

# Face Detection Performance Using CNNs and Bug Bounty Program (BBP)

Yasmin Makki Mohialden<sup>1</sup>, Saba Abdulbaqi Salman<sup>2\*</sup>, Nadia Mahmood Hussien<sup>3</sup>

<sup>1</sup> Mustansiriyah University, Baghdad, Iraq

<sup>2</sup> Aliraqia University, Baghdad, Ira

<sup>3</sup> Mustansiriyah University, Baghdad, Iraq

\*Corresponding Author: Yasmin Makki Mohialden

DOI: <https://doi.org/10.52866/ijcsm.2024.05.02.006>

Received September 2023; Accepted January 2024; Available online March 2024

## ABSTRACT:

Bug bounty schemes make use of outside ethical hackers to find and fix a variety of security flaws, guaranteeing quicker and more affordable problem solving. Better confidence in and image of the company in the cybersecurity space, faster solving issues, and increased community collaboration are some of its results. Computer vision relies on face detection, which has several uses. This article uses convolutional neural networks (CNNs) and an error reward algorithm in the facial recognition simulation library to enhance face detection. Trainers trained CNNs to detect faces from other visual components and extract human facial traits, making them powerful facial identification tools. These networks classify and extract face characteristics automatically, obtaining approaching 100% identification rates. CNNs have greater identification rates and easier face-image extraction than earlier methods. Network architecture determines its performance, transcending machine learning methodologies. This article suggests a bug reward scheme to discover and resolve bugs in the face recognition library. The program has helped Google find flaws in its intelligent systems, including model manipulation and adversarial assaults. These activities enhance AI safety and security studies, highlight possible concerns, and promote AI safety. CNN-based facial recognition models enhance accuracy and offer advantages over previous approaches. The CNN-based method and Bug Bounty software improved the facial recognition library.

**Keywords:** Bug Bounty program, vulnerabilities, robustness, convolutional neural networks (CNNs), face detection.

## 1. INTRODUCTION

Although bug bounty schemes are not fresh to the field of software creation, more businesses and open source groups are depending on outside contractors to evaluate their software's security in exchange for payment. On the other hand, empirical data regarding the traits of bug bounty program participants is scarce. Programs known as bug bounty's provide an innovative way for companies to crowdsource the safety of software as well as for security experts to get paid appropriately for discovering flaws in software. On the other hand, not much is known about the bug bounty schemes' incentives and how they promote participation and new bug findings. Face identification is essential in computer vision and has various applications. CNNs enhance face recognition, and this work evaluates the fake face recognition library's mistake reward system. Bug bounties reward ethical hackers for reporting application flaws to developers [1], [2]. Facial recognition software vulnerability study [3], [4]. A developer examines and pays ethical hackers who utilize exposure to find software security flaws [1], [5]. Bug Bounty finds and patches software security problems, including facial recognition software, to strengthen it [6].

CNN advancements have accelerated face identification, a significant computer vision technology with many applications [1]. Face recognition applications presume automatic face detection in many visual conditions [3]. Traditional approaches are imprecise and unreliable, thus researchers developed a CNN-based face identification algorithm [6]. A bug bounty scheme finds and fixes library vulnerabilities [6].

Developers compensate ethical hackers for revealing flaws. The CNN-based strategy and Bug Bounty campaign improve library security [6]. Finally, CNN-based models enhanced face identification accuracy and robustness, while Bug Bounty campaigns fixed library bugs.

**The Contribution:** This study introduces CNN-based face identification and evaluates Bug Bounty, a novel facial recognition library vulnerability discovery and patching method. The combination of bug bounty with CNN technology makes this effort more engaging and comprehensive in enhancing computer vision security and performance.

Many applications require good computer vision face identification, yet traditional approaches may be incorrect. This paper solves these vulnerabilities utilizing a bug reward program and CNN-based approach to secure the face recognition library.

**The Outline of the Paper:** Section Two: Literature review. Section Three: Methodology. Section Four: The proposed methodology. Section Five: Conclusion and recommendations for future work.

