset	verify if the feature is extracted without issues	1. Must have a dataset and the algorithm selected 2. An IDE or Code Editor 3. Running System, Laptop, PC	1. Run the algorithm on the dataset 2. Check the extraction results	1. Features are extracted accurately without any issues 2. Related Graphs and Other metrics Information	1. Features are extracted accurately without any issues 2. Related Graphs and Other metrics Information	Pass

3.8.6. TC06: Model Training

Table 6 TC-06

Test case ID	Test Title	Test cases Description	Pre-conditions	Test steps	Expected result	Actual result	Status
TC-01	Verification of Training of model	verify if the model is trained without any problem	1. Must have a dataset and the algorithm selected 2. Must have loss function, optimizer and evaluative matric 3. Running System, Laptop, PC 4. An IDE or Code Editor	1. Run the algorithm on the dataset 2. Check the model training results	1. Model is trained without issues	1. Model is trained without issues	Pass

3.8.7. TC07: Model Testing

Table 7 TC-07

Test case ID	Test Title	Test cases Description	Pre-conditions	Test steps	Expected result	Actual result	Status
TC-01	Verification of test of model	Check if the model is tested without issues	1. Must have a trained model for testing 2. Must have loss function, optimizer and evaluative matric	1. Run the testing dataset on the trained model 2. Check the model testing results	1. Model is tested, and the accuracy is high as desired	1. Model is tested, and the accuracy is high as desired	Pass

			3. Running System, Laptop, PC 4. An IDE or Code Editor				

3.8.8. TC08: Disease Prediction

Table 8 TC-08

Test case ID	Test Title	Test cases Description	Pre-conditions	Test steps	Expected result	Actual result	Status
TC-01	Verification of Tree's Diseases	verify if the plants in image dataset are properly labelled as well as use appropriate algorithms	1. Must have a dataset and the algorithm selected 2. Must have loss function, optimizer and evaluative matric 3. Running System, Laptop, PC 4. An IDE or Code Editor	1. Train the algorithm on the dataset. 2. Output the Disease Prediction results.	1. Tree's Diseases are displayed with high accuracy.	1. Tree's Diseases are displayed with high accuracy.	Pass

3.8.9. TC09 Data Pre-processing

Table 9 TC-09

Test case ID	Test Title	Test cases Description	Pre-conditions	Test steps	Expected result	Actual result	Status
TC-01	Verify Pre-processing of Data	verify if images are properly tiled, augmented, resized	1. Must have a dataset and the algorithm selected 2. Must have an augmentation software or code 3. Running System, Laptop, PC 4. An IDE or Code Editor	1. Augment, Resize, Tile the dataset. 2. Output the pre-processed dataset	1. Output pre-processed data	1. Output pre-processed data	Pass

3.9. Testing Levels

Testing levels are critical evaluation methods used to assess the performance and reliability of a machine learning model. These levels involve systematically testing the model across various stages of complexity to ensure it meets the desired quality standards and performs as expected under different conditions. The key levels of testing include:

Unit Testing: Unit testing is the initial phase of testing where individual units or components of a software system are tested in isolation. This method focuses on ensuring that each unit functions correctly on its own. Typically performed by developers, unit testing is crucial because it allows early identification and resolution of defects, making it easier and less costly to fix problems. By validating the smallest components of the system independently, unit testing ensures that the overall software remains reliable and of high quality before it moves to more complex testing phases. In addition to identifying defects early in the development process, unit testing helps to ensure that each module of the software is functioning correctly before it is integrated with other components. This level of testing is essential because it allows for the detection of issues at a granular level, reducing the risk of compounding errors during subsequent testing stages. [12]

Integration Testing: Integration testing follows unit testing and involves testing the interactions and integrations between different units or components of the software system. The primary goal is to ensure that these units work together seamlessly and meet the specified requirements. Integration testing can be approached in two ways:

Top-Down Approach: Higher-level modules are tested first, followed by testing of the lower-level modules.

Bottom-Up Approach: Lower-level modules are tested first, with testing gradually moving up to the higher-level modules.

This testing phase ensures that the integrated units function correctly as a combined system and that any issues arising from their interactions are identified and resolved.

System Testing: System testing is a comprehensive method that involves evaluating the entire system, including all its integrated components, to ensure it functions as a cohesive unit. The goal of system testing is to verify that the system as a whole meets the specified requirements and performs well within its intended operational environment. This type of testing is typically carried out by a specialized testing team that focuses on ensuring the system aligns with business requirements and operates smoothly in real-world conditions.

