Layout Validation Using Generative AI: A New Approach to Ensuring UI Consistency

Regis Bernard (Author), Shripad Deshpande (Author), Sapan Tiwari (Co-Author), Dharmam Buch (Co-Author), Himanshu Pathak (Co-Author)
RegisRozario@gmail.com, Shripad.Deshpande18@gmail.com, SapanSanu@gmail.com, BuchDharmam@gmail.com, Himanshu.Pathak086@gmail.com

Abstract

In modern software development, ensuring visual alignment and layout accuracy across multiple platforms presents significant challenges that are not fully addressed by traditional testing methods, such as end-to-end (E2E) tests. While E2E tests focus on functional verification, they fail to identify layout-related issues, such as misplacement of UI elements, misalignments, or overlaps. This paper introduces a novel approach for automated layout validation using generative AI. The proposed method compares the rendered UI against a baseline design system, which can be any design tool like Figma, Sketch, or Adobe XD. The process involves capturing screenshots of the rendered page, comparing these screenshots with the design specifications, detecting discrepancies, and generating reports for further review. The solution is designed to work across mobile, tablet, and web platforms, ensuring consistent design implementation and providing a more effective approach to cross-platform layout validation.

Our experiments show that the generative AI–driven Siamese Network approach achieved a similarity score–based detection accuracy of over 95%, outperforming baseline methods such as pixel-difference, CLIP embeddings, and DOM-based diffs. This demonstrates its robustness against rendering noise, responsiveness to layout shifts, and reliability in detecting UI inconsistencies without requiring pixel-perfect matching.

Biography

Regis Bernard is a Software Development Engineer in Test at a large social media company, focusing on automation frameworks and UI validation through AI.

Shripad Deshpande is a Senior QA Engineering Manager at a large social media company, leading GenAI QA efforts. He is an AI for QA enthusiast, with a strong focus on efficiency.

1. Introduction

The importance of maintaining design consistency across various platforms and screen sizes is central to the user experience (UX) in modern software development. While traditional testing methods such as end-to-end (E2E) tests are effective at validating the functionality of software, they often fail to address visual design issues such as the alignment, proportion, and placement of UI elements. As user interfaces become more complex, it is critical to ensure that these elements display correctly on all devices mobile, tablet, and web.

This paper presents an automated layout validation system that utilizes generative AI to compare rendered UIs with their corresponding design systems. Unlike E2E tests, this approach focuses on verifying the visual fidelity of the user interface. By automating the process of identifying and reporting discrepancies, this solution aims to streamline the validation process, reducing the risk of UI-related issues slipping through the cracks during development.

2. Background

Ensuring UI consistency across devices has long been a challenge in software development. Traditional functional tests validate that components work as intended but often overlook layout-specific issues like misalignments, inconsistent spacing, and overlapping elements. These problems, while seemingly minor, can significantly degrade user experience and brand perception. This section first reviews prior approaches to layout validation and their limitations, and then highlights notable real-world failures where poor layout design had measurable negative consequences.

2.1 Related Work

Traditionally, UI testing has been focused on ensuring the functionality of an application, verifying that buttons, links, and forms operate as expected. However, these tests do not account for layout issues such as misaligned elements, inconsistent padding, or elements overlapping one another. These issues, while visually apparent, are often difficult to detect using traditional functional testing approaches.

Several solutions have been proposed to address visual testing, including image-based comparison tools. These tools compare screenshots of the rendered page to predefined reference images, flagging any discrepancies. However, these solutions are often limited to rigid one-to-one comparisons and may not adequately handle dynamic content or responsive design, where layouts change depending on the device or screen resolution.

Generative AI offers an alternative approach. By analyzing and comparing images intelligently, generative AI can detect layout issues even when there are small variations between the rendered UI and the design. This enables a more flexible and accurate comparison, particularly in complex or responsive designs.

3. Methodology: Layout Validation Using Generative AI

The approach proposed in this paper involves the use of generative AI to automate the process of layout validation by comparing rendered UI elements with a reference design system. The steps in this process are as follows:

- **Step 1: Page Load and Screenshot Capture**
  The first step involves loading the rendered page on the target device (e.g., mobile, tablet, or web) and capturing a screenshot of the UI. This screenshot represents the actual layout that needs to be compared against the design system.

