Visual Test Automation with Convolutional Neural Networks: Comparative Study of Pre-trained Models for Android Screenshot Validation

Leonardo Tiago
Sidia Institute of Science and Technology
Manaus, Brazil
leonardo.albuquerque@sidia.com

Elza Simoes Bisneta
Sidia Institute of Science and Technology
Manaus, Brazil
elza.bisneta@sidia.com

Mateus Fernandes da Costa
Sidia Institute of Science and Technology
Manaus, Brazil
mateus.fernandes@sidia.com

Ana Paula Silva
Sidia Institute of Science and Technology
Manaus, Brazil
silva.ana@sidia.com

Lennon Correa Chaves
Sidia Institute of Science and Technology
Manaus, Brazil
lennon.chaves@sidia.com

Flávia Camila Oliveira
Sidia Institute of Science and Technology
Manaus, Brazil
flavia.oliveira@sidia.com

Abstract

Context: In a software institute, a test team receives test requests for mobile devices and handles them using automation tools, which generate screenshots as evidence of the success or failure of the test execution, but still need to be visually checked by a human tester, a task that becomes gradually more costly as the amount of test requests is increased. Objective: This work's goal is to propose the automation of the visual check using pre-trained Convolutional Neural Networks (CNNs) with embedding extraction and cosine similarity. Method: Four CNN architectures were employed from the models MobileNetV2, EfficientNetB0, ResNet50V2 and InceptionV3, which were all pre-trained with weights from the ImageNet dataset, then tested with screenshots of 13 different applications in a positive scenario, to check if the images are similar, and in a negative scenario, to check if the images are different. Results: For the positive scenario, InceptionV3 and ResNet50V2 achieved higher similarity rates when compared to EfficientNetB0 and MobileNetV2, although not by much; for the negative scenario, InceptionV3, MobileNetV2 and ResNet50V2 achieved variable results for identifying differences, while EfficientNetB0 achieved the best result by a wide margin. Conclusion: Among the analysed models, the EfficientNetB0 model stood out in regards to performance, achieving similarity rates more consistent with the expected results. For future studies, it is suggested to apply this approach to other contexts of test cases, or to try to improve it including a training step for the models using a dataset built from real screenshots captured by the test team.

Keywords

Android Screenshot Validation, Software Testing, Test Automation, Pre-Trained Models

1 Introduction

Test automation is an essential component in software development, specially for the validation of visual requirements in mobile devices applications [5, 9, 12]. In this context, automation tools execute test scripts and generate visual evidence such as screenshots that display the equivalence of the device's interface with the specified requirements [1]. Techniques such as Visual GUI Testing (VGT) employ image recognition to interact with the system's visual layer, offering flexibility and platform independence [2].

In a software institute, software tests are executed frequently by a test team via automation tools, which take screenshots as visual proof of the implementation of requirements. However, the test process as a whole is only semi-automated, as while the tools execute the scripts and capture evidence [5], it is still necessary to have a human tester analyse the screenshots to determine if the image capture corresponds to the expected result. This dependence makes the process costly, slow and prone to error, and these problems become gradually bigger as the amount of test requests grows and, consequently, the amount of screenshots that need to be manually inspected grows as well with each test cycle. The complete automation of this validation step is, therefore, crucial to enhance the efficiency and productivity of the development cycle.

In this work, we propose an approach that employs pre-trained Convolutional Neural Networks (CNNs) [10] to automate the evaluation of the visual similarity between a set of reference screenshots and a set of screenshots obtained during test execution. When using high level feature extraction techniques (embeddings), the system compares the images' vector representations to determine their similarity level in an automated manner. This methodology is based on transfer learning, applying four distinct CNN architectures (EfficientNetB0 [13], InceptionV3 [15], MobileNetV2 [7], ResNet50V2 [8]) pre-trained on the ImageNet dataset [6] with the goal of identifying which model offers the best performance, *i.e.*, classification effectiveness in the specific context of visual test automation for mobile devices in the test team from the software institute.