## 2. RELATED WORKS

The following are some related works:

This article [7] presents A bug bounty case study showing how they may find software security issues. Bug bounty research and corporate security advantages will be assessed. Successful bug bounty schemes that found important vulnerabilities are evaluated qualitatively. Bug bounty hunting appears to find security flaws in gadgets, mobile apps, network protocols, and Internet apps. Skilled security researchers are valued .

In [8], digital platforms implement Bug Bounty Programs (BBPs) to improve software reliability following a rise in security breaches of third-party applications. BBPs help platforms and sellers but may increase prices and impact adoption incentives for suppliers, according to the report. A methodology is presented for evaluating the strategic decisions of platforms and external suppliers during BBP launch and engagement. Security breach loss and vendor investment efficiency are key variables in these choices. The article found that the use of BBP is only balanced when the potential loss is high and the investment efficiency is low. In some settings, BBPs may reduce software stability, platform reliability, and end-user experience, making them socially disadvantageous .

In [9], user confidence in the security of online data is essential to the smooth operation of digital markets and societies, where security is fundamental. Given the complexity and cost of cybersecurity threats, more companies and governments are using bug bounty programs (BBPs) to improve cybersecurity. Hackers are paid by BBP to reveal system vulnerabilities.

In [10], this study evaluates whether a bug bounty program reduces data breaches and how a company's appetite for risk affects this communication. Study reveals that bug bounty schemes encourage data breaches. A bug bounty program reduces data breaches for risk-averse companies, although this benefit is reduced by risk aversion. The study adds to the knowledge on crowdsourcing and cybersecurity and provides practitioners with useful tips.

In[11], the paper covers bug bounty services like Hacker One that outsource vulnerability disclosure to hackers. The paper outlines the costs and benefits for companies and hackers. Running a bug bounty program for a year costs less than hiring two software developers, highlighting its cost-efficiency in resolving vulnerabilities.

## 3. PROPOSED SYSTEM

The general steps of the proposed methodology are:

### 1. Data Preparation:

- Gather a dataset of approved facial images for training.
- Data set preprocessing, including resizing, normalization and data enhancement.

### 2. CNN Model Structure:

- Determine the structure of a convolutional neural network (CNN) suitable for face detection.
- Create layers for convolution, pooling, and fully connected layers.
- Add activation functions, such as ReLU.
- Compile the model using appropriate loss functions and optimizers.

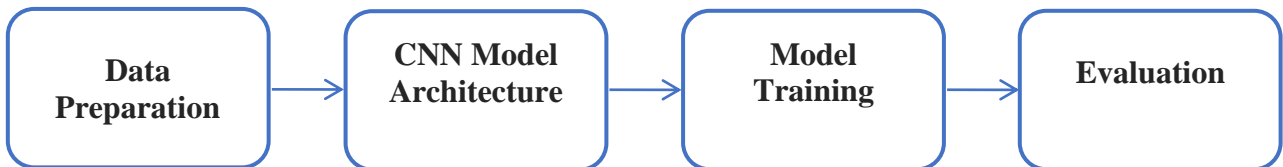
### 3. Model Training:

- Split the data set into training and validation sets.
- Train the CNN model on the training data.
- Monitor training progress and adjust hyper parameters as needed.

#### 4. Evaluation:

- Evaluates the trained model on a separate test dataset.
- Calculate precision, precision, recall, and F1 score for face detection. Rephrase it linguistically and replace it with synonyms

The steps as shown in Figure 1.



**FIGURE 1.** General Diagram for Face Detection

### 3.1 Implementing a Bug Bounty Program in a Face Detection Library Preparation and Implementation Steps:

#### 3.1.1 Setting up the Emulation Library:

- Create a Python library that simulates the face detection function.
- Include known vulnerabilities or issues in the library code.

#### 3.1.2 Error Reporting Mechanism:

- Developed a reporting system that allows users to submit bug reports.
- Include a form for users to explain issues and upload code or sample data.

#### 3.1.3 Bug Tracking:

- Create a bug-tracking system to record and categorize incoming bug reports.
- Assign severity levels to each reported issue.