- **Step 2: Design System Comparison**
  The captured screenshot is then compared to the design system, which may be a design file from Figma, Sketch, or Adobe XD. AI-powered image recognition algorithms are employed to identify the positions, sizes, and alignment of key UI elements, comparing these against the design specifications.

- **Step 3: Discrepancy Detection**
  Any discrepancies, such as misaligned elements, incorrect button sizes, or overlapping UI components, are detected **using our ResNet-based Siamese Neural Network model**. The model compares layout features extracted from the rendered screenshot and the reference design, learning to recognize differences in spatial arrangement rather than relying on raw pixel matching. When the similarity score falls below the predefined threshold, the system flags the layout as inconsistent. These discrepancies are then highlighted, and the system generates a marked-up version of the screenshot for review.

- **Step 4: Reporting and Analysis**
  The marked screenshot is forwarded to tools for further analysis, such as Vision Pro or ChatGPT, which can provide additional insights into potential issues or resolutions. This step allows designers and developers to quickly address identified discrepancies and iterate on the design.

4. Initial Approaches and Limitations

The approaches described below were our early experiments for **Step 3: Discrepancy Detection** in the methodology. At this stage, the goal was to automatically identify misalignments, overlaps, and other layout regressions by comparing rendered UI screenshots against baseline designs. We evaluated several techniques ranging from pixel-level comparisons to embedding-based methods. However, each had specific limitations that made them unsuitable for robust cross-platform layout validation, as summarized in the table below.

Approach	**What We Tried**	**Why It Didn't Work Well**
Pixel Difference	Used OpenCV cv2.absdiff, SSIM	Too sensitive to rendering noise and minor shifts
CLIP + Cosine Similarity (pre-trained)	Used CLIP embeddings with similarity	Could not capture layout semantics or overlaps
DINOv2 Embeddings (pre-trained)	Used timm pretrained DINOv2 ViT	Better than CLIP, but lacked layout specificity

DOM-based Diff	Compared DOM trees and structure	Not image-based, failed on native/canvas UIs

Pixel Difference

We first experimented with traditional image-difference techniques using OpenCV's cv2.absdiff and SSIM. While these methods are straightforward, they proved far too sensitive to minor rendering noise such as anti-aliasing, font smoothing, or small pixel-level shifts. As a result, they generated a large number of false positives, making them unsuitable for dynamic and responsive layouts.

CLIP + Cosine Similarity

CLIP + cosine similarity is great for **semantic similarity** (e.g., "does this page contain a login button?"), but **not reliable for layout verification** where **spatial fidelity** is critical. That's why we moved to **Siamese networks**, which learn **pairwise differences in spatial structure**, making them far better for detecting misalignments, overlaps, or missing elements.

DINOv2 Embeddings (pre-trained):

DINOv2 embeddings capture global context and object categories.Great for tasks like image retrieval or clustering ("this is a cat, this is a dog"), but they are not trained to care about alignment or spacing. It recognizes "what the screen is" but not "how the elements are arranged." For layout regression, we need models (like a Siamese network) that explicitly learn pairwise differences in spatial arrangement rather than just global semantic similarity**.**

DOM-based Diff

We also explored DOM tree comparisons to detect structural differences in rendered layouts. While effective for static, HTML-based UIs, this approach fell short in cases where rendering was handled by native mobile views or canvas-based graphics. Furthermore, DOM-based diffs do not account for visual alignment, spacing, or overlap issues, limiting their applicability in modern cross-platform applications.

Why CLIP & VIT based model response for layout regression doesn't work:

This figure provides a visual comparison between a baseline UI (left) and a regressed UI (right). In this example, the regressed version introduces an overlap in text ("sdsds"), clearly violating the layout consistency of the baseline design. Such discrepancies are supposed to be flagged by the CLIP & VIT models but due to the nature of the model these discrepancies are often overlooked.

