Clustering Websites by Salient Design Features

Thisaranie Kaluarachchi *, Sumedhe Dissanayake, and Manjusri Wickramasinghe
School of Computing, University of Colombo, Colombo, Sri Lanka
Email: thisaranie@spc.cmb.ac.lk (T.K.); sumedhedissanayake@gmail.com (S.D.); mie@ucsc.cmb.ac.lk (M.W.)
*Corresponding author

***Abstract*—Designing websites to meet user and client expectations often requires repeated refinement cycles, making the process time-consuming and resource-intensive. This study proposes an automated classification system that categorizes real-world websites based on their salient structural design features to support data-driven automatic website generation. The system integrates Self-Organizing Maps (SOMs) with a novel image-processing pipeline that combines edge detection, gradient analysis, and morphological filtering and image smoothing to extract structural wireframe layouts from website screen captures. Experiments were conducted on three datasets: manually created wireframes, screen captures of top 100 websites, and screen captures of top 1500 websites ranked by SimilarWeb. The analysis revealed seven representative layout archetypes: dashboard interfaces, simple information pages, fixed-width product grids, informational pages with sidebars, basic search interfaces, multi-section content layouts, and tabular data interfaces. The classification quality was evaluated using topographic error, quantization error, Silhouette coefficient, and Davies-Bouldin index, demonstrating consistent and meaningful clustering. Our findings highlight the potential of SOM-based clustering for automatic website template generation, offering a scalable and data-driven foundation for design automation and front-end prototyping.**

***Keywords*—website structure analysis, automatic website generation, web design clustering, salient design features, self-organizing maps**

I. Introduction

Modern websites serve as central interfaces for information access, communication, commerce, and user interaction across digital platforms. Web development involves designing, constructing, and maintaining websites and web applications, while web design specifically focuses on crafting user interfaces that are visually appealing, intuitive, and accessible. During the design phase, designers conceptualize and implement layouts that balance aesthetic value with functional usability to ensure a positive user experience. The visual presentation, organization, and interactivity of a website depend largely on a designer's understanding of User Interface (UI) and User Experience (UX) principles. By effectively applying design strategies such as alignment, spacing, and hierarchy, designers create digital environments where users can easily locate and engage with content. Web designers create websites using a variety of methods: coding them from scratch, using a Content Management System (CMS), using website builders, or using frameworks, depending on their preferences and clients' needs.

A website user interface is the place where humans and machines interact. Through a web browser, users interact with websites or web applications by means of their visual and interactive features. While UX focuses on the entire feel and experience of using a website, UI is concerned with the appearance and functionality of the website's interfaces. The UI design of a website plays a crucial role in the user experience since it influences how easily users can discover information, accomplish tasks, and achieve their goals on the website. An intuitive and thoughtfully designed interface enhances user navigation, draws attention to essential information, and supports users in accomplishing their tasks more effectively. In the context of Software-as-a-Service (SaaS) businesses, where the digital platform often serves as the primary point of contact with customers, a functional and visually appealing UI can significantly influence user satisfaction and overall business outcomes. According to Garett *et al.* [1], seven core elements contribute to user engagement in website design: navigation, visual hierarchy, structural organization, content relevance, clarity of purpose, simplicity, and readability. Web designers can apply these elements to craft engaging interfaces by following key principles such as maintaining a clean and straightforward UI, ensuring visual consistency with familiar components, and using colour and texture purposefully.

The visual appeal and accessibility of a website are crucial in forming users' initial impressions and shaping how they perceive a brand. To support this, web designers employ various techniques to make interactive elements such as animations, images, icons, and logos intuitive and user-friendly. They also ensure that visual elements such as symmetry, colour schemes, and layout proportions align with the client's branding and marketing goals. However, the initial design may not always be completely aligned with the client's expectations, and multiple revisions are often required before final approval is achieved. This traditional Graphical User Interface (GUI) creation process requires significant human effort, making it both time-intensive and resource-intensive. In response to these

Manuscript received September 18, 2025; revised October 10, 2025; accepted November 10, 2025; published 26, 2026.

challenges and the growing interest in web design automation, Kaluarachchi and Wickramasinghe [2] have proposed automatic website generation approaches. The idea focuses on optimizing the design workflow by automatically generating layout structures and interfaces based on parameters specified by the user. Through automation, designers can cut down on repetitive tasks, reduce development expenses, and limit human errors that often occur in manual coding and layout construction [3–5].

Moreover, some existing websites encapsulate the best practices that web designers used through different epochs of the internet. This knowledge can be used when creating new websites. Therefore, an opportunity exists to reduce costs in the website creation process by extracting knowledge from existing websites and using this knowledge as input to generate mock-ups. Various strategies can be employed to bridge this knowledge gap. In one such approach, real-world websites are analyzed, classified, and converted into code that can be customized using certain algorithms. The first step is to classify real-world websites into categories based on their design features and then derive new websites based on categories. This idea can be further extended to automating the website design process to reduce costs and human errors. This study focuses on classification of real-world websites.

Image feature classification is a fundamental task in computer vision and pattern recognition, where the goal is to categorize images based on their visual features. Traditional classification methods rely on supervised learning approaches, but unsupervised learning techniques, such as Self-Organizing Maps (SOMs), have gained significant attention due to their ability to map high-dimensional data into lower-dimensional representations while preserving topological relationships. The SOM, introduced by Kohonen [6], is a type of artificial neural network that performs unsupervised learning to organize input data based on similarities. SOMs operated by mapping input vectors onto a two-dimensional grid, are similar feature representations cluster together. This makes SOMs particularly useful for tasks such as feature extraction, dimensionality reduction, and image classification.

In the context of image feature classification, SOMs play a crucial role in identifying patterns and structures within image datasets. By learning the intrinsic distribution of features, SOMs can group images with similar characteristics, making them effective for various applications, including medical imaging, remote sensing, facial recognition, and object detection. This approach parallels traditional image-based pattern recognition tasks such as object detection and document segmentation, where feature extraction and clustering play a key role in classification. By leveraging SOMs in the classification of website structures, we extend their application to a novel domain, categorizing web designs based on structural attributes derived from wireframes. A notable application of SOMs in clustering visual data can be seen in clustering websites, where websites can be categorized based on their design features. This demonstrates the versatility of SOMs in various domains beyond image classification.

In this study, we introduce an unsupervised SOM-based website classification framework that categorizes real-world websites based on their salient structural features extracted from screen captures. Unlike conventional SOM implementations, the proposed method integrates edge detection, morphological operations, and gradient-based filtering to emphasize wireframe geometry before training, improving structural fidelity and cluster cohesion. This hybrid pre-processing pipeline enhances the SOM's ability to capture fine-grained design variations, yielding more accurate and interpretable structural clusters. The resulting clusters represent distinct design archetypes, which can serve as the foundation for automatic web template generation and design pattern discovery. Accordingly, this research addresses the following key questions:

- RQ1: How can real-world website designs be grouped based on structural similarity?
- RQ2: What visual features are most indicative of underlying layout patterns?
- RQ3: How does pre-processing influence the effectiveness and interpretability of SOM-based clustering?

This work makes the following key contributions:

- It proposes a structure-oriented classification pipeline that clusters real-world websites based on wireframe-level design features using SOMs.
- It introduces a hybrid pre-processing framework combining edge detection, morphological filtering, and gradient-based enhancement to improve clustering quality;
- It provides quantitative and visual analyses of the resulting clusters, identifying seven recurring website layout archetypes that can inform automated design and template generation systems.

The remainder of this paper is organized as follows: Section II reviews related literature on automatic GUI generation, SOM applications, and web page classification. Section III describes the design framework and implementation details of the proposed website classification system. Section IV presents the experimental results and comparative analysis. Section V discusses the study's limitations and outlines future research directions. Finally, Section VI concludes the paper by summarizing the findings and outlining their broader implications for automated web design.

II. Literature Review

Research on automatic website generation and design analysis has evolved rapidly with advances in computer vision and machine learning. To clarify the technical landscape, this section reviews prior studies across three interrelated domains: (A) automatic GUI generation, (B) image feature extraction using SOMs, and (C) web page classification. Each domain contributes partial solutions to automating aspects of interface design, but none directly address the large-scale clustering of real-world websites by structural layout, which is the focus of this study.

A. Automatic Graphical User Interface (GUI) Generation

The automation of GUI generation has become a crucial research area, aiming to streamline UI design and development while minimizing manual effort. Traditional GUI development is labour-intensive, requiring significant human intervention, which increases time and resource costs. To address these challenges, researchers have explored machine learning, computer vision, and pattern recognition techniques to automate design pattern extraction, structured layout generation, and the conversion of sketches or screenshots into functional UI code. Similarly, website classification can be effective in identifying common design patterns that can inform automatic web generation. By categorizing websites based on their structural and visual elements, these classification systems facilitate the development of template-based UI generation models, minimizing the iterative design process. The following section reviews recent research studies on automatic GUI generation, highlighting key advancements and methodologies in the field.

Kaluarachchi and Wickramasinghe [4] proposed WebDraw, a machine learning-driven framework for automatic website prototyping that translates high-fidelity mock-up artifacts, such as screen captures of real websites, into functional HTML/CSS code. The system follows a three-phase pipeline:

(1) GUI element detection using image processing to identify atomic web elements,
(2) Convolutional Neural Network (CNN)-based classification of these elements into domain-specific categories (e.g., headings, images, or buttons),
(3) recursive Document Object Model (DOM) hierarchy construction to generate a functional code structure.

The resulting prototypes maintain strong visual and structural similarity to the input websites, achieving an average of 90% classification accuracy. Distinctively, WebDraw demonstrates that real-world web screen captures, rather than hand-drawn or synthetic sketches, can serve as effective mock-up artifacts for automatic code generation. In contrast to WebDraw's supervised, code-oriented generation pipeline, our study adopts an unsupervised clustering approach to discover structural regularities in website wireframes. Rather than translating individual examples into HTML/CSS code, our method generalizes across large-scale datasets to identify recurrent design patterns that can later inform systems like *WebDraw* in automating layout generation.

Natarajan and Csallner [7] presented Pixels to Animated (P2A) UI, a system designed to automate the generation of animated mobile application interfaces from static screen designs. The tool uses a combination of computer vision, Optical Character Recognition (OCR), and perceptual hashing to extract UI components and infer transitions between screens and in-screen animations. Unlike earlier systems such as REMAUI, which primarily reconstruct static layouts, P2A advances UI prototyping by incorporating animation behaviour [8]. Evaluated on a set of top-rated Android applications, P2A showed strong performance in replicating visual and temporal aspects of mobile UIs. However, its capabilities are currently limited to predefined animations and non-interactive elements, making it less suitable for complex dynamic interfaces. In the context of website design analysis, P2A highlights how visual inference from screen captures can inform interface reconstruction. While P2A focuses on animation extraction, our work diverges by targeting the structural clustering of web page layouts using unsupervised learning, with the aim of identifying high-level design archetypes across large website collections.

Beltramelli [9] introduced pix2code, an early deep learning framework that translates GUI screen captures into source code, aiming to reduce the manual effort required in front-end development. The system processes screen captures from IOS, Android, and web platforms and outputs a Domain-Specific Language (DSL) that describes the layout and components of the interface. This DSL is then compiled into platform-specific code. The architecture combines three neural networks: one for interpreting the visual content of the UI and two for generating corresponding DSL sequences. While pix2code marked an important step toward automating GUI generation, its scope is limited to single-screen applications and basic layouts. In contrast to pix2code's supervised code generation from individual UI images, our work focuses on unsupervised clustering of structural design features across many website wireframes, offering a layout-centric perspective that could support template-driven automatic website generation without the need for manually labelled training data.

Kumar [10] developed SketchCode, a neural network-based system that translates hand-drawn website wireframes into HTML code. Unlike earlier models that relied on digital mock-ups or screenshots, SketchCode specifically targets the variability and imperfections of sketch-based inputs, allowing developers and designers to create functional prototypes directly from freehand drawings. The model processes input sketches and maps them to structured HTML layouts, bridging the gap between early-stage ideation and web development. However, the system is sensitive to inconsistencies in sketching style and layout structure, which can result in significant variability in the generated websites. While SketchCode focuses on semantic translation of individual UI sketches, our research shifts attention toward automatically clustering layout structures in large-scale website screen captures. This structural classification offers a scalable path toward template generation, independent of user-drawn input, by identifying recurring layout patterns in real-world web designs.