The evaluation was performed via an experiment with screenshots from 13 mobile applications in two scenarios: (i) positive scenario, in which the reference and test images match the same screens, and (ii) negative scenario, in which the test images do not match the reference images, but instead display the mobile device's home screen, to simulate the application not being correctly loaded. Cosine similarity was adopted as the main metric, with a value of 0.93 being used as threshold to measure if the images are, in fact, similar. We selected this threshold by design, reflecting a project-driven decision and our confidence in the reliability of the resulting classifications.

We also applied statistical tests to check for significant differences among the models. Specifically, our hypothesis are:

- **Null Hypothesis (H0)**: There is no significant difference between the similarity scores for the models MobileNetV2, EfficientNetB0, ResNet50V2 and InceptionV3 in both positive and negative scenarios.
- **Alternative Hypothesis (H1)**: There is a significant difference between the similarity scores for the models MobileNetV2, EfficientNetB0, ResNet50V2 and InceptionV3 in both positive and negative scenarios.

Therefore, the main contributions of this study are: (i) a systematic comparison of four CNN architectures pre-trained for identifying visual similarity in the context of test automation for mobile devices; and (ii) a practical and reproducible approach to automate visual validation, potentially reducing the need for human intervention and speeding up the software development process.

2 Background

Convolutional Neural Networks have been applied in many works to compare and classify images efficiently. Ahmed [1] demonstrated that pre-trained CNNs, such as ResNet18 and ShuffleNetV2, significantly improve methods based on pixel-by-pixel difference threshold by detecting visual anomalies in rendering screenshots and by learning hierarchical representations that capture everything from edges and textures to high-level semantic patterns. Liu and Lee [11] also employed CNNs to classify screenshots from malicious websites, showing that visual analysis of images can overcome limitations based on source code. The transfer learning technique enhances this approach by using pre-trained weights from big datasets such as ImageNet [6], reducing the need for labeled data from a specific domain and speeding up the model's convergence [1]. The evaluation of image similarity is, therefore, crucial for automating visual tests, specifically in mobile devices, where small variations such as anti-aliasing or color adjustment can affect results and their interpretation as well [1].

3 Test Process

In the software test team in which this study was conducted, tests are performed in mobile devices with the Android operational system [1]. Testers receive test requests which contain the necessary information to setup the device that must be tested and a test cycle composed of the test cases associated to the request. The test cases included in the cycle vary depending on the scope of the test request, and posses detailed instructions of what must be tested and of the expected results for each test [5, 12].

Testers must provide evidence in screenshot format that the tests were successfully finished or that bugs were found during testing. If there are no bugs, the test request receives the "PASS" status, but if there are bugs, the request receives the "FAIL" status and a bug report must be written to describe the bug and the context in which it was found [3, 4].

While the test process is usually performed manually, the test team possesses a test automation tool used to make the process more efficient, as it executes the test cases and collects evidence automatically. However, the level of reliability of these tools, *i.e.*, the correctness of the automated evaluation is not 100%, so it is still necessary to have tester check the evidence collected by the automation tools to determine if the test result was a success or if a bug was found.

To reduce the amount of time and effort needed to check the evidence collected by the automation tool, this work was developed to compare the performance of different neural network models for the task of identifying the similarity level of two distinct screenshots through the cosine similarity metric. To this end, a test case was chosen in which the tester must open a selection of applications embedded on the device and check if the home screen of the application is loaded successfully.

4 Pre-Trained Models for Android Screenshot Validation

In this section we describe the step by step process of this work's methodology, from the choice of images that would be used to test the models, to the selection of candidate models for testing and the selection of metrics to evaluate the performance of these models.

4.1 Selecting Classes

Two classes of images were separated to measure the performance of the models in two different scenarios, images of the positive scenario and images of the negative scenario. In the positive scenario the images display the home screen of the application loaded successfully, while in the negative scenario the images display the home screen of the mobile device, simulating the application not being correctly loaded. 13 screens of 13 different applications were captured to be used in this work.