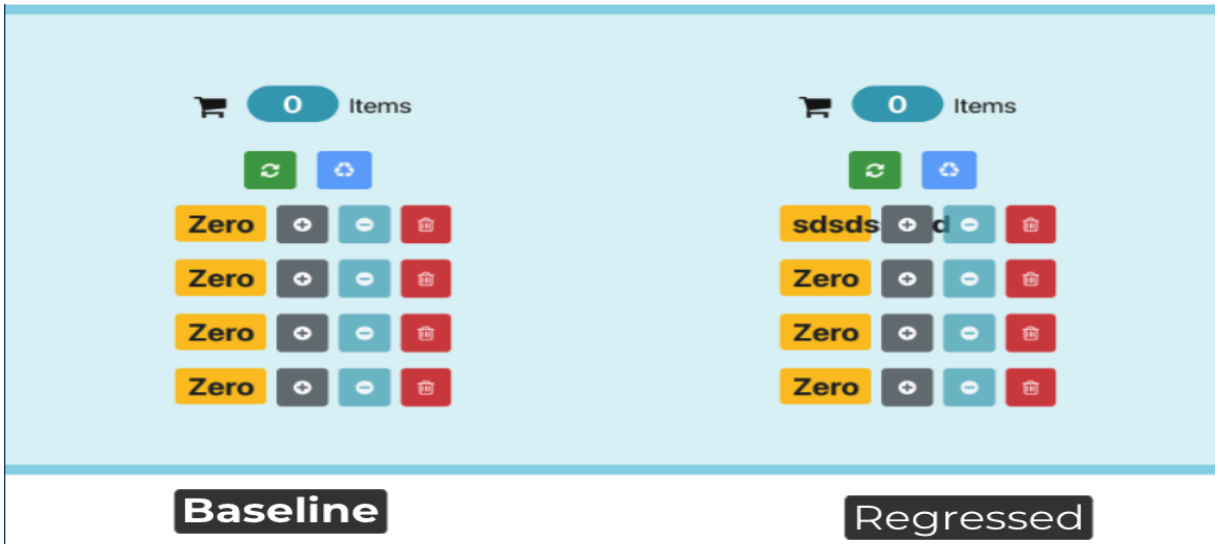

Figure 1. Example of baseline versus regressed UI and where CLIP & VIT based models unable to identify the layout issues.

This figure shows an example of the structured output produced by the system after a layout comparison. It includes the baseline and candidate screenshots, similarity score, regression detection flag, threshold value, and additional metadata such as pixel-level differences and reasoning for the result. This machine-readable format enables easy integration into CI/CD pipelines and automated reporting workflows


```
{
  "name": "counter_app_text_overlap",
  "image_comparison_result": {
    "baseline": "baseline/counter_app_text_overlap.png",
    "candidate": "candidate/counter_app_text_overlap.png",
    "similarity": 0.9995681047439575,
    "regression_detected": false,
    "threshold": 0.99
  },
  "layout_issues": "pixel_diff: 0.05%",
  "regression_reason": "no significant layout deviation"
},
```


Figure 2. JSON output of layout comparison results

5. Proposed: Siamese Neural Network

We tried a ResNet-based Siamese Network to learn visual distances between UI image pairs.

This figure illustrates the complete workflow of the proposed system.

- **Test Execution Phase:** Baseline (expected) and current (test) screenshots are captured and pre-processed through resizing, cropping, and normalization to ensure consistency before model input.

- **Siamese Neural Network:** Both baseline and current screenshots pass through ResNet-based feature extractors to generate layout features. Differences between the two feature sets are computed, and a regression head produces a similarity score between 0 and 1.
    - Apply the same resize (244 px) /normalization to the incoming current screenshot.
    - Look up the cached embedding based on baseline file name and retrieve the precomputed baseline embedding otherwise, encode the baseline image on the fly.
    - Compute the current screenshot with the same backbone.
    - Compute $d=|fb-fc|$, MLP → logit → probability $p=\sigma(\text{logit})$.
    - If score< Threshold, flag layout regression; else no regression.
    - Emit a visual diff artifact (e.g., pixel diff or Grad-CAM/activation map overlay) to aid debugging.

- **Training Data Pipeline:** Screenshots from good and broken UIs are collected and labeled, then used to train or fine-tune the Siamese network. This enables the system to continuously improve at detecting subtle layout regressions.