#### 3.1.4 Bug Fix:

- Develop a process to address reported errors.
- Prioritize errors and fix them based on their severity.
- Update library code with bug fixes.

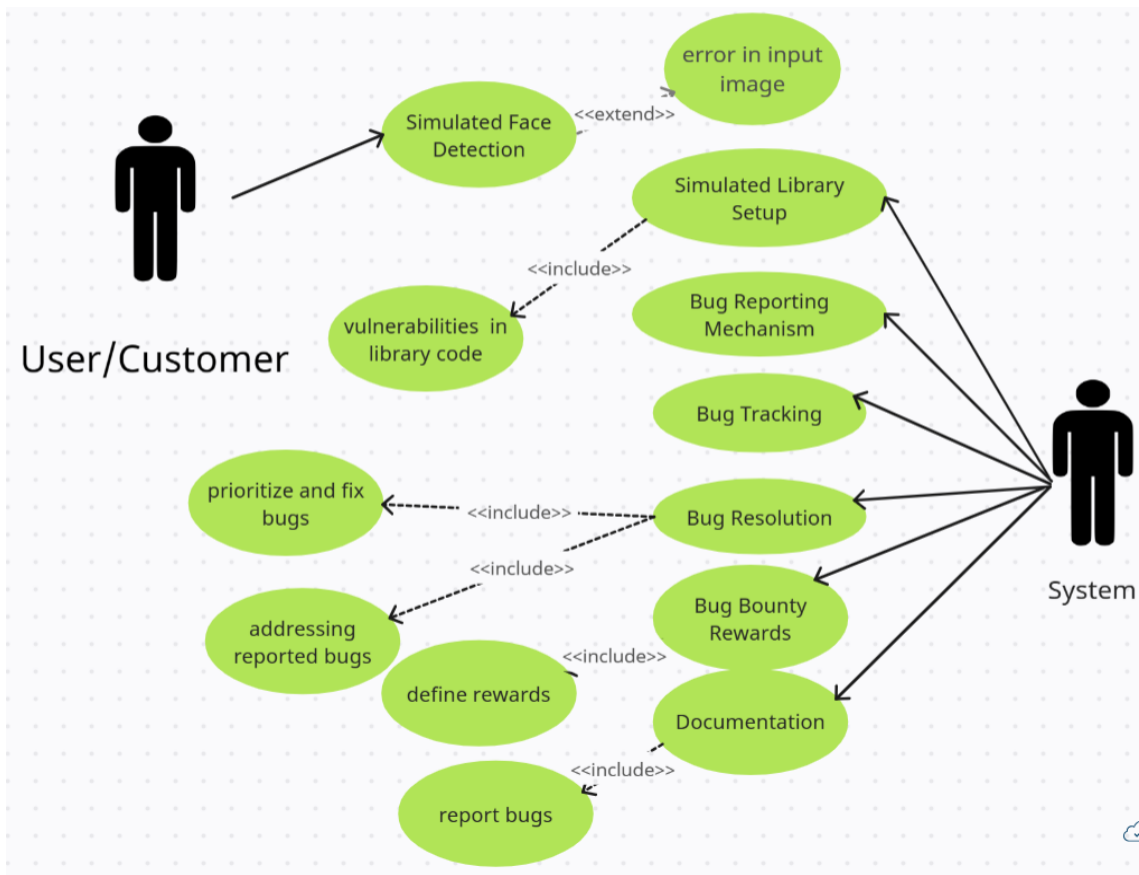
#### 3.1.5 Bug Bounties:

- Set rewards or incentives for users who report valid vulnerabilities.
- Implement a mechanism to distribute rewards to successful informants.

#### 3.1.6 Documentation:

- Provide clear documentation for the library, including how to report bugs and participate in the Bug Bounty program.

The use case of the proposed method is shown in Figure 2



**FIGURE 2. Use Case Diagram for the Proposed Method**

**3.2 Evaluating the effectiveness of a bug bounty program in detecting defects in facial recognition algorithms:**

This approach replicates bug bounty via false facial detection. This study tests a bug bounty program's facial recognition algorithm defect detection. The system contains face detection, quality evaluation metrics, bug bounty simulation, vulnerability assessment, and results presentation simulation libraries.