Acceptance Testing: Acceptance testing is the final phase in the testing process. At this level, the system undergoes rigorous testing to determine whether it meets the end-user requirements and is ready for deployment. Acceptance testing evaluates the model from the perspective of the end user, ensuring that the system is not only functionally correct but also practical and ready for real-world application. The successful completion of acceptance testing signals that the system is prepared for implementation and use in its intended environment.

Testing Sequence and Complexity: The sequence and complexity of testing progress from unit testing, which focuses on individual components, to integration testing, which examines how these components work together, followed by system testing, which evaluates the entire system, and culminating in acceptance testing, where the system is assessed for deployment readiness. This structured approach ensures that the system is robust, reliable, and capable of meeting both technical and business requirements.

4. Conclusion

This research has successfully demonstrated the potential of utilizing AI-driven machine learning and computer vision techniques for the automated detection of diseases in mango trees. The project faced several challenges, including data availability, diversity, image quality, and issues related to data skewness and overfitting. Despite these challenges, the use of drones to capture multispectral images, combined with advanced models such as CNNs and GANs, allowed for the effective processing and analysis of the data. The implementation of data augmentation techniques further helped mitigate some of the issues related to data imbalance and overfitting, enhancing the accuracy of the disease detection model. The findings underscore the importance of high-quality, diverse datasets and the need for sophisticated image processing methods to improve the reliability of automated disease detection systems in agriculture. The integration of this technology into smart farming practices holds significant promise for enhancing crop management, allowing for early detection of diseases and more efficient resource allocation. Future research should explore the application of these techniques to a broader range of crops and consider incorporating additional data sources, such as environmental factors, to further refine the model's predictive capabilities. As smart farming and precision agriculture continue to evolve, the adoption of AI-driven solutions like the one presented in this study will be crucial in addressing the challenges of modern agriculture and ensuring sustainable crop production.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Daund, R. P., Mate, A., Shinde, M., Kadam, K., & Sonawane, P. (2024). Prediction of Plant Leaf Diseases using Drone and Image Processing Techniques. International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), 4(1), 8-14. https://doi.org/10.48175/IJARSCT-15002

[2] Liu, Jun, and Xuewei Wang. "Tomato diseases and pests detection based on improved Yolo V3 convolutional neural network." Frontiers in plant science 11 (2020): 898.

[3] Turkoglu, Muammer, Berrin Yanikoğlu, and Davut Hanbay. "PlantDiseaseNet: Convolutional neural network ensemble for plant disease and pest detection." Signal, Image and Video Processing 16.2 (2022): 301-309.

[4] Thapa, Ranjita, et al. "The Plant Pathology Challenge 2020 data set to classify foliar disease of apples." Applications in Plant Sciences 8.9 (2020): e11390.

[5] Kumar, R., S. Kumar, and P. Bansal. "Disease detection in apple leaves using deep convolutional neural network." (2021)

[6] Vinita, V., & Dawn, S. (2022, August). Intuitionistic Fuzzy Representation of Plant Images captured using Unmanned Aerial Vehicle for Measuring Mango Crop Health. In Proceedings of the 2022 Fourteenth International Conference on Contemporary Computing (pp. 190-195).

[7] Joseph, D. S., Pawar, P. M., & Pramanik, R. (2022). Intelligent plant disease diagnosis using convolutional neural network: a review. Multimedia Tools and Applications, 1-67.

[8] Rao, U. S., Swathi, R., Sanjana, V., Arpitha, L., Chandrasekhar, K., & Naik, P. K. (2021). Deep learning precision farming: grapes and mango leaf disease detection by transfer learning. Global Transitions Proceedings, 2(2), 535-544.

[9] Prasad, A., Mehta, N., Horak, M., & Bae, W. D. (2021). A two-step machine learning approach for crop disease detection: an application of GAN and UAV technology. arXiv preprint arXiv:2109.11066.

[10] Dahiya, R., & Ali, S. (2019). Importance of Manual and Automation Testing. Proceedings of the 9th International Conference on Computer Science, Engineering and Information Technology (CSIT-2019), 159-168.

[11] Khan, M. E., & Khan, F. (2012). A Comparative Study of White Box, Black Box and Grey Box Testing Techniques. International Journal of Advanced Computer Science and Applications, 3(6), 12-15.

[12] Umar, M. A. (2020). Comprehensive Study of Software Testing: Categories, Levels, Techniques, and Types. TechRxiv. Preprint.