- **Decision Logic:** The similarity score is compared against a threshold. Layouts above the threshold are marked as passed, while those below are flagged as failed.

- **Reporting and Alerts:** Results are summarized into developer-friendly reports with similarity scores, heatmaps, and marked-up screenshots. Alerts integrate with CI/CD systems, automatically preventing regressions from being deployed and notifying teams via established channels.

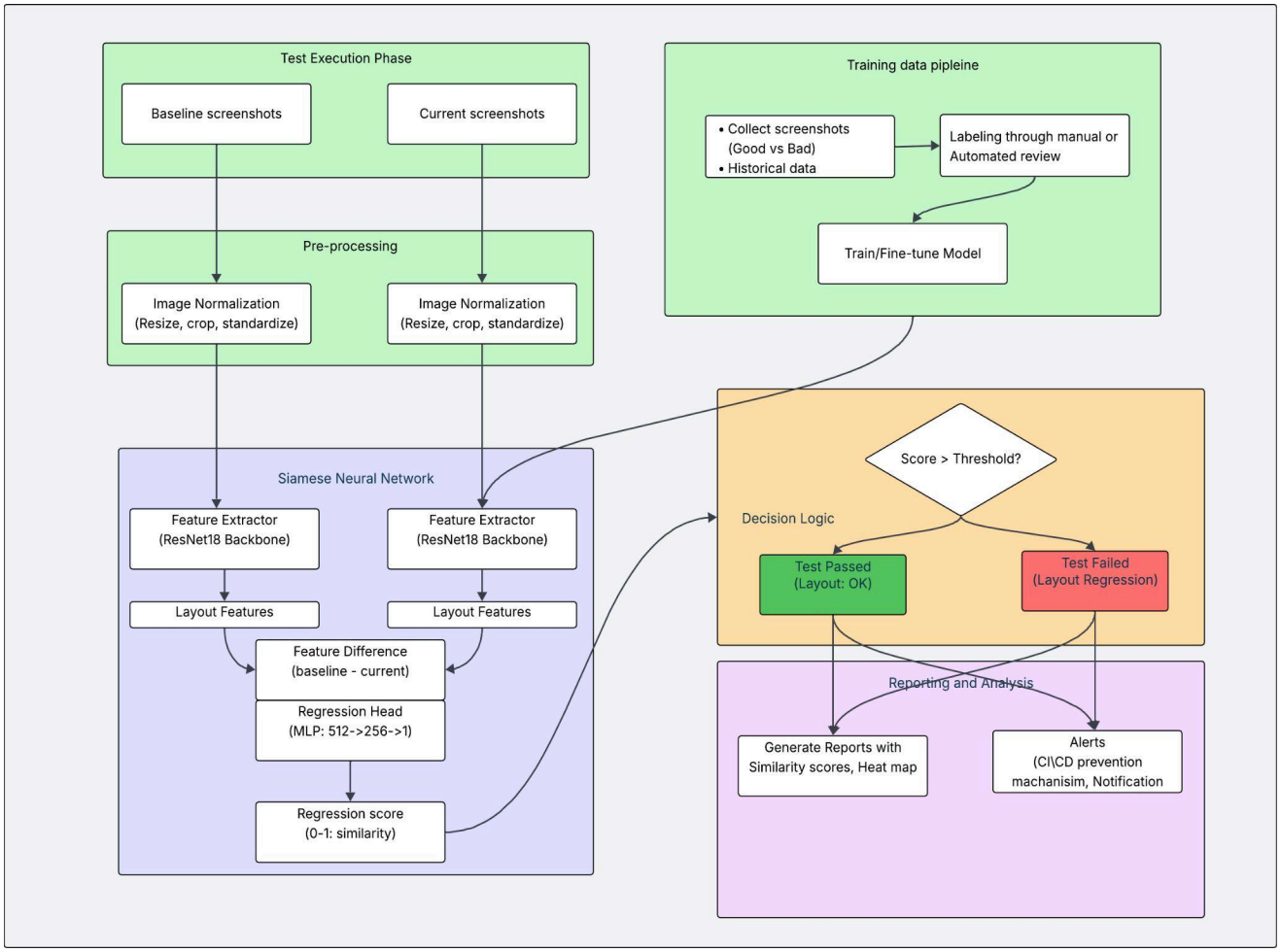

Figure 3. End-to-end workflow for layout validation using a Siamese Neural Network

With Siamese Neural Network output:

This is the output image generated based on baseline and current screenshot differences. With the Siamese Neural network model, we were able to identify the subtle layout mismatch as well.