**3.2.1 Face detection simulation library:**

This custom library simulates facial detection. To simulate real-world facial recognition systems, this library has intended shortcomings. The library correctly and incorrectly recognizes facial positions.

**3.2.2 Quality measurement standards:**

The quality metrics system assesses bug bounty performance. It assesses detection accuracy and completeness using precision and recall. Accuracy is the percentage of faces properly identified, whereas recall is the percentage of faces discovered.

**3.2.3 Simulation Bug Bounty Technology:**

This method employs a face detection library and real faces with known locations. The error reward simulation uses the simulation library to test face detection. Comparing recognized faces to actual faces yields accuracy and recall measurements.

**3.2.4 Vulnerability assessment:**

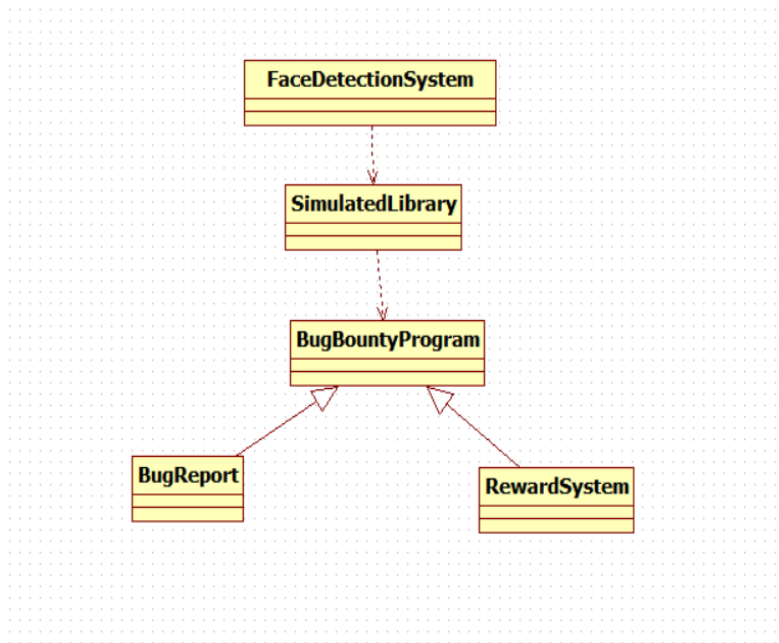
After detection, specific criteria are used to evaluate the vulnerability. Vulnerabilities are identified if the number of recognized faces exceeds a specified minimum, indicating flaws in the detection algorithm.

### 3.2.5 Results:

The results of the bug bounty simulation are presented in the form of text and images. The bug bounty program aims to report precision, recall, and vulnerability. The performance of the algorithm is demonstrated by displaying detection images that show squares around faces.

### 3.2.6 Efforts to reward repetitive errors:

Ongoing efforts are being made to develop a bug bounty program to test the library's performance. Face detection simulation, quality metrics, vulnerability assessment, and results display are included in each iteration. Figure 3 shows the class diagram of the proposed software.



**FIGURE 3.** Class Diagram for the Proposed Program

### 3.3 Pseudo-code Representation of the Described Process

```

# Step 1: Face Detection Library Simulation
class FaceDetectionLibrary:
    method simulate_face_detection(image):
        # Simulate face detection algorithm with intentional flaws
        # Identify true positives and false positives in face detection
        // Implementation details omitted

# Step 2: Error Reporting Mechanism
class ErrorReportingSystem:
    method report_errors(user_description, code_samples):
        # Allow users to describe issues and upload relevant code samples
        // Implementation details omitted

# Step 3: Error Tracking System
class ErrorTrackingSystem:
    variable error_reports

    method __init__():
        error_reports = []

    method track_errors(user_report):
        # Record and categorize error reports based on severity
        // Implementation details omitted

# Step 4: Error Resolution Process
class ErrorResolutionProcess:
    method prioritize_errors():
        # Prioritize errors based on severity levels
        // Implementation details omitted

    method fix_errors():
        # Implement fixes for identified errors in the face detection algorithm
        // Implementation details omitted

# Step 5: Bug Bounty Rewards
class BugBountyRewards:
    method determine_rewards(valid_error_reports):
        # Define a mechanism for distributing rewards to successful error reporters
        // Implementation details omitted

# Step 6: Documentation
class Documentation:
    method create_documentation():
        # Include rules on error reporting and Bug Bounty contribution
        // Implementation details omitted

# Main Program
# Instantiate objects for each step
face_detection_library = FaceDetectionLibrary()
error_reporting_system = ErrorReportingSystem()
error_tracking_system = ErrorTrackingSystem()
error_resolution_process = ErrorResolutionProcess()
bug_bounty_rewards = BugBountyRewards()
documentation = Documentation()

# Simulate the Bug Bounty process
image = load_test_image()
detected_faces = face_detection_library.simulate_face_detection(image)
user_report = error_reporting_system.report_errors("Found issues in face detection", code_samples)
error_tracking_system.track_errors(user_report)
error_resolution_process.prioritize_errors()
error_resolution_process.fix_errors()
bug_bounty_rewards.determine_rewards(valid_error_reports)
documentation.create_documentation()

```


**3.4 Testing environment for bug reward strategies in a face detection simulation library**



This method works well for controlled bug bounty program evaluations. False positives and negatives during detection help evaluate bug reward methods and find security vulnerabilities. Real-world face detection may be analyzed and improved using this method. Frequent and comprehensive evaluations reveal ways to improve bug bounty and program performance in a controlled setting. Table 1 shows these result. Table 2 Illustrates the combined results of face detection and bug bounty simulation

**Table 1.** Simulated Face Detection Library Results

Attempt 1		
Quality Metrics	Precision: 0.00	Recall: 0.00
No vulnerabilities were found.		
Attempt 2		
Quality Metrics	Precision: 1.00	Recall: 1.00
No vulnerabilities were found.		
Attempt 3		
Quality Metrics	Precision: 1.00	Recall: 1.00
Vulnerabilities found!	Could you report them?	
Attempt 4		
Quality Metrics	Precision: 0.00	Recall: 0.00
No vulnerabilities were found.		
Attempt 5		
Quality Metrics	Precision: 1.00	Recall: 1.00
Vulnerabilities found!	Could you report them?	

**Table 2:-** Illustrates the combined results of face detection and bug bounty simulation

Image	Detected 2 faces in the image.	Bug Bounty Report:		
		Bug: False positive in face detection	Bug: Memory leak in face detection algorithm	Bug: Localization issue in error messages
		Status: not a bug	Status: high severity	Status: medium severity
		Reward: \$0	Reward: \$200	Reward: \$100

	<p>Detected 1 faces in the image.</p>	<p>Bug Bounty Report:</p>	<p>Bug: Memory leak in face detection algorithm</p>	<p>Bug: Localization issue in error messages</p>
		<p>Bug: False positive in face detection</p>	<p>Status: high severity</p>	<p>Status: medium severity</p>
		<p>Status: not a bug</p>	<p>Reward: \$200</p>	<p>Reward: \$100</p>
		<p>Reward: \$0</p>		
	<p>Detected 4 faces in the image.</p>	<p>Bug: False positive in face detection</p>	<p>Bug: Memory leak in face detection algorithm</p>	<p>Bug: Localization issue in error messages</p>
		<p>Status: not a bug</p>	<p>Status: high severity</p>	<p>Status: medium severity</p>
		<p>Reward: \$0</p>	<p>Reward: \$200</p>	<p>Reward: \$100</p>

### 3.5 Conclusion and future work

The suggested face detection library simulation method evaluates error reward programs with high accuracy. The efficiency of bug bounty schemes may be assessed using simulation technologies and real-world occurrences. Recall and accuracy assess detection quality and efficiency.

Accuracy is defined as the ratio of positive detections (either true or false) relative to all detected vulnerabilities. The recall metric reflects the percentage of true vulnerabilities, whether detected or not, that lead to true positive outcomes out of all good scenarios. This evaluation adds complexity to measuring the success of a bug bounty program.

Regarding the flexibility of the face detection algorithm, a vulnerability assessment is included to indicate technical issues that improve the security of the system. A bug bounty effort is to frequent evaluation, which makes it easier to measure the success of the program and stabilize the library. Presenting the results of the bug bounty program through graphs and text reports helps understand how it detects flaws in the facial recognition algorithm.

Face detection systems are in demand in social media and security systems, hence it is suggested to enhance them:

**1. Improving the Simulation Environment:** Complex factors like illumination and viewing angles improve simulation accuracy.

**2. Improving Assessment Criteria:** Adding new criteria to better assess bug bounty program performance.

**3. Expanding Security Vulnerabilities:** Analyze and evaluate more security vulnerabilities to improve face detection system security.

**4. Using Machine Learning:** Using machine learning to increase bug reward and detection accuracy.



## Funding

None

## ACKNOWLEDGEMENT

Mustansiriyah University (<https://uomustansiriyah.edu.iq/>) and Al-Iraqia University in Baghdad, Iraq, supported this effort

## CONFLICTS OF INTEREST

The authors declare no conflict of interest.

## REFERENCES

- [1] Yicheng An, Jiafu Wu, Chang Yue "CNNs for Face Detection and Recognition", CS231n, Stanford University / Department of Electrical Engineering Stanford University, 2017.
- [2] Hiba Hameed Ali, Jolan Rokan Naif, Waleed Rasheed Humood A New Smart Home Intruder Detection System Based on Deep Learning <https://mjs.uomustansiriyah.edu.iq/index.php/MJS/article/view/1267>, Volume 34, Issue 2, 2023
- [3] Fean Zhang, Xinyu Fan, Guo Ai, Jianfei Song, Yongquang Qin, Jiahong " Accurate Face Detection for High Performance", AIInnovation Technology Ltd, Beijing, China, 2019.
- [4] Sarab M. Taher, Mustafa Ghanim, Chen Soong Der, Applied Improved Canny Edge Detection for Diagnosis Medical Images of Human Brain Tumors, <https://doi.org/10.23851/mjs.v34i4.1392>, Volume 34, Issue 4, 2023
- [5] Mohammed Haqi Al-Tai, Bashar M. Nema, Ali Al-Sherbaz " Deep Learning for Fake News Detection: Literature Review ", DOI: <http://doi.org/10.23851/mjs.v34i2.1292> Volume 34, Issue 2, 2023.
- [6] Deisy Chaves, Eduardo Fidalgo, Enrique Alegre, Rocío Alaiz-Rodríguez, Francisco Jáñez-Martino, George Azzopardi "Assessment and Estimation of Face Detection Performance Based on Deep Learning for Forensic Applications", published in the 9th International Conference on Imaging for Crime Detection and Prevention (ICDP-19), London, UK, 16–18 December 2019.
- [7] Maulani, I. E., & Anggraeni, R. (2023). Bug Bounty Hunting: A Case Study of Successful Vulnerability Discovery and Disclosure. *Devotion: Journal of Research and Community Service*, 4(8), 1735-1740.
- [8] Tianlu, Z. H. O. U., Ma, D., & NAN, F. (2023). The Use of Bug Bounty Programs for Software Reliability Improvement.
- [9] Wachs, J. (2022). Making markets for information security: the role of online platforms in bug bounty programs. arXiv preprint arXiv:2204.06905.
- [10] Aaltonen, A., & Gao, Y. (2021). Does the Outsider Help? The Impact of Bug Bounty Programs on Data Breaches. The Impact of Bug Bounty Programs on Data Breaches (August 20, 2021). Fox School of Business Research Paper.
- [11] Walshe, T., & Simpson, A. (2020, February). An empirical study of bug bounty programs. In 2020 IEEE 2nd International Workshop on Intelligent Bug Fixing (IBF) (pp. 35-44). IEEE.