4.2 Selecting Models

We decided to implement a solution based on CNN models, which are capable of identifying increasingly complex features as the data reaches the models' deepest layers. The models chosen to evaluate this study were:

- **EfficientNetB0**: Possesses an architecture with uniform scaling via neural search. With 5.3 million parameters, its efficiency is balanced, displaying a solid performance with less resources [13].
- **InceptionV3**: Possesses an architecture of Inception modules and convolution factorization. With 42 million parameters, it is able to capture fine details via multi-scale detection [15].
- **MobileNetV2**: Possesses a 53-layer architecture, linear expansion modules and residual connections. With 3.4 million parameters, it is a low complexity model, but of high speeds. [7].
- **ResNet50V2**: Possesses a 50-layer architecture with residual blocks. With 25.6 million parameters, it achieves a high accuracy level even for complex data [8].

We also employed the transfer learning technique using pre-trained weights from the ImageNet dataset, which possesses 1.28 million images across 1000 classes. This way, it was possible to reduce the study's costs, avoiding having to spend time and resources on training the models from scratch, and making them capable

[1] https://www.android.com/

of generalization as the features learned via transfer learning are applicable to the study's scenario, which involves visual tests.

4.3 Selecting Metrics

For this study, two fundamental metrics were analysed:

- **Cosine Similarity**: Cosine similarity measures the angle between two vectors, calculated using Equation 1. Its main advantage is the robustness to variations of lighting and contrast, because pre-trained CNNs models normalize feature vectors and computational efficiency for high-dimensional vectors, such as outputs of intermediate layers of CNNs, which is ideal for semantic comparisons, where the focus is on the structural similarity of objects or patterns, not on pixelated details.

$$CosineSimilarity = (A * B)/(||A|| * ||B||) \quad (1)$$

- **SSIM - Structural Similarity Index**: Structural similarity [14] combines luminance, contrast and structure in sliding windows, calculated using Equation 2. Its main advantage is the higher correlation with human perception, capturing slight differences in textures and edges, while its main challenge is the costly processing for big images due to sliding windows and the sensibility to local changes, which may generate false positive results in automated tests.

$$SSIM(A,B) = (2\mu_A\mu_B + C_1) * (2\sigma_{AB} + C_2)/(\mu_A^2 + \mu_B^2 + C_1) * (\sigma_A^2 + \sigma_B^2 + C_2) \quad (2)$$

Considering these two metrics, Cosine Similarity is preferred for pre-trained CNNs, as the feature vectors already encapsulate semantic abstractions, reducing the need for detailed spatial processing. For more specific contexts, SSIM can be used as a complement for cases in which perceptual precision is critical, but its computational cost must be evaluated.

Therefore, Cosine Similarity was established as the main metric to evaluate the similarity between two images. This metric quantifies the cosine value of the angle between two different vectors in a multidimensional space, varying from -1 being completely different and 1 being identical. The threshold value is 0.93, and if the similarity score exceeds this threshold, the images are classified as similar; otherwise, the images are considered different, as shown in Figure 1.

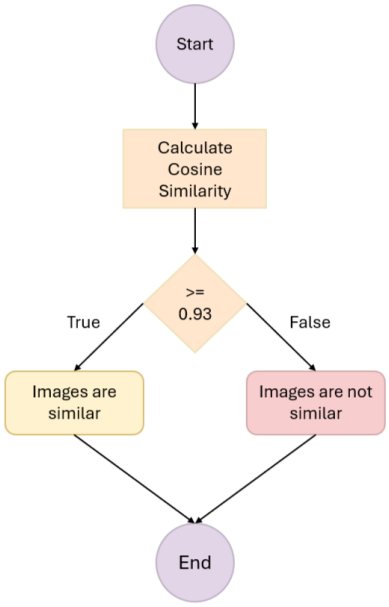

Figure 1: Image Similarity Calculation Process

5 Results

Each model was submitted to the task of validating the evidence collected by the automation tool by comparing them to a set of reference images, and the results obtained for each image were used to calculate the median similarity score for both positive and negative scenarios. This approach was chosen to avoid having the real results being distorted due to outliers, and the specific scores were organized in Table 1 and are also displayed in Figure 2.

Table 1: Experiment results for each CNN model

Model Name	Positive Scenario	Negative Scenario
EfficientNetB0	0.90	0.09
InceptionV3	0.95	0.67
MobileNetV2	0.85	0.38
ResNet50V2	0.97	0.75

The experiment showed that different CNN models had variable performances for each scenario, with similar median scores between them for the positive scenario, but with very different scores for the negative scenario.