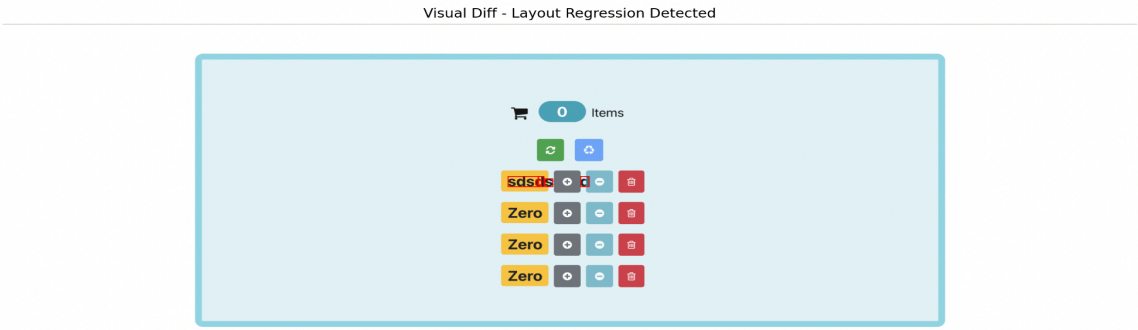

Figure 4. Output for layout validation using a Siamese Neural Network

6. Results and Observations

Using a pixel-diff baseline, **83%** of flagged issues were noise/false positives from antialiasing, sub-pixel shifts, and font smoothing. We therefore adopted a Siamese CNN that compares **feature embeddings** rather than raw pixels, cutting noise to **32%** (a **51-point, ≈61%** relative reduction) while improving robustness across devices and renderers. The model stays sensitive to true layout regressions, **shifts, missing components, overlaps** and outperforms generic **CLIP/ViT** semantic encoders for fine-grained layout verification, reducing false negatives. Because it requires neither pixel-perfect matching nor **DOM** access, it works for native, canvas/WebGL, and responsive UIs, and it produces actionable outputs, **heatmaps, marked-up screenshots, and concise explanations** that speed triage and help teams prioritize fixes.

6.1 Limitations

Although effective, the proposed approach has certain limitations. Its accuracy depends on the quality and variety of training data, and highly dynamic or personalized layouts may still pose challenges. The similarity threshold requires fine-tuning to balance false positives and negatives, and the method introduces higher computational cost compared to traditional techniques. Additionally, while large language models (LLMs) can also be used by directly passing screenshots for analysis, this approach is not cost-effective since LLMs charge based on tokens. In contrast, our method reduces the token usage by approximately four times by only sending summarized results to LLMs for analysis, making it significantly more efficient. Finally, while strong in detecting spatial inconsistencies, the system does not fully capture higher-level design intent such as brand or accessibility guidelines.

7. Future Work and Conclusion

The approach presented in this paper offers a promising solution for automated layout validation. Future work will focus on improving the accuracy of AI comparisons, particularly for complex and dynamic layouts. Additionally, integrating this system into continuous integration/continuous deployment (CI/CD) pipelines will allow for real-time validation during the development process, further enhancing its utility.

In conclusion, generative AI provides an efficient and effective solution for ensuring design consistency across platforms. By automating layout validation, this approach allows teams to focus on other critical tasks while ensuring that the visual integrity of the product is maintained.

References

Long, M. M. J. 2013. Ford's MyFord Touch: A Case Study of Design Failures. Design Management Review.

Morville, M. 2016. User Experience Design: Principles and Practices.

Norman, D. A. 2013. The Design of Everyday Things.

O'Reilly, K. A. 2014. "What Went Wrong with Healthcare.gov." Harvard Business Review.