Xu *et al.* [11] introduced image2emmet, a system designed to assist front-end web developers by automating the translation of GUI images into structured HTML/CSS code. The tool employs computer vision techniques to detect and interpret graphical UI components within static web interface images, converting them into Emmet-style

HTML/CSS representations. Evaluated on a synthetic dataset, image2emmet achieved an 80% detection accuracy for GUI elements and a 60% accuracy in generating valid HTML/CSS markup. While it supports rapid prototyping and code scaffolding from visuals, its performance depends on input quality and layout regularity. Compared to this direct code-generation approach, our research focuses on unsupervised layout clustering from real-world website screen captures. Rather than translating individual interfaces into code, our method extracts design structure patterns at scale, enabling data-driven support for template-based web generation and UI design analysis.

Kim *et al.* [12] proposed a deep learning-driven pipeline for automating the generation of functional HTML code from hand-drawn website sketches. Their method begins with CV techniques to extract UI components from the sketch, which are then segmented and passed through a CNN that classifies each element (e.g., buttons, text fields, containers). An HTML code generation module, based on Bootstrap, then constructs a fully functional web interface based on the classified components. The system achieved 96% accuracy in component classification on a dataset of 186 sketches. While their work focuses on translating manually created low-fidelity prototypes into code, our approach centres on analysing and clustering real-world, high-fidelity screen captures using unsupervised learning. Instead of direct code generation, we offer a structural classification of website layouts, which can serve as a foundation for template creation and interface design automation.

Beltramelli [13] presented Uizard, a rapid prototyping platform that leverages deep learning and CV to convert hand-drawn wireframes into interactive, high-fidelity UI prototypes. The system identifies GUI components from sketches and maps them to functional interface elements, which can be customized using built-in style guides. Uizard's output supports multiple export formats including HTML/CSS, React, and Android code, facilitating seamless integration into development pipelines. In usability studies involving freelance designers and front-end developers, Uizard demonstrated a 24× improvement in prototyping speed, highlighting its effectiveness in accelerating early design stages. While Uizard focuses on quickly transforming individual sketches into production-ready UIs, our work contributes to the broader challenge of structural UI pattern discovery. By clustering real-world website designs using wireframe representations, our method enables template-level understanding and supports scalable, data-driven UI design automation.

Ferreira *et al.* [14] introduced a system for automatically generating responsive websites from hand-drawn mock-ups, addressing the inefficiencies in early-stage UI prototyping. Their method, called Pix2Pix, involves a four-step pipeline: image acquisition, UI element detection, container detection, and hierarchical reconstruction. The pipeline begins by enhancing sketch input via adaptive thresholding and morphological filtering. Then, You Only Look Once (YOLO) (https://pjreddie.com/darknet/yolo) is used to detect atomic UI components, while a conditional Generative Adversarial Network (cGAN) (https://github.com/phillipi/pix2pix) generates element containers, which are also identified using YOLO. The system builds the layout hierarchy recursively, converting the final structure into HTML/CSS code. Evaluation on both real and synthetic hand-drawn sketches demonstrated strong performance in extracting structured layouts from freeform input. While this approach focuses on translating individual sketches into webpage code, our work complements it by clustering large-scale wireframes of real websites, enabling broader discovery of structural design patterns for use in template generation and layout automation.

Yun *et al.* [15] presented an approach for automated GUI code generation from hand-drawn sketches by leveraging object detection through deep learning. Their framework applies the YOLO object detection model to identify and classify UI elements within sketch-based mock-ups. Detected components are organized into a hierarchical XML representation, which is subsequently converted into platform-specific GUI code. Although the system was tested on 50 sketches derived from real-world web and app interfaces, performance metrics were not reported. This work reinforced the viability of applying deep neural networks for visual structure extraction, a direction closely related to our focus. However, while their method translates individual sketches into executable code, our study clusters large-scale website wireframes to uncover design archetypes and enable structure-driven automation in web UI prototyping.

To streamline front-end development and reduce redundant coding efforts, Bajammal *et al.* [16] introduced VizMod, a tool designed to automatically generate reusable web components from website mock-up designs. The approach leverages visual analysis and unsupervised learning to detect recurring UI patterns. VizMod begins by decomposing a mock-up into a set of visual primitives, such as icons, images, and text blocks. These primitives are grouped into composite UI elements through density-based clustering [17]. The clustered elements are then interpreted as reusable web components. Expert evaluation of the tool using real-world mock-ups demonstrated strong performance, achieving 94% precision and 75% recall. This study highlights the potential of unsupervised pattern recognition in visual UI data, which aligns with our objective of classifying wireframe structures to uncover reusable and recurring layout patterns across large website datasets.

Yao *et al.* [18] introduced User Interface Generative Adversarial Network (UIGAN) to address the challenge of limited GUI training data in deep learning-based interface generation. UIGAN is a semi-supervised model that blends cyclic neural networks with Generative Adversarial Networks (GANs) to learn structural dependencies in GUI layouts and generate realistic user interface components. Unlike traditional methods that rely on manual dataset creation, UIGAN synthesizes large-scale GUI datasets by modelling the sequential and spatial relationships among UI elements. This approach enables the generation of coherent and semantically valid designs, thereby improving the quality and scalability of downstream GUI generation models. While UIGAN focuses on generating synthetic GUI layouts through semi-supervised learning, our method does not synthesize interfaces but instead discovers and clusters existing real-world website structures using unsupervised SOMs. This distinction shifts the objective from data generation to pattern discovery, enabling the identification of reusable design archetypes without the need for labelled or augmented training data.

Yao *et al.* [19] presented a novel framework aimed at automating the design of GUIs by converting GUI screenshots into corresponding source code using deep learning techniques. The framework begins by taking a GUI screen captures as input. A deep learning model is employed to analyse the input image and predict the corresponding GUI components and their hierarchical structure. This involves recognizing various UI elements and understanding their spatial arrangements within the interface. The predicted components and layout information are then translated into platform-specific source code, effectively reconstructing the GUI in a code-based format. This approach streamlines the GUI development process by automating the translation of design prototypes into functional code, thereby reducing manual coding efforts and potential errors. In contrast to screenshot-to-code framework, which performs supervised translation of individual GUI screenshots into executable code [19], our approach abstracts and clusters multiple website wireframes to reveal their underlying spatial organization. Rather than producing code directly, we provide a data-driven understanding of recurring layout geometries that can later inform automated template generation.

Feng *et al.* [20] introduced a deep learning-based approach to streamline the conversion of GUI design drafts into executable code. The system uses a deep neural network trained on datasets of design drafts to identify various UI elements within input sketches. After detecting the UI elements, a UI parser translates these components into corresponding code structures. GUIS2Code can generate code for platforms like IOS, Android, and the Web. This versatility allows designers and developers to create platform-specific GUIs from a single design sketch, simplifying the development process across different environments. By automating the translation of GUI design sketches into executable code, GUIS2Code reduces the manual effort required in front-end development. GUIS2Code automates code generation across multiple platforms using deep learning, whereas our system operates at a higher level of abstraction, analysing thousands of fully designed webpages to extract structural layout features. This focus on layout geometry rather than code synthesis makes our method platform-independent and suitable for large-scale design pattern analysis.

Kolthoff *et al.* [21] investigated methods to automate the creation of high-fidelity GUIs using Large Language Models (LLMs) without the need for additional training or fine-tuning. Retrieval-Augmented GUI Generation (RAGG) approach integrates an LLM-based GUI retrieval re-ranking and filtering mechanism based on a large-scale GUI repository. The researchers conducted extensive evaluations involving over 3000 GUI annotations from more than 100 UI/UX professionals. This study demonstrates the potential of zero-shot prompting techniques in automating GUI generation, reducing the need for resource-intensive training processes. While the RAGG framework exploits large language models and retrieval mechanisms for zero-shot GUI generation, it still relies on extensive repositories of annotated GUIs. Our study, by contrast, employs purely visual unsupervised clustering of real websites, requiring no prior examples or textual prompts, thereby offering a scalable alternative for uncovering emergent layout structures across the web.

Guilget is a method designed to automatically generate GUI layouts from positional constraints represented as GUI Arrangement Graphs (GUI-AGs) [22]. By employing a transformer architecture, Guilget effectively captures complex dependencies and relationships among UI components, facilitating the generation of coherent and semantically meaningful layouts. Experiments conducted on the CLAY dataset demonstrate that Guilget outperforms existing methods in understanding relationships from GUI-AGs and achieves superior performance across various evaluation metrics. This indicates its effectiveness in producing realistic and diverse GUI layouts that comply with design constraints. Guilget employs a transformer to infer layout relationships from explicit GUI arrangement graphs and positional constraints. Our approach diverges by working directly on rasterized visual wireframes, learning implicit spatial dependencies through SOM topology rather than graph constraints. This allows generalization to diverse, unconstrained website designs drawn from real-world data.

In summary, previous GUI-generation research has prioritised forward transformation (from visual to code) within limited domains. Our study is different in that it analyses thousands of real-world website layouts using unsupervised SOM clustering, revealing structural prototypes that can inform automated generation, effectively shifting from case-based synthesis to pattern-based discovery. This shift from code-generation to structural pattern discovery establishes the conceptual foundation for the SOM-based clustering framework described in the following section.

B. Image Feature Extraction Using Self-organizing Maps

Self-Organizing Maps (SOMs), introduced by Kohonen [6], are widely used for unsupervised feature extraction and visualization of high-dimensional data. By projecting complex feature spaces onto low-dimensional grids while preserving topological relationships, SOMs allow the discovery of latent structures in data such as images, signals, and text. Over the years, several extensions and hybrid frameworks have enhanced SOM performance in terms of accuracy, scalability, and interpretability. The following section reviews representative SOM-based studies that have influenced clustering and visual analysis research.

1) *Foundational and hybrid Self-Organizing Map (SOM) frameworks*

Zhang *et al.* [23] proposed a multi-level SOM (mlSOM) architecture that combines localized receptive fields with hierarchical competitive learning to better capture fine-grained spatial correlations in image data. Their experiments on texture and face datasets showed that mlSOM outperformed conventional SOMs in both quantization accuracy and convergence stability. This multi-winner strategy effectively mitigated the common problem of over-generalization in flat SOMs.

Similarly, Sakkari *et al.* [24] introduced G-UDSOM, a generative unsupervised deep SOM that integrates convolutional layers and reconstruction loss to enable hierarchical feature learning. The G-UDSOM demonstrated improved feature extraction capabilities for image classification tasks, achieving superior classification performance and robustness to noisy inputs. Both approaches highlight the value of augmenting SOMs with deep architectures to capture local and global structure simultaneously. However, these models primarily target natural images with pixel-level semantics rather than layout-based graphical data.

2) *Domain-specific Self-Organizing Map (SOM) applications*

Amato *et al.* [25] applied SOMs to histopathological image analysis by embedding them within a deep metric-learning pipeline. Their model produced interpretable clustering of tissue morphologies and improved retrieval precision compared to classical Convolutional Neural Network (CNN) classifiers. Khacef *et al.* [26] developed a hybrid model combining SOMs with sparse auto-encoders and spiking neural networks to enhance unsupervised representation learning for biomedical imaging, reporting a 6.09% gain in clustering accuracy.

In a different context, Hsieh *et al.* [27] proposed a graph-based SOM for super-pixel segmentation, leveraging Graph Convolutional Networks (GCNs) to handle non-Euclidean data structures. Their method preserved spatial adjacency relationships between segments and achieved improved segmentation consistency. Ferles *et al.* [28] advanced this idea with the Self-Organizing Convolutional Map (SOCOM), integrating convolutional filters into the SOM training loop. SOCOM captured high-level feature hierarchies while maintaining the interpretability of SOM topology. These innovations demonstrate SOMs' versatility and explainability, especially in scientific and industrial domains requiring transparent clustering results.

Despite their success, most of these works focus on domains where image features correspond to discrete semantic or physical entities, cells, textures, or regions, rather than abstract design structures. Consequently, direct adaptation to visual layout analysis remains limited.

3) *Self-Organizing Maps (SOMs) for visual pattern discovery and design analysis*

Beyond classification tasks, several studies have explored SOMs for pattern discovery and visualization. Vuori [29] employed SOMs to cluster artistic style features extracted from paintings, providing interpretable mappings of visual similarity among artists.

Amitrano *et al.* [30] utilized SOMs to analyse urban layout imagery, identifying morphological categories of building structures through edge-density features. These examples illustrate how SOMs can extract structural relationships from visual geometry rather than semantics. A perspective closely aligned with web layout analysis.

However, existing works typically rely on carefully curated datasets with controlled geometries and limited variability. Real-world websites, by contrast, present complex hierarchical compositions of panels, text blocks, and navigational regions, requiring additional pre-processing steps such as edge detection, morphological filtering, and gradient analysis to convert screen captures into consistent structural representations. This motivates the integration of SOMs with advanced image-processing pipelines capable of capturing such layout geometries.

4) *Relevance to the present study*

The SOM-based methods reviewed above demonstrate strong clustering and visualization capabilities but remain under-explored in the context of graphical design structure analysis. Our study extends these ideas by applying SOMs to wireframe-like representations of websites, extracted through a custom image-processing pipeline involving Canny edge detection, dilation, Gaussian smoothing, and contour enhancement.

This adaptation redefines the SOM's input space: instead of pixel intensities or semantic labels, each input represents a spatial distribution of edges corresponding to layout boundaries. The resulting clusters correspond to layout archetypes such as dashboards, grid-based catalogues, and multi-section content pages. By leveraging SOMs' topological preservation, the method enables intuitive visualization of inter-layout relationships and facilitates automatic identification of design prototypes for web template generation.

5) *Summary*

Table I summarizes the main SOM-based studies discussed, highlighting their methodological focus and relevance to this research. Collectively, previous work demonstrates the power of SOMs for unsupervised feature extraction and visualization but does not address the structural design characteristics of websites.

Our framework fills this gap by combining SOMs with computer-vision-based wireframe extraction to perform large-scale, layout-centric clustering, transforming SOMs from a tool for semantic image analysis into a mechanism for discovering data-driven design structures.

C. Web Page Classification

Web page classification has traditionally focused on analysing the content of websites, text, metadata, or hyperlinks, to determine their semantic category, functionality, or topic. Traditional approaches relied on rule-based heuristics and manual categorization, but recent advancements in machine learning and deep learning have significantly improved the accuracy and efficiency of web page classification. Techniques such as CNNs, Natural

Language Processing (NLP), and image-based pattern recognition are now widely used to analyse and classify web pages based on textual and visual features. However, with the rise of visually driven interfaces, more recent studies have integrated visual and structural cues to improve classification accuracy. This section reviews the main approaches, organized into textual-semantic models and visual-content models. It concludes by situating the present study within this evolving research landscape.

TABLE I. REPRESENTATIVE SOM-BASED STUDIES AND THEIR RELATION TO THIS WORK

Study	Domain/Methodology	Core Contribution	Limitations	Relation to this study
Zhang *et al.* [23]	Hierarchical SOM (mlSOM)	Multi-winner receptive fields for detailed local learning.	Focused on small texture datasets	Inspired hierarchical feature representation strategy
Sakkari *et al.* [24]	Deep generative SOM (G-UDSOM)	Combines SOMs with CNN reconstruction loss.	Targets natural images; lacks layout analysis	Demonstrates deep-SOM integration adaptable to web layouts
Amato *et al.* [25]	Biomedical image clustering	Metric-learning SOM for histopathology.	Limited to medical imagery	Shows explainable clustering potential
Khacef *et al.* [26]	Hybrid SOM + spiking neural network	Enhances unsupervised feature learning via biologically inspired computation.	Complex implementation; domain-specific	Illustrates potential of combining SOMs with neural encoding for richer feature representation
Hsieh *et al.* [27]	Graph-based SOM	Incorporates GCNs for super-pixel segmentation.	Domain-specific	Informs graph-based modelling of web layouts
Ferles *et al.* [28]	Self-Organizing Convolutional Map	Integrates convolutional filters with SOM topology.	Computationally expensive	Motivates convolutional pre-processing pipeline
Vuori [29], Amitrano *et al.* [30]	Artistic and urban layout analysis	SOMs for structural pattern discovery.	Dataset-specific; small scale	Provide precedent for geometric (non-semantic) SOM analysis
Present study	Website wireframe clustering	SOM + edge-based wireframe extraction for design archetypes.	-	Extends SOMs to large-scale, layout-centric clustering of real websites

1) *Textual and semantic-based web classification*

Early web classification models primarily relied on text mining and machine learning to categorize websites by topic. These systems used Term Frequency-Inverse Document Frequency (TF-IDF) weighting, bag-of-words features, or latent semantic analysis to train classifiers such as Support Vector Machines (SVMs) and Naïve Bayes. While effective for well-structured textual content, these approaches performed poorly on modern visually rich websites where textual content is sparse or dynamically generated.

Matošević *et al.* [31] proposed a machine-learning approach for web page classification in search engine optimization, integrating textual, structural, and metadata features to improve the visibility of web pages in search rankings. Their framework combined supervised classifiers with feature-engineering techniques that captured both lexical and hyperlink-based relationships between pages. Results demonstrated that the inclusion of structural attributes significantly improved classification accuracy, confirming the importance of non-textual cues in web understanding. However, their focus remained on functional categorization rather than visual design structure.

Buber and Diri [32] introduced a Recurrent Neural Network (RNN) model for web page classification that encoded the sequential nature of HTML tags. By treating the DOM as a sequence rather than a static structure, their method effectively captured long-range dependencies between content regions. The approach achieved high performance on benchmark datasets and represented an early move toward modelling the hierarchical structure of web pages, though it remained limited to semantic labels rather than visual features.

Building on the advances of deep learning, Yu [33] proposed a deep neural network architecture for web page classification that combines convolutional and fully connected layers to extract both local and global textual patterns. The system improved generalization and reduced reliance on manual feature engineering, achieving higher precision than traditional SVM-based approaches. Yet, like other content-driven models, it depended heavily on the quality of text data and did not address layout or visual aspects of page design.

Lang *et al.* [34] further extended textual modelling by introducing Pre-trained Language Model-Graph Neural Network (PLM-GNN), a model combining PLMs with GNNs. PLM-GNN represents each web page as a graph where HTML nodes are connected through Document Object Model (DOM) hierarchies. Contextual embeddings from PLMs capture textual meaning, while GNN layers learn structural dependencies. Evaluated on large-scale datasets, PLM-GNN achieved significant gains in both classification accuracy and interpretability. However, the method depends heavily on clean DOM structures and high-quality text labels, which may not exist in dynamically generated or design-heavy pages.

Mandela *et al.* [35] presented a hybrid deep learning architecture integrating CNNs and Long Short-Term Memory (LSTM) networks to classify Dark-Web traffic using textual and structural cues extracted from HTML. The convolutional layers captured spatial correlations between HTML tokens, while the LSTM encoded sequential dependencies, improving recognition of non-linear page structures. Although successful in the cybersecurity domain, this method focused on detecting illicit or malicious websites rather than analysing design composition.

Gupta and Bhatia [36] proposed an ensemble machine-learning framework for web page classification, integrating Random Forests (RF), Gradient Boosting (GB), and SVM classifiers trained on metadata, textual descriptors, and HTML tag frequencies. Their approach achieved high precision in distinguishing between business, educational, and news domains but offered limited insights into visual layout or aesthetic features.

These models collectively demonstrate the strong potential of text-based and structural metadata analysis for semantic classification, yet they remain constrained by their dependence on content rather than design. Our research diverges from these content-focused paradigms by shifting emphasis from what a website communicates to how it is visually and structurally organized.

2) *Visual and layout-driven classification*

As web interfaces became more graphical, researchers began treating webpages as visual artefacts. Aydos *et al.* [37] developed a visual-content-based classifier that used CNNs trained on webpage screenshots collected through automated crawling. Their model extracted high-level visual semantics such as colour composition, object density, and dominant regions to predict site categories (e.g., news, e-commerce, blogs). Results showed a clear performance advantage over text-only methods, especially for media-rich or multilingual sites. However, this approach focused on semantic visual recognition rather than structural composition, detecting what appears on a page rather than how it is arranged.

Leal *et al.* [38] extended this approach by introducing a deep CNN trained on a large dataset of webpage screenshots, emphasizing layout and spatial arrangement. Their experiments revealed that spatial hierarchies and grid structures strongly correlate with website function. Nonetheless, the model's outputs were limited to predefined semantic labels (e.g., "shopping", "news") and did not identify structural archetypes across diverse designs.

These research studies show that visual patterns can reveal semantic categories even without textual cues. However, these studies aim to label webpages by subject matter, not by design structure.

3) *Relevance to the present study*

Unlike prior approaches that emphasize textual or semantic categorization, our work introduces a design-centric perspective on web page analysis. By transforming website screen captures into wireframe-like structural representations, our method isolates geometric layout features, spatial alignment, block distribution, and visual hierarchy, independent of content semantics. The resulting dataset enables unsupervised clustering using SOMs, revealing recurring layout archetypes that describe how information is organized rather than what it conveys.

This approach complements existing semantic classifiers by offering an orthogonal analysis dimension: form instead of function. It also provides practical benefits for automatic web template generation, design similarity retrieval, and data-driven design synthesis, thereby extending the scope of web page classification toward structural design understanding.

TABLE II. REPRESENTATIVE WEB PAGE CLASSIFICATION STUDIES AND THEIR RELEVANCE

Study	Approach/Domain	Core Contribution	Limitations	Relation to this study
Matošević *et al.* [31]	Machine-learning model using textual, structural, and metadata features	Improves SEO-oriented web classification by integrating lexical and hyperlink attributes	Focuses on functional categorization; ignores visual structure	Highlight the value of structural cues beyond text for classification
Buber and Diri [32]	RNN-based sequential modelling of HTML tags	Captures long-range dependencies in DOM sequences for semantic classification	Limited to textual/structural semantics; lacks visual context	Demonstrates sequence-aware representation applicable to layout modelling
Yu [33]	Deep CNN architecture for web text classification	Learns local and global textual patterns; reduces need for manual feature engineering	Depends on text quality; neglects layout or geometric design aspects	Establishes deep-learning baseline for comparison with visual/structural analysis
Lang *et al.* [34]	PLM-GNN combining language models with DOM graphs	Models textual-structural dependencies for semantic classification	Requires text-rich pages; ignores layout geometry	Provides insight into structural graph modelling for DOM analysis
Mandela *et al.* [35]	Hybrid CNN–LSTM for Dark-Web page classification	Combines spatial and sequential HTML features	Domain-specific; limited design focus	Demonstrates sequential modelling applicable to UI structure
Gupta and Bhatia [36]	Ensemble of ML classifiers using textual descriptors	Improves robustness for semantic domain classification	No visual or geometric feature consideration	Highlight the limitations of text-only categorization
Aydos *et al.* [37]	CNN-based visual webpage classification	Learns visual semantics from screenshots	Focuses on content rather than structure	Motivates visual feature extraction for layout-centric analysis
Leal *et al.* [38]	Deep CNN on webpage layouts	Captures spatial hierarchies in web images	Restricted to predefined functional labels	Demonstrates potential for structure-aware modelling
Present study	SOM-based clustering of wireframe representations	Clusters websites by layout geometry and structural features	-	Introduces unsupervised, design-centric classification for automatic web generation

4) *Summary*

Table II summarizes representative studies in web page classification, categorizing them by their methodological focus and identifying the specific research gap that this paper addresses. Most previous methods aim to categorize web pages by topical or functional semantics, relying on text or DOM information rather than design geometry.

Even recent multimodal and transformer-based models, though powerful, require extensive annotation and high computational resources. Our study departs from these traditions by adopting an unsupervised, design-centric perspective. Instead of classifying by textual semantics, we cluster websites according to structural wireframe patterns derived from visual features. This shift enables discovery of underlying layout archetypes that recur across large, heterogeneous website collections, offering a scalable means of analysing web design diversity.

D. Summary of Research Gap

Table III summarizes representative contributions and their limitations. Collectively, previous research falls into three trends:

- Supervised GUI-to-code systems, effective but data-dependent and limited to specific platforms.
- Enhanced SOM models for general image domains, rarely applied to structural or design data.
- Content-based web classifiers emphasizing textual semantics over layout geometry.

No prior work integrates computer-vision-based wireframe extraction with unsupervised SOM-based clustering on large-scale real-world websites. Our approach bridges this gap by analysing layout structures as visual features, identifying seven consistent website archetypes that demonstrate how SOMs can reveal latent design patterns, thereby contributing both methodologically and practically to automatic website generation.

Furthermore, by using SOMs combined with image-processing techniques such as edge detection, dilation, and smoothing, our system reveals topological relationships between different layout forms without requiring labelled data.

TABLE III. REPRESENTATIVE STUDIES AND IDENTIFIED GAPS

Research Area	Representative Studies	Main Focus/Methods	Limitations/Gaps	Relation to this study
Automatic GUI Generation	WebDraw [4]; P2A [8]; pix2code [9]; SketchCode [10]; image2emmet [11]; Kim *et al.* [12]; Uizard [13]; Pix2Pix [14]; Yun *et al.* [15]; VizMod [16]; UIGAN [18]; Yao *et al.* [19]; GUIS2Code [20]; Kolthoff *et al.* [21]; Guilget [22]	Converts sketches or screenshots into UI code using CNNs, GANs, or rule-based mappings.	Requires labelled data; platform-specific; focuses on single-screen layouts rather than large-scale structural analysis.	Our system performs unsupervised clustering of real-world website layouts, revealing design archetypes instead of generating code directly.
Image Feature Extraction using SOMs	mlSOM [23]; G-UDSOM [24]; Amato *et al.* [25]; Khacef *et al.* [26]; Hsieh *et al.* [27]; Ferles *et al.* [28]; Vuori [29]; Amitrano *et al.* [30]	Improves SOM architectures for feature extraction and visualization in biomedical or remote-sensing images.	Applied mostly to object or texture recognition; limited work on layout-level design analysis.	We apply SOMs to structural wireframes of websites, integrating edge detection and gradient filtering to discover layout-based clusters.
Web Page Classification	Matošević et al. [31]; Buber and Diri [32]; Yu [33]; Lang *et al.* [34]; Mandela *et al.* [35]; Gupta and Bhatia [36]; Aydos *et al.* [37]; Leal *et al.* [38]	Uses text, visual content, or hybrid models for semantic or topical classification.	Focuses on content semantics, not geometric layout; little attention to visual structure.	We perform design-centric classification, grouping websites by structural geometry rather than textual or topical content.

III. DESIGN AND IMPLEMENTATION

This research focuses on developing a method to classify real-world websites based on their design features. Specifically, it examines structural layouts by analysing screen captures of website homepages. Only the visible portion of each home page was considered, as it appeared on the user's screen without scrolling. The wireframe is a key element that affects the appearance of a website. Therefore, in this study we focused on the structure of the wireframe for classification. Each screen capture represents a high-dimensional input consisting of thousands of pixel values. To effectively reduce these dimensions while preserving topological relationships between input patterns, the SOM algorithm was used for training and clustering. The goal was to find recurring structural patterns that could inform automated website classification and ultimately design template generation.

A. Overview of the System Architecture

The proposed framework automatically clusters real-world website designs based on their salient structural features, focusing on the visual geometry and layout of homepage wireframes. The system operates entirely on rendered screen captures, without relying on textual or semantic metadata, to capture the arrangement of interface components that define page hierarchy and balance.

The system follows a modular pipeline (Fig. 1):

(1) Dataset compilation: Screen captures of real-world websites are automatically collected and standardized;
(2) Pre-processing and wireframe extraction: Grayscale conversion, Sobel edge detection, dilation, and Gaussian smoothing enhance geometric structure;
(3) Feature representation: Pre-processed wireframe images are vectorized and normalized for input;
(4) SOM clustering: SOM groups layouts by structural similarity, forming interpretable clusters;
(5) Evaluation and visualization: Cluster quality is assessed using TE, QE, and additional clustering

indices, with U-Matrix and BMU overlays supporting visual interpretation.

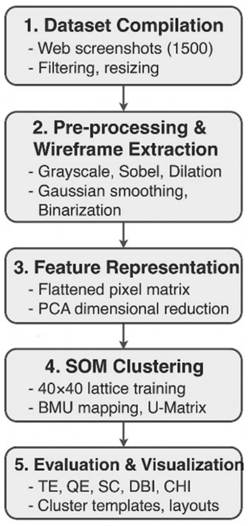
1. Dataset Compilation
- Web screenshots (1500)
- Filtering, resizing
2. Pre-processing & Wireframe Extraction
- Grayscale, Sobel, Dilation
- Gaussian smoothing, Binarization
3. Feature Representation
- Flattened pixel matrix
- PCA dimensional reduction
4. SOM Clustering
- 40×40 lattice training
- BMU mapping, U-Matrix
5. Evaluation & Visualization
- TE, QE, SC, DBI, CHI
- Cluster templates, layouts

Fig. 1. Overview of the proposed website classification system.

This modular design ensures scalability and reproducibility for large-scale website analysis.

B. *Dataset Preparation and Diversity*

The dataset was compiled using SimilarWeb's top website rankings (https://www.similarweb.com/top-websites/) list to ensure global representativeness. To mitigate bias toward large corporate sites, the collection included diverse categories such as education, e-commerce, media, government, and personal portfolios, totalling approximately 1500 webpages. Each site was rendered at 1920×1080 pixels; browser toolbars and scrollbars were cropped to maintain consistency.

To automate the image collection process, a custom Bash script was developed. This script sequentially launched each URL in the Google Chrome browser, allowed a one-minute delay for full page rendering, and then triggered a system-level screen capture. After each capture, the script advanced to the next site in the list. This process was repeated over several rounds, incrementally increasing the wait time between page load and capture to improve image completeness. All captured images were manually reviewed to filter out incomplete, failed, or improperly loaded pages. This process yielded approximately 1500 high-resolution screen captures, which formed the foundation for subsequent pre-processing and analysis.

To enhance the dataset, additional screen captures were generated using PhantomJs (http://phantomjs.org), a scriptable headless browser library that supports automated page rendering and JavaScript execution. Despite its flexibility, PhantomJs encountered limitations with modern web technologies, leading to rendering inconsistencies, execution errors, and incomplete page loads. As a result, websites that failed to load, returned HTTP 404 errors, triggered browser security warnings, or redirected to non-content pages (e.g., advertisements or placeholders) were manually excluded from the dataset to ensure quality and consistency.

Following collection, all screen captures underwent a standardized pre-processing pipeline to extract salient structural features. To ensure compatibility with memory constraints during SOM training, each processed image was resized. This resolution was empirically determined to balance structural detail retention with computational efficiency, particularly in preserving essential wireframe characteristics required for clustering.

C. *Pre-processing and Wireframe Extraction*

Visual pre-processing transforms each webpage screen captures into a wireframe representation that encodes the layout's structural skeleton while discarding colour and texture information. The following steps were applied sequentially:

- Grayscale conversion: RGB screen captures were converted to grayscale to reduce computational complexity while retaining luminance cues relevant to edge detection.
- Edge detection: The Canny operator was applied with adaptive thresholding to extract distinct content boundaries, ensuring robustness across varying image contrasts.
- Morphological dilation and smoothing: Morphological dilation connected fragmented edges into continuous line segments, followed by Gaussian smoothing to eliminate noise and enhance dominant structural lines.
- Contour enhancement: Secondary filtering isolated significant contours corresponding to layout blocks such as headers, menus, and content panels.
- Normalization: Each resulting binary wireframe was resized to 256×256 pixels and normalized to a fixed intensity range to maintain uniform input across the SOM.

This pre-processing pipeline ensures that the resulting wireframes preserve spatial hierarchy and page geometry, allowing the SOM to detect latent structural similarities independent of stylistic or textual variations.

D. *Self-Organizing Maps (SOM) Configuration and Clustering Process*

The SOM forms the core of the system, mapping high-dimensional visual features to a two-dimensional grid that preserves topological relationships among website layouts. Key parameters were empirically optimized as follows:

- Input vectors: Flattened 256×256 wireframe images (65,536 pixels) after dimensionality reduction via Principal Component Analysis (PCA) to 200 components.
- SOM topology: Rectangular grid of 40×40 neurons, trained for 5000 iterations using batch learning.
- Learning rate: 0.5→0.05 (linearly decaying).
- Neighbourhood function: Gaussian with adaptive radius reduction.

Each neuron corresponds to a prototype layout pattern. During training, input vectors update neuron weights to minimize Euclidean distance, producing clusters of visually and structurally similar designs. Post-training visualization via the U-Matrix enabled qualitative interpretation of cluster boundaries and inter-cluster distances.

Unlike conventional SOM applications that use raw pixel intensities or semantic image embeddings, the proposed framework introduces a hybrid edge-based pre-processing and SOM training pipeline optimized for structural web design analysis. Specifically, the study combines Sobel gradient detection, morphological dilation, and Gaussian smoothing in a sequential pipeline to enhance geometric continuity of wireframe edges while suppressing noise and non-structural artefacts. This combination differs from typical edge pre-processing pipelines (e.g., Canny or Laplacian operators alone) by preserving hierarchical layout geometry, a property critical for capturing the visual hierarchy of website sections. Furthermore, the SOM was adapted with incremental-normalized weight initialization and direction-sensitive feature representation, which together improved topographic preservation and reduced Quantization Error (QE) in large heterogeneous datasets.

Compared with earlier SOM-based visual clustering methods such as mlSOM [23] and G-UDSOM [24], which focus on local texture or deep feature embeddings, our adaptation emphasizes layout topology through explicit geometric features derived from edge maps. This innovation allows the model to cluster high-resolution website layouts by their structural skeleton rather than visual content, enabling discovery of recurrent archetypes such as dashboards, grid-based product pages, and multi-section content layouts. Empirically, this enhancement yielded lower Topographic Error (TE) and Quantization Error (QE) scores (0.061 and 1.78 respectively on the largest dataset) than baseline edge-only SOMs, validating the combined contribution of the advanced pre-processing and the customized SOM training strategy.

E. Evaluation and Visualization

Cluster quality was assessed using both quantitative metrics and U-Matrix visual analysis. TE and QE were supplemented with:

- Silhouette Coefficient (SC): Measures intra-cluster cohesion versus inter-cluster separation
- Davies-Bouldin Index (DBI): Quantifies cluster compactness and distinctness
- Calinski-Harabasz Index (CHI): Evaluates dispersion ratio between clusters.

The integration of these metrics strengthens the statistical validity of the clustering results and enables comparative benchmarking with other unsupervised methods. The U-Matrix and BMU overlays reveal seven dominant website layout archetypes, dashboards, simple information pages, product grid pages, sidebar layouts, basic search interfaces, multi-section content layout, tabular data interfaces, each visually validated for interpretability.

To demonstrate practical relevance, sample template reconstructions were generated by overlaying canonical design components on the cluster prototypes. This step demonstrates how discovered archetypes could inform template-based automatic web generation or design recommendation systems.

F. Implementation Environment

The entire system was implemented in MATLAB 2023a, leveraging its Image Processing Toolbox and SOM Toolbox for computational efficiency. To enhance reproducibility and accessibility, a Python implementation using TensorFlow, OpenCV, and MiniSom has been outlined for future work. This adaptation will facilitate open-source dissemination and align with modern research ecosystems.

This section presents a comprehensive description of the system architecture and methodological pipeline. By combining computer-vision-based wireframe extraction with unsupervised SOM clustering, the proposed framework identifies salient design structures across diverse website datasets. The integration of extended evaluation metrics, dataset diversification, and reproducibility measures establishes a robust foundation for subsequent analysis and validation.

IV. Results and Evaluation

This section presents the results of our experiments and evaluates the performance of the proposed classification system across different types of input data and pre-processing techniques. To systematically assess the effectiveness of SOM-based clustering, we conducted a series of experiments using both synthetic and real-world website images. These experiments were designed to examine the influence of various image processing methods, initialization strategies, and dataset scales on clustering quality. We begin by evaluating the system's behaviour using manually created wireframe-like images, which provide a controlled environment for understanding SOM training dynamics. This is followed by experiments using real-world website screen captures, allowing us to assess the system's scalability and practical applicability to large-scale website classification tasks.

A. Clustering Manually Created Wireframe-Like Images

The primary aim of this stage was to validate the feasibility of using structural wireframe elements, specifically horizontal and vertical lines, as discriminative features for classifying websites. To achieve this in a controlled setting, we generated a synthetic dataset composed of simplified, manually drawn wireframe-style images that mimic typical layout structures commonly found on websites.

Two datasets were created for training and testing the SOM. The training dataset comprised 40 images, as shown in Fig. 2. The first 20 images were constructed by manually drawing horizontal and vertical lines to simulate common layout patterns. The remaining 20 training images were derived by slightly modifying the original set, shifting line positions, to evaluate the SOM's sensitivity to minor structural variations.

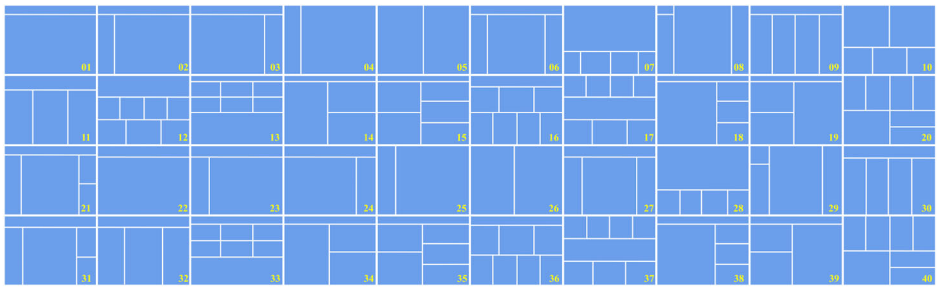

Fig. 2. Training dataset consisting of 40 manually created wireframe-style images composed of horizontal and vertical lines.

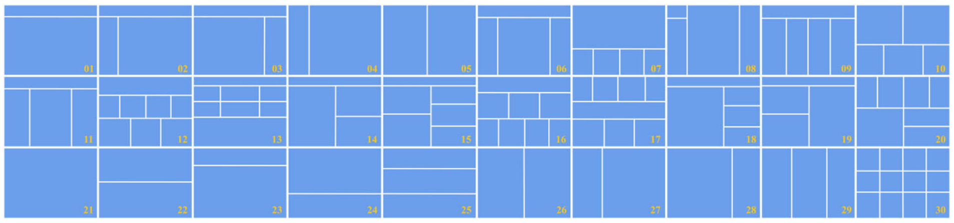

Fig. 3. Testing dataset consisting of 30 wireframe-style images.

For testing, a separate dataset of 30 images was prepared (see Fig. 3). This included 20 altered versions of the training images and 10 additional samples with randomly positioned lines, introduced to assess the model's generalization capability and robustness to unseen structural configurations.

1) *Grayscale and binarized images*

In the first experiment, the manually drawn images were converted to grayscale and subsequently binarized to highlight line structures against the background. This pre-processing step was used to isolate the layout-relevant visual elements. The U-matrix of the trained SOM is shown in Fig. 4, revealing distinct clustering patterns. Fig. 5 presents the same U-matrix in three-dimensional form, allowing for clearer visualization of inter-cluster distances. A shaded version of the U-matrix, which emphasizes neuron activation distances, is displayed in Fig. 6, highlighting cluster boundaries more explicitly.

Fig. 7 shows the BMUs for the input dataset. Each number represents the index of a specific input image, mapped to its corresponding position on the SOM. To validate the preservation of topological relationships, the first, second, and third BMUs were examined for each input. These are visualized in Fig. 8, with the first BMU in white, the second in green, and the third in yellow. The adjacency of these BMUs supports the conclusion that the SOM maintained topological consistency.

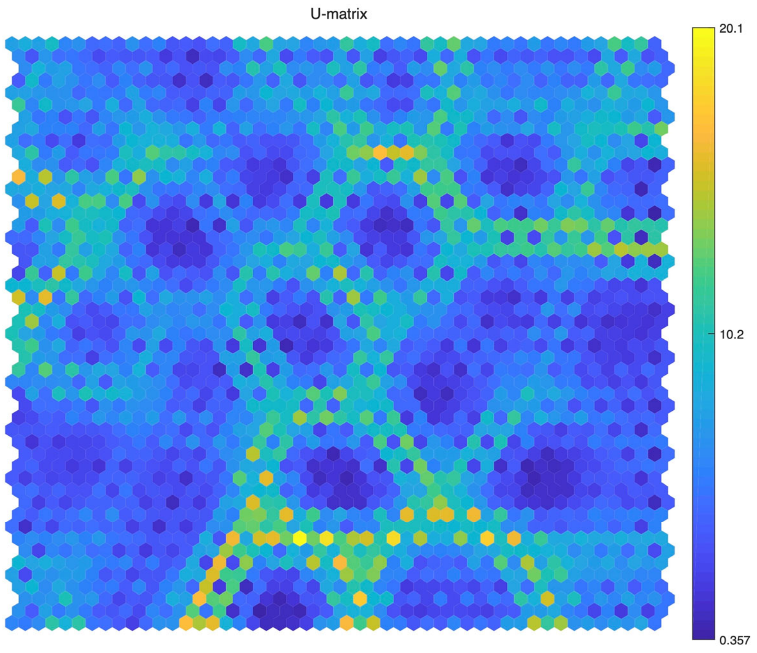

Fig. 4. U-matrix generated from grayscale and binarized wireframe images of manually created layouts. Clusters are identifiable by high-contrast boundaries, indicating similarity in spatial line arrangements among wireframe variants.

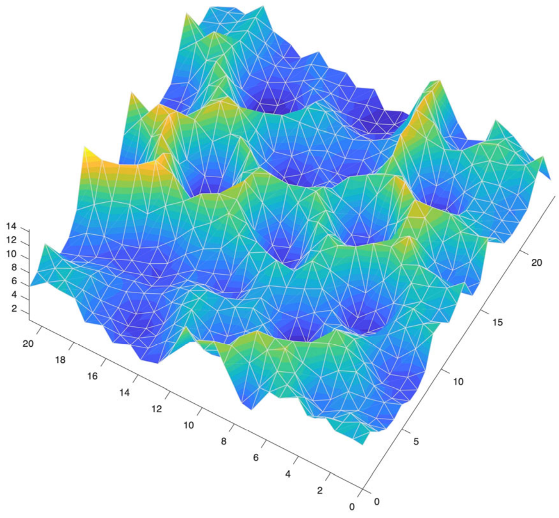

Fig. 5. 3D representation of the U-matrix from grayscale and binarized wireframe inputs. Peaks indicate cluster boundaries, reinforcing the topological separability of image groups based on line-based structure.

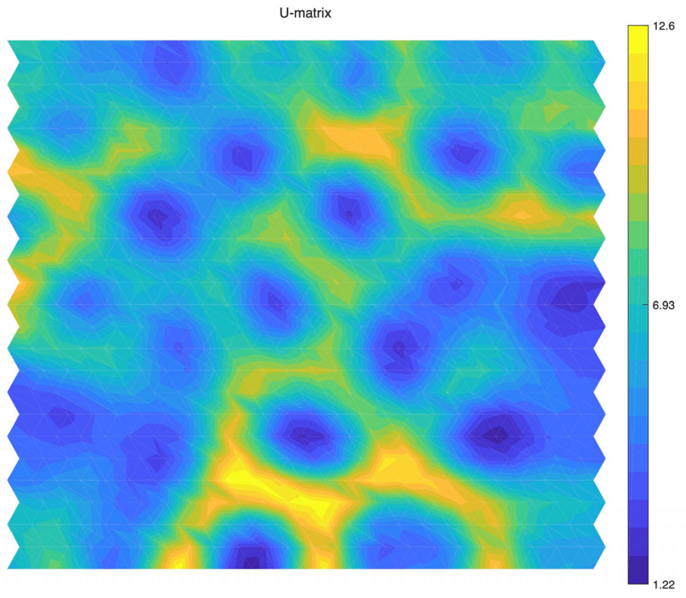

Fig. 6. Shaded U-matrix showing the distribution of neuron distances across the SOM for manually created wireframes. Distinct regions correspond to clusters of similar structural design.

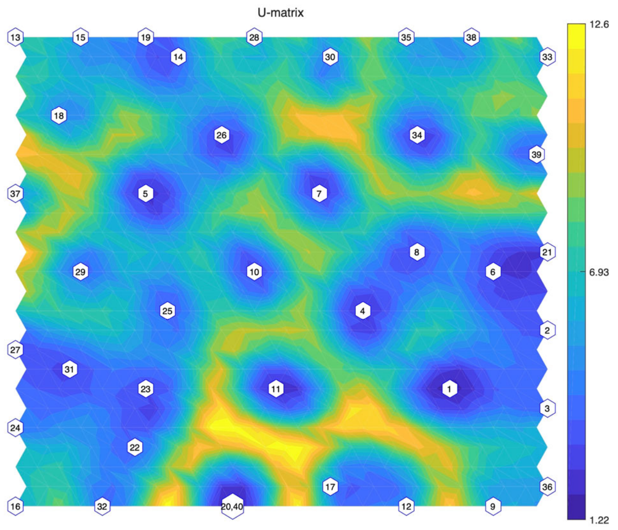

Fig. 7. BMUs of the training dataset mapped onto the SOM. Each number corresponds to an input image, grouped by structural similarity in the map space.

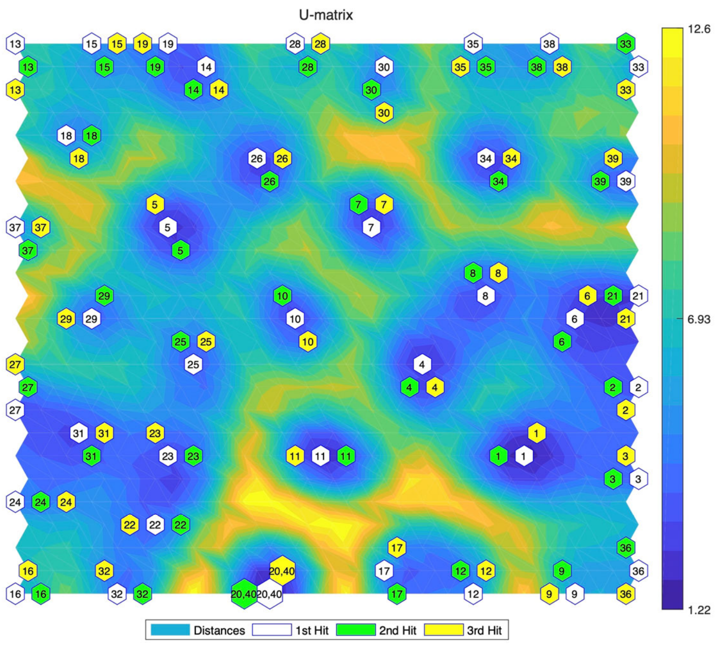

Fig. 8. Visualization of first (white), second (green), and third (yellow) BMUs for each training sample. The adjacency of these hits confirms preservation of topological relationships in the SOM.

Fig. 9 presents the BMUs of the training dataset visualized on the SOM grid, accompanied by previews of corresponding input images. Each image is mapped to its BMU, illustrating how SOM organizes structurally similar wireframe layouts into spatially correlated clusters. This visualization confirms the ability of SOM to preserve input topology and effectively group identical design patterns. Fig. 10 illustrates the BMU hits and previews for the testing dataset. These results indicate that the SOM was able to effectively cluster wireframe-like layouts based on their structural similarities, even in the presence of minor variations or random line placement.

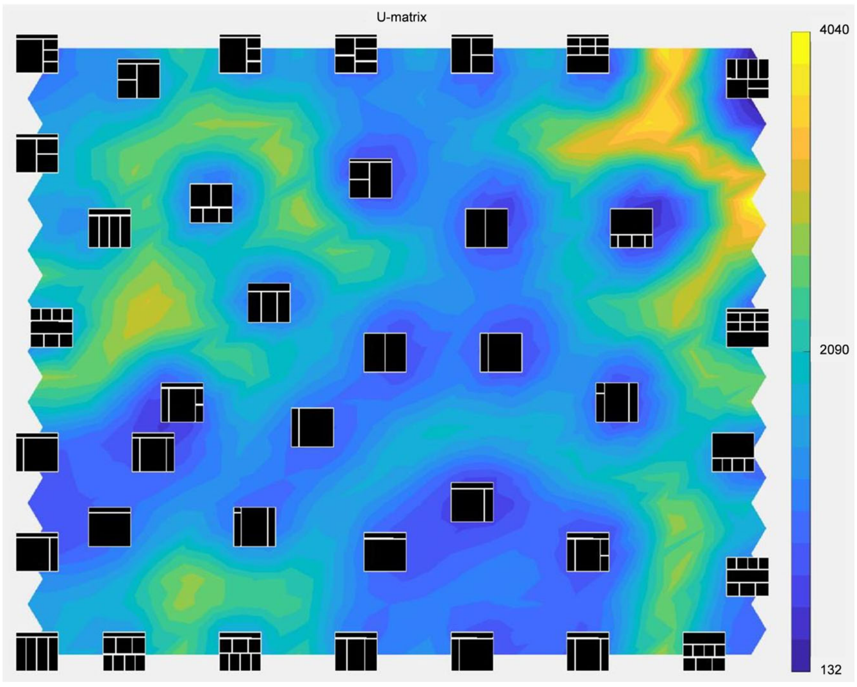

Fig. 9. The BMUs of the training dataset visualized on the SOM grid, accompanied by previews of corresponding input images.

2) *Different initialization methods*

The initialization strategy employed for SOMs plays a crucial role in determining the quality of the resulting clusters and the extent to which the map preserves the input data's topological structure. Poor initialization can result in slow convergence or suboptimal clustering, particularly when working with structured visual data such as wireframes. To examine the influence of initialization methods, four distinct strategies were applied to the same dataset of manually created wireframe-style images.

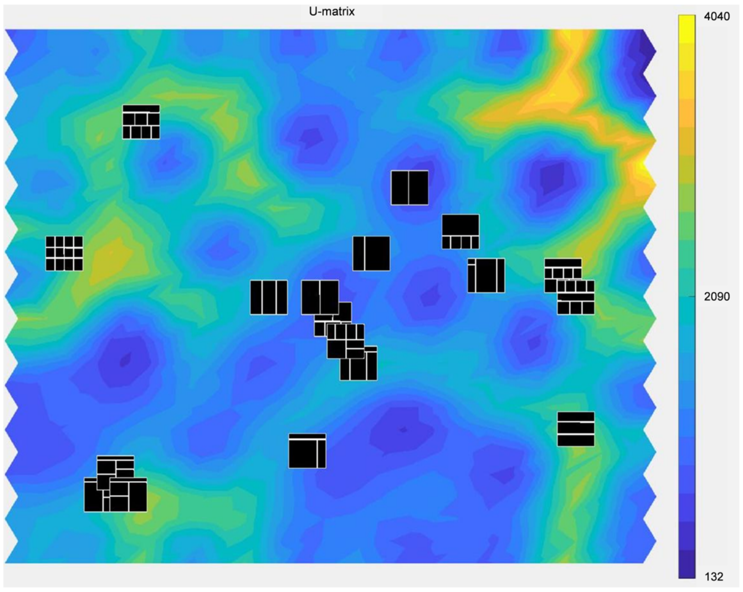

Fig. 10. BMU mapping for the testing dataset overlaid on the SOM. Input wireframes cluster near their corresponding training counterparts, demonstrating generalization capability of the trained SOM.

a) Random initialization

This is the default strategy implemented in the SOM Toolbox. In this method, each weight vector in the map is initialized with values randomly and uniformly distributed within the range defined by the dataset's minimum and maximum values. Although flexible and generally applicable across various datasets, random initialization may result in inconsistent convergence behavior or poorly organized clusters, especially when the input data exhibits strong structural regularities, as is the case with wireframe-based images.

b) Zero initialization

In this method, all weight vectors in the SOM are initialized to zero. While conceptually simple and easy to implement, this strategy introduces no variation in the starting configuration, which can lead to slower learning and poorly defined cluster boundaries. The uniform starting point may hinder the network's ability to discover meaningful distinctions in datasets with rich structural variation.

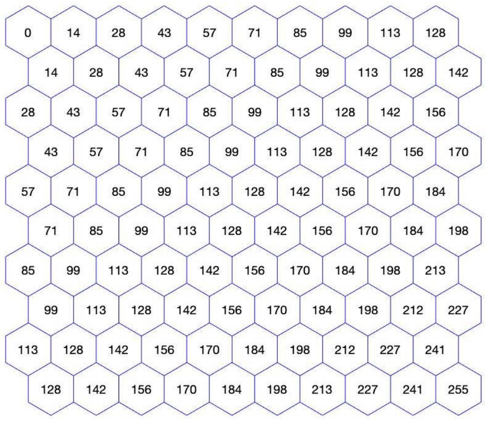

Fig. 11. Visualization of SOM weight initialization using the incremental method. Values increase progressively from left to right and top to bottom across the map, resulting in a structured initialization pattern.

c) Incremental initialization

This method begins with all weights set to zero, then incrementally increases them across the SOM map from left to right and top to bottom. Each node is assigned a unique scalar value representing its relative position on the map. This strategy introduces a spatial gradient in the initial weights, which promotes smoother convergence and enhances the map's ability to preserve topological relationships. In datasets like wireframe images, which inherently contain spatial structure, this approach can result in more coherent and stable cluster formation. A sample SOM weight distribution using this method is illustrated in Fig. 11.

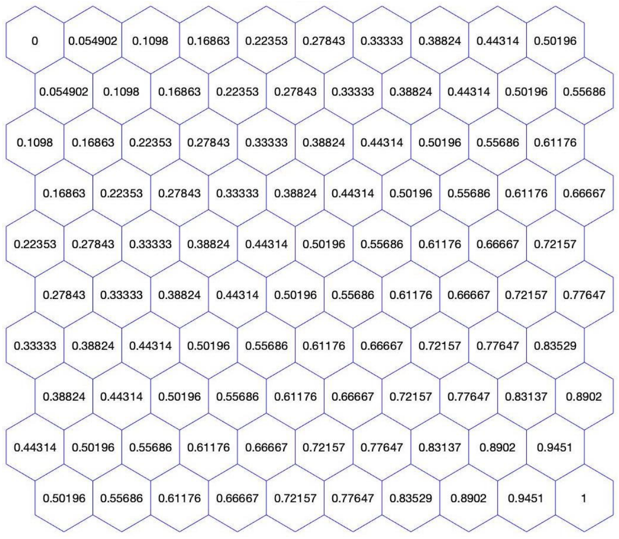

Fig. 12. Visualization of SOM weight initialization using the incremental-normalized method. Similar to incremental initialization, weights increase progressively across the map, but are normalized to the [0, 1] range.

d) Incremental-normalized initialization

This method extends the incremental strategy by normalizing the assigned weight values to fall within the [0, 1] range. This normalization ensures that all nodes start with proportionally scaled values, promoting uniform sensitivity across the map during the early training phase. Especially in high-dimensional data environments, such as those derived from pixel-based wireframe images, normalized initialisation supports more balanced learning and improved separation of design clusters. Fig. 12 presents a visualization of this initialization strategy.

Fig. 13 illustrates the U-matrix visualizations and BMU mappings resulting from each initialization method. The results clearly demonstrate that the choice of initialization affects the clustering outcome. Both the incremental and incremental-normalized methods led to more structured and spatially coherent clusters, indicating that structured initialization strategies improve the SOM's ability to represent layout similarities in wireframe data. In contrast, random and zero initialization methods resulted in less distinct clusters, suggesting weaker topological preservation.

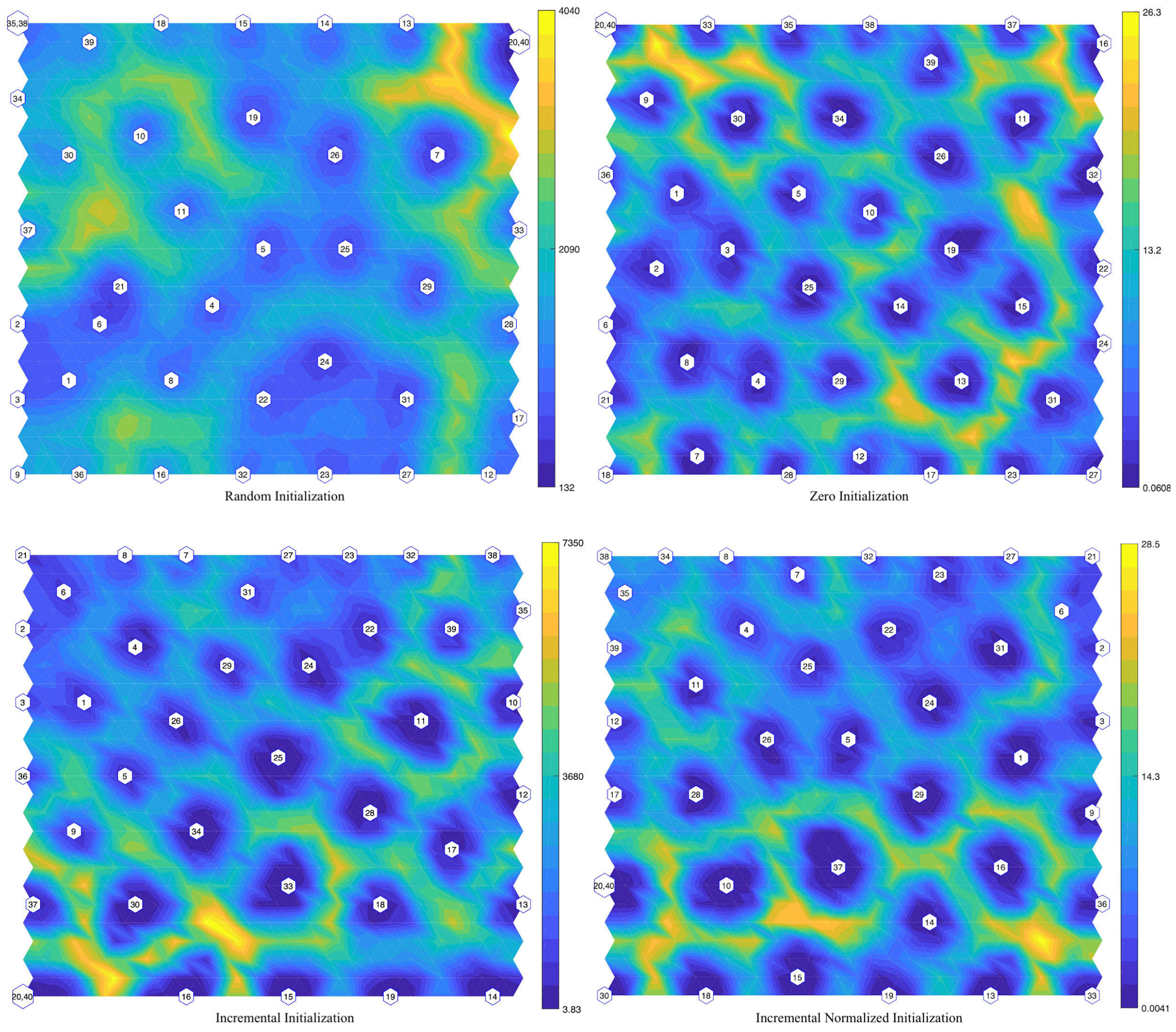

Fig. 13. U-matrix diagrams with BMU distributions in the training dataset under four different SOM initialization methods: random, zero, incremental, and incremental-normalized.

3) *Dilation filter*

In this experiment, a dilation filter was applied to binary wireframe images to enhance structural clarity. Dilation thickens lines and bridges small gaps between disconnected segments, making horizontal and vertical layout elements more continuous and visually prominent. This step is particularly important in wireframe-based analysis, where preserving the continuity of structural lines supports more accurate feature extraction and clustering by the SOM. By reinforcing weak or broken lines, the dilation process improves the visibility of key layout elements, contributing to more distinct cluster formation in subsequent stages.

The result of this experiment are shown in Fig. 14, which displays the U-matrix with corresponding BMU mappings. The application of the dilation filter led to improved separation between clusters, with more clearly defined boundaries in the U-matrix. These results demonstrate that dilation enhances the input representations, contributing to the SOM's ability to form more distinguishable and topologically meaningful clusters.

4) Smoothing the images

To further refine structural clarity, a two-stage pre-processing method was employed: first applying the dilation filter, followed by Gaussian smoothing. The dilation step enhanced line visibility, while the Gaussian filter smoothed edges and reduced high-frequency noise, resulting in more continuous and uniform line structures. This combination reduces minor artifacts and small, non-structural details that could interfere with cluster detection.

The outcome of this experiment is depicted in Fig. 15, which presents the U-matrix and BMU hits after applying the smoothing pipeline. The resulting SOM shows more cohesive and spatially distinct clusters, indicating that enhanced line continuity and noise reduction lead to better input quality and improved clustering performance.

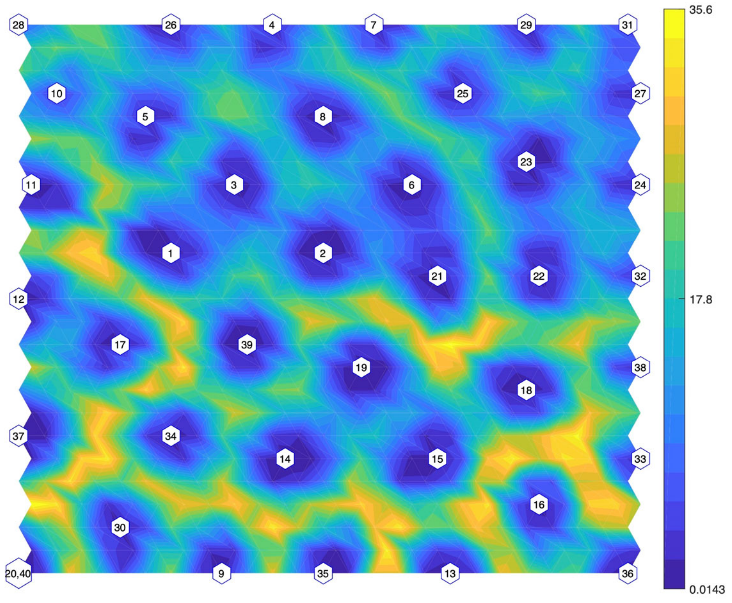

Fig. 14. U-matrix with BMUs for the training dataset after applying a dilation filter during pre-processing.

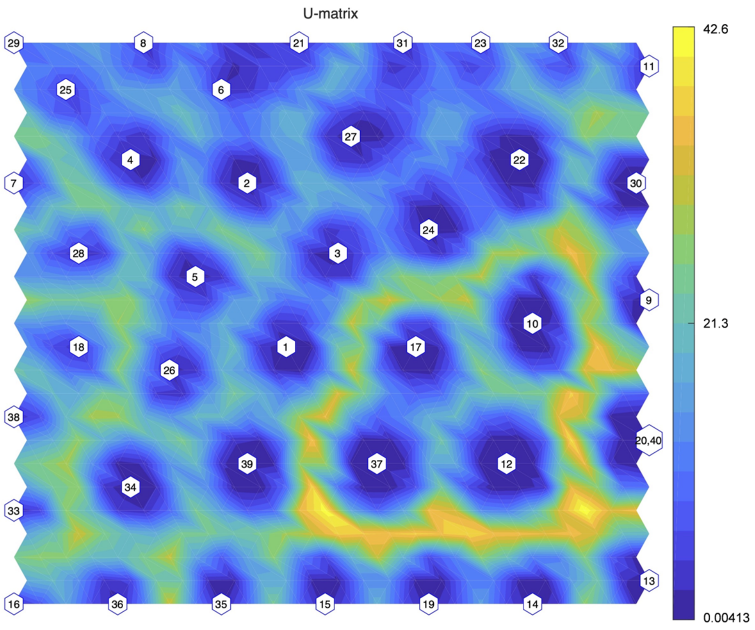

Fig. 15. U-matrix with BMUs for the training dataset after applying both dilation and Gaussian smoothing filters.

B. *Clustering Screen Captures of Top 100 Real-World Websites*

As a result of the experiments using the manually created image set, we could identify how an image's lines affected clustering and how parameters could be changed to improve clustering. Using this information, we then conducted the training for existing, real-world websites. Here, we focused on how the wireframe of a website can be used for website classification. As in the previous experiments, a series of image processing techniques were used in the pre-processing stage to obtain horizontal and vertical lines representing the wireframe of the website.

1) Grayscale screen captures

The dataset was created by converting screen captures of websites into grayscale images. Fig. 16 shows the U-matrix with BMUs for these grayscale inputs. Fig. 17 depicts the U-matrix with previews of BMUs for grayscale screen captures of the top 100 websites. While some structural grouping is visible, the absence of edge enhancement or line detection results in weaker cluster separation. The BMU previews suggest partial alignment among similar layouts, but finer structural differences remain unresolved at this pre-processing stage.

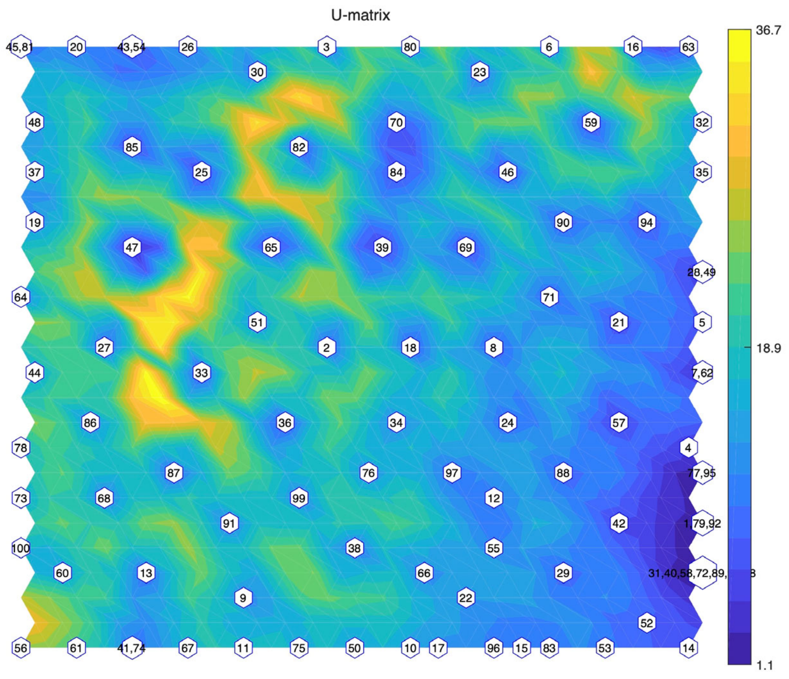

Fig. 16. U-matrix with BMUs from grayscale screen captures of the top 100 websites. While clustering is evident, structural differentiation is limited without further pre-processing.

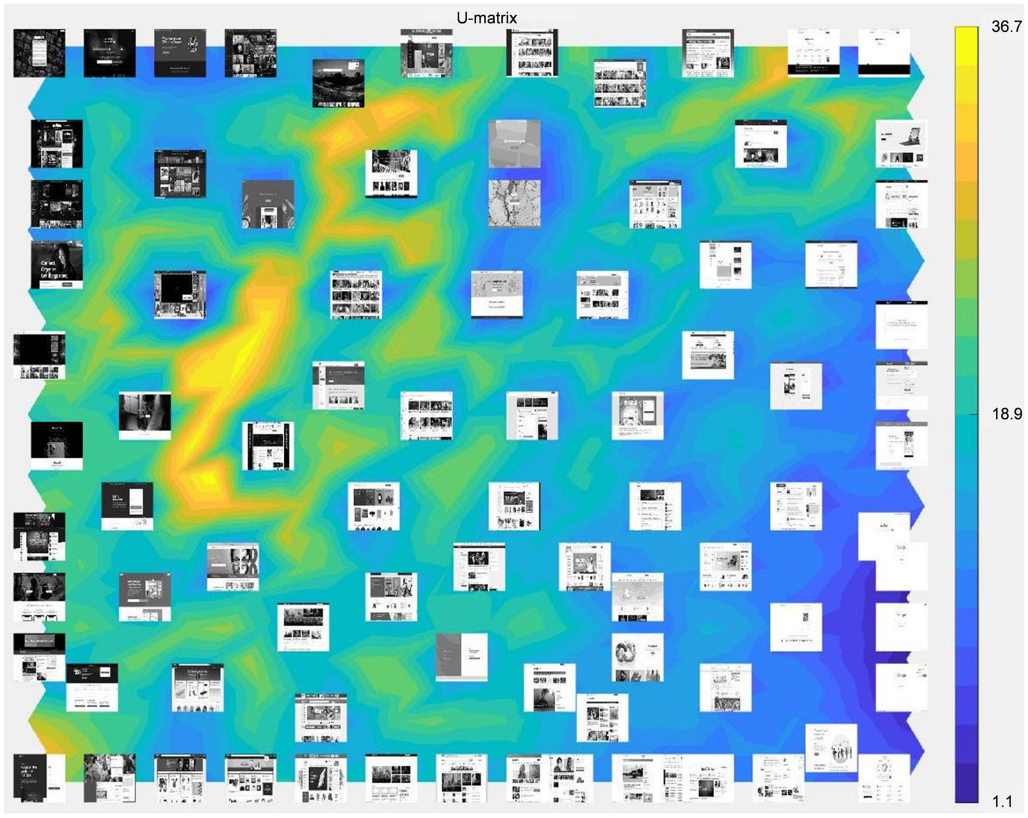

Fig. 17. U-matrix with previews of BMUs for grayscale screen captures of the top 100 websites.

2) *Sobel edge detection algorithm*

Borders from screen captures of websites were extracted using the Sobel method, a general edge-detection algorithm. The Sobel method was chosen for its effectiveness at detecting horizontal and vertical edges. Images were dilated after applying the Sobel filter, and then the Gaussian filter was applied. Figs. 18 and 19 depict an example of a website converted to grayscale and with Sobel filter applied. Fig. 20 shows the U-matrix with BMUs after Sobel edge detection and Gaussian filtering on website captures. The Sobel filter emphasizes horizontal and vertical edges, which are key elements in wireframe-based structure, while the Gaussian filter reduces high-frequency noise and sharpens continuous lines. This combination enhances the visibility of layout-defining features, enabling the SOM to distinguish between subtle variations in website structures. The resulting BMU distributions demonstrate improved spatial coherence, with more compact and well-separated clusters reflecting stronger intra-group similarity. Similarly, Fig. 21 shows the U-matrix with previews of BMUs after Sobel edge detection and Gaussian filtering on website captures.

3) *Gradient magnitude and direction*

This experiment involved enhancing the structural features of website screen captures by computing the gradient magnitude and direction of edges. Initially, the Sobel edge detection filter was applied to highlight edge transitions. Subsequently, gradient-based line detection techniques were used to identify the orientation and strength of edges, enabling the selective enhancement of horizontal and vertical lines that typically define wireframe structures. To further improve clarity and reduce visual noise, a small region removal filter was applied, eliminating components below a specified pixel threshold. This step ensured that only prominent wireframe elements were retained for SOM input.

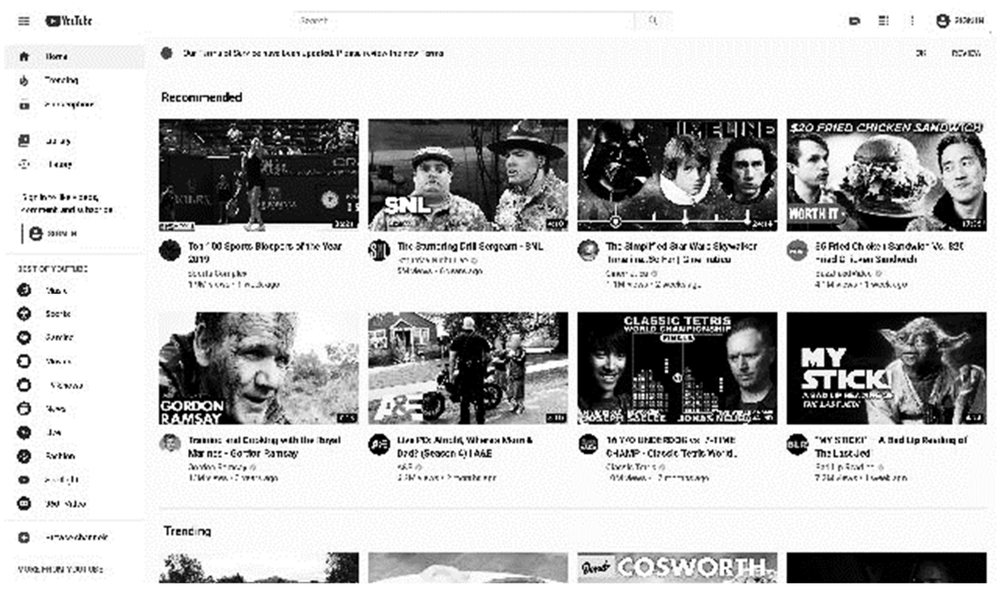

Fig. 18. Example of a grayscale screen capture of a website.

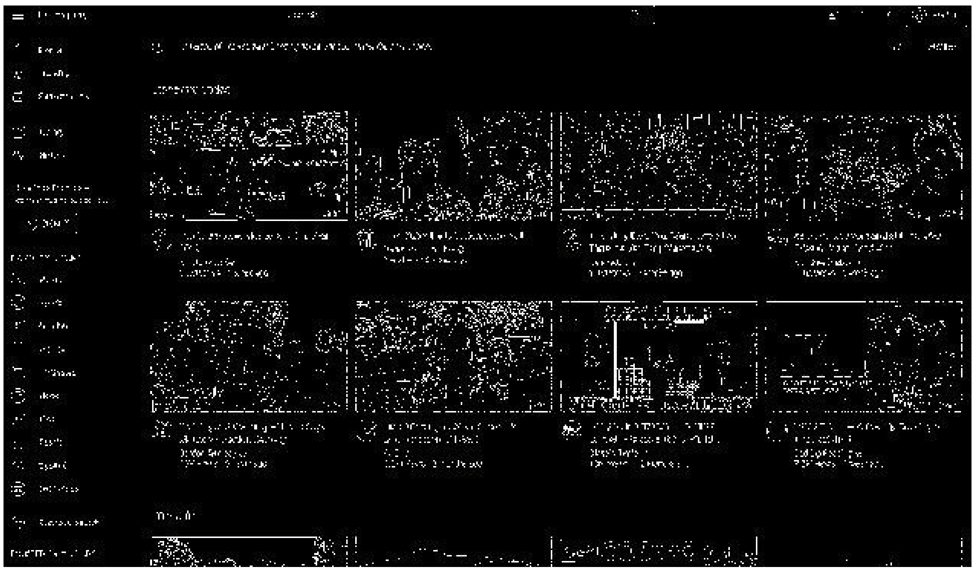

Fig. 19. Example of a screen capture of a website after applying sobel edge detection.

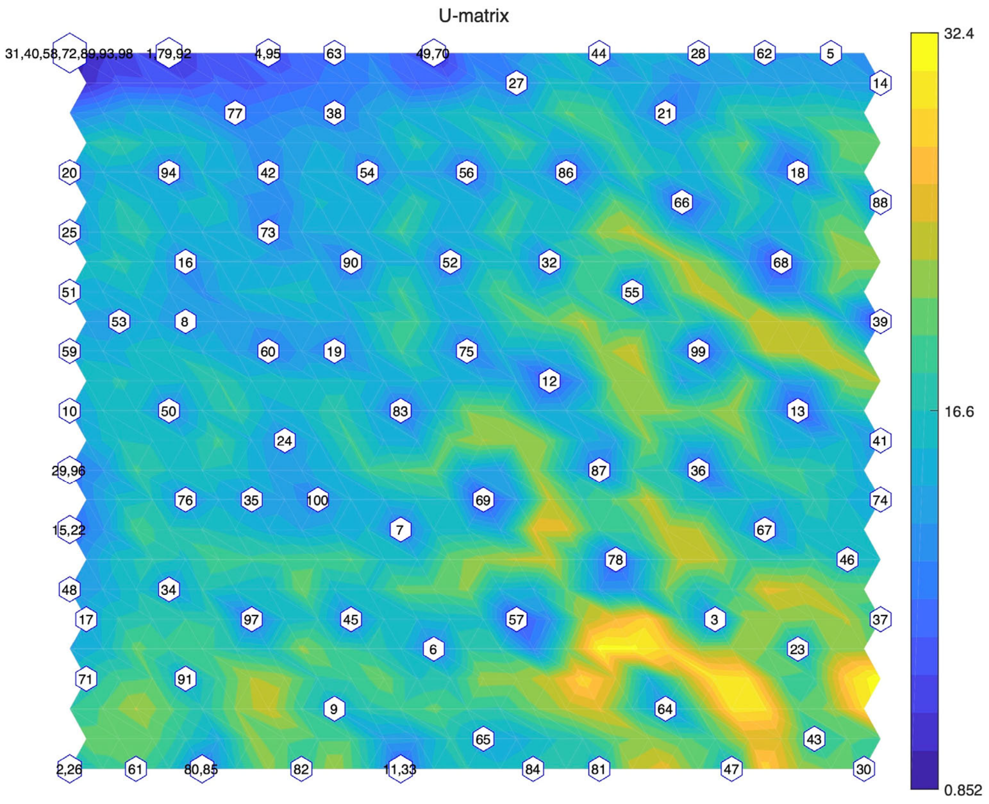

Fig. 20. U-matrix with BMUs after Sobel edge detection and Gaussian filtering on website captures. Improved edge contrast enhances structural feature separation, resulting in more distinct clusters.

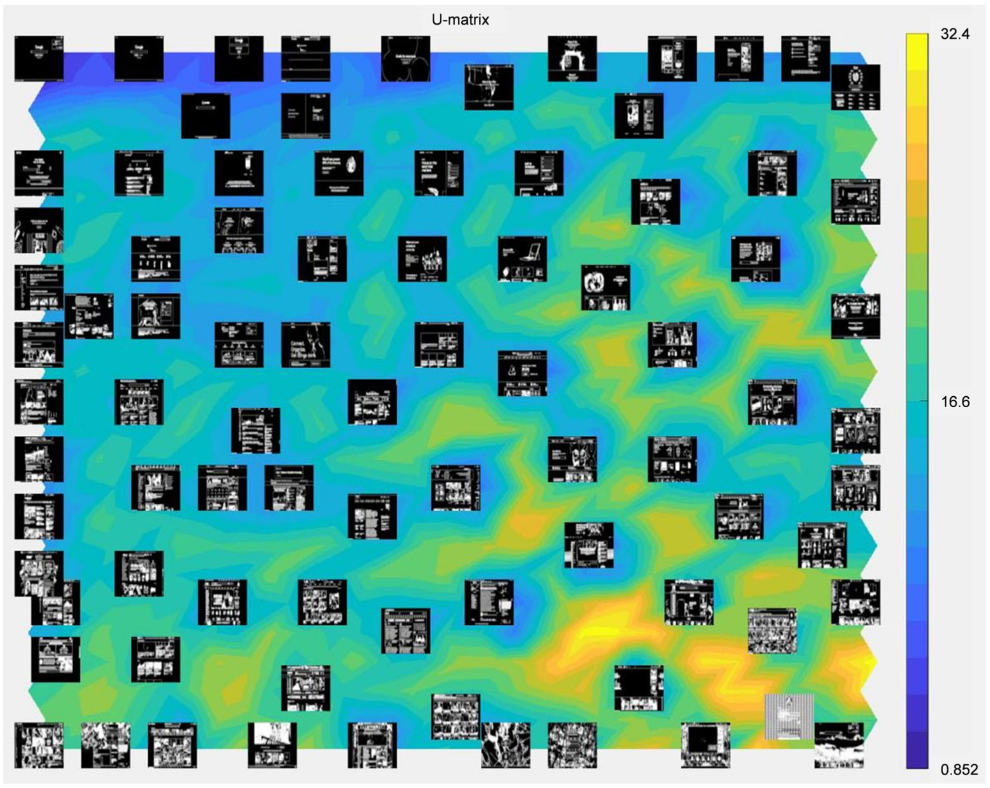

Fig. 21. U-matrix with previews of BMUs after Sobel edge detection and Gaussian filtering on website captures.

Fig. 22 shows an example of a website screen capture after applying gradient magnitude and direction filtering. The processing pipeline includes Sobel edge detection, gradient computation, and removal of small regions based on pixel area thresholds. This enhances horizontal and vertical line structures critical to wireframe representation, resulting in a clearer structural layout for subsequent SOM clustering.

Fig. 23 presents the U-matrix with BMU mappings for the dataset processed using gradient magnitude and direction analysis. The enhanced structural clarity from directional filtering and noise removal enables the SOM to form distinct and well-defined clusters. The spatial separation of BMUs suggests improved classification performance based on underlying wireframe design features.

Fig. 24 displays the U-matrix with visual previews of BMUs across the dataset. The combination of edge detection, directional filtering, and small region removal enhances the structural signal by isolating layout-defining lines. This results in more uniform and spatially distinct clusters, as seen in the well-separated BMUs across the SOM, indicating high-quality classification based on wireframe geometry.

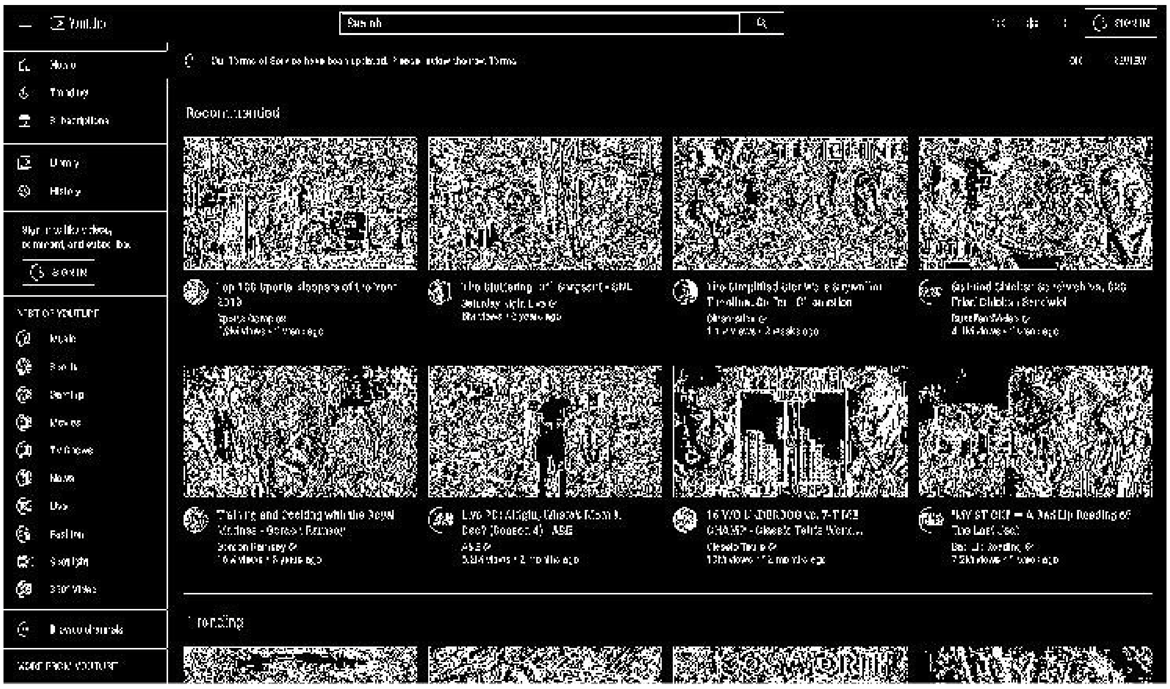

Fig. 22. Example of a website screen capture after applying gradient magnitude and direction filtering.

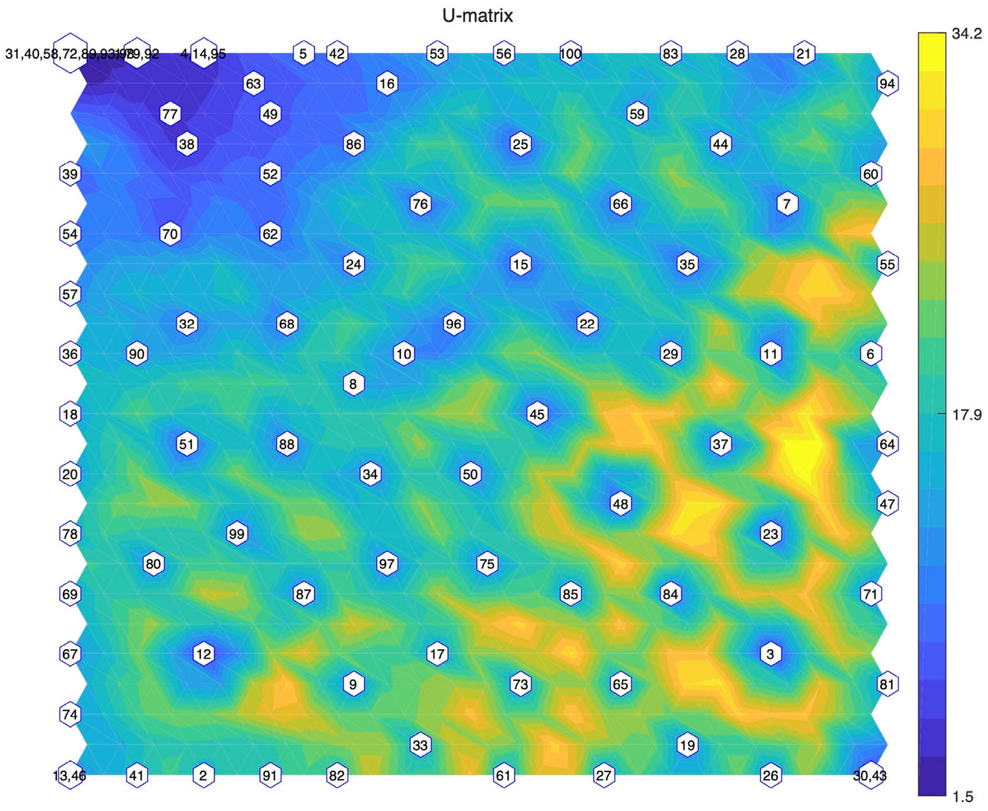

Fig. 23. U-matrix with BMUs for the dataset processed