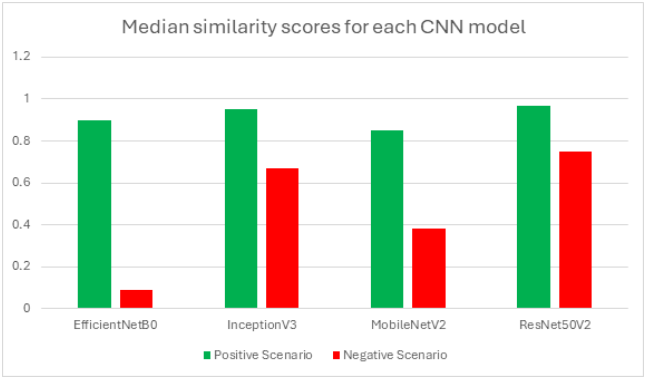

Figure 2: Median similarity scores for each CNN model

In the positive scenario, it is expected that the models achieve high similarity scores (close to 1), indicating a good capacity to determine if the images are similar. For this scenario, the models InceptionV3 and ResNet50V2 achieved the highest similarity scores, exceeding the established threshold of 0.93, while EfficientNetB0's score did not exceed the threshold and MobileNetV2's score was lower than all of the others.

In the negative scenario, a low similarity rate (close to 0) is desirable, indicating a good capacity to find differences among the images. All the models had very distinct results, with InceptionV3, MobileNetV2 and ResNet50V2 achieving high similarity scores, while EfficientNetB0 achieved the similarity score closest to zero among them.

To check which of the hypothesis (Null and Alternative) can be confirmed, we performed the Shapiro-Wilk statistical test to check the normality of the samples' distribution, and it was verified that they do not follow a normal distribution. Thus, we employed the

non-parametric Friedman test considering a significance level of 0.05 to check if there is a significant difference between the models' performance.

The results of the statistical tests indicate that we can refute the Null Hypothesis (H0) and confirm the Alternative Hypothesis (H1), therefore there is a significant difference in the similarity scores among the MobileNetV2, EfficientNetB0, ResNet50V2 and InceptionV3 models for both positive and negative scenarios. This suggests the models have different behaviors when comparing images from both scenarios.

Among the evaluated models, the one who stands out is EfficientNetB0, for its reliability for negative scenarios compared to the other models, its high similarity score for positive scenarios, and its combination of an efficient architecture and consistent performance. Its design optimized for resources makes it an ideal option for tasks that require precision to identify similarities or differences, specially in contexts where performance and scalability are priority.

6 Conclusions

In this study, we discussed the importance of test automation in the context of a software test team from a software institute that performs tests on mobile devices. To optimize the test process, the team employs automation tools that are capable of executing the test cases' steps and automatically collect evidence of tests' results, needing only that the testers check the evidence to determine if the tests were correctly performed and if their expected results were achieved. However, due to the frequency with which the team receives new test requests, the amount of evidence tends to accumulate and the process of individually checking them becomes costly and prone to error.

With the goal of reducing the test team's workload, this study aimed to evaluate the performance of 4 distinct CNN models pretrained on the ImageNet dataset for the task of comparing images, in which they must check if the image captured by the automation tool corresponds to the positive or the negative scenario of the test case of opening all the applications embedded on the device. In the positive scenario, the captured image must correspond to the home screen of the loaded application, and in the negative scenario, the captured image must correspond to the home screen of the device, indicating that the app could not be opened.

Among the evaluated models - MobileNetV2, EfficientNetB0, ResNet50V2 and InceptionV3 - it was noticed that they had varying performances depending on the scenario, with EfficientNetB0 standing out among them for displaying a better performance for identifying differences in negative scenario, despite not having the best result in the positive scenario.

For future studies, it would be interesting to evaluate the application of this approach in a broader context, including more evidence from different test cases instead of focusing on a single one, or to try and improve this approach by training the models with real images captured by the test team so the model under evaluation learns to identify inconsistencies that can help it to determine if the result of a test was a success or a failure.

Artifact Availability

Owing to the confidentiality of the project data and the stipulations of the Non-Disclosure Agreement (NDA), the artifacts and data sets used in this study cannot be made publicly available.

Acknowledgements

This paper is a result of the Research, Development & Innovation Project (ASTRO) performed at Sidia Institute of Science and Technology sponsored by Samsung Eletrônica da Amazônia Ltda., using resources under terms of Federal Law No. 8.387/1991, by having its disclosure and publicity in accordance with art. 39 of Decree No. 10.521/2020. Also we thank GPT-OSS (Open AI) for assisting with language editing, spelling correction, and stylistic improvements. We have reviewed the manuscript, assume full responsibility for its content, and all the authors endorse the final version.

References

[1] Abdel Ahmed. 2025. Machine Learning for Rendering Regression Detection: Enhancing Screenshot-Based Validation for Game Engine QA.

[2] Emil Alégroth, Robert Feldt, and Lisa Ryrholm. 2015. Visual gui testing in practice: challenges, problemsand limitations. *Empirical Software Engineering* 20, 3 (2015), 694–744.

[3] Lennon Chaves, Davi Gonzaga, Leonardo Tiago, Ana Silva, and Flávia Oliveira. 2025. Automatic Generation of Bug Reports Using Large Language Models: An Evaluation in a Software Institute. In *Anais do XXIV Simpósio Brasileiro de Qualidade de Software* (São José dos Campos/SP). SBC, Porto Alegre, RS, Brasil, 337–345. doi:10.5753/sbqs.2025.13599

[4] Lennon Chaves, Flávia Oliveira, and Leonardo Tiago. 2024. Automating issue reporting in software testing: Lessons learned from using the template generator tool. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*. 278–282.

[5] Lennon Chaves, Flavia Oliveira, Leonardo Tiago, and Renata Castro. 2024. Benefits and Challenges of a Physical Device Farm for Automated Software Testing: A Case Study of the STELA Tool Implementation. In *Anais do IX Simpósio Brasileiro de Testes de Software Sistemático e Automatizado* (Curitiba/PR). SBC, Porto Alegre, RS, Brasil, 89–91. doi:10.5753/sast.2024.3880

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[7] Ke Dong, Chengjie Zhou, Yihan Ruan, and Yuzhi Li. 2020. MobileNetV2 model for image classification. In *2020 2nd International Conference on Information Technology and Computer Application (ITCA)*. IEEE, 476–480.

[8] Delvi Hastari, Salsa Winanda, Aditya Rezky Pratama, Nana Nurhaliza, and Ella Silvana Ginting. 2024. Application of convolutional neural network ResNet-50 V2 on image classification of rice plant disease. *Public Research Journal of Engineering, Data Technology and Computer Science* 1, 2 (2024).

[9] Yu-Chung Hsiao, Fedir Zubach, Gilles Baechler, Srinivas Sunkara, Victor Cărbune, Jason Lin, Maria Wang, Yun Zhu, and Jindong Chen. 2025. Screenqa: Large-scale question-answer pairs over mobile app screenshots. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 9427–9452.

[10] Phil Kim. 2017. Convolutional neural network. In *MATLAB deep learning: with machine learning, neural networks and artificial intelligence*. Springer, 121–147.

[11] Dongjie Liu and Jong-Hyouk Lee. 2020. CNN Based Malicious Website Detection by Invalidating Multiple Web Spams. *IEEE Access* 8 (2020), 97258–97266. doi:10.1109/ACCESS.2020.2995157

[12] Flavia Oliveira, Leonardo Tiago, and Lennon Chaves. 2024. Reducing the Allocation of Software Testing Demand: A Study on the Pros and Cons using STELA Tool. In *Anais do IX Simpósio Brasileiro de Testes de Software Sistemático e Automatizado* (Curitiba/PR). SBC, Porto Alegre, RS, Brasil, 77–79. doi:10.5753/sast.2024.3673

[13] Charmy H Patel, Devang Undaviya, Harsh Dave, Sheshang Degadwala, and Dhairya Vyas. 2023. EfficientNetB0 for brain stroke classification on computed tomography scan. In *2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*. IEEE, 713–718.

[14] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

[15] Xiaoling Xia, Cui Xu, and Bing Nan. 2017. Inception-v3 for flower classification. In *2017 2nd international conference on image, vision and computing (ICIVC)*. IEEE, 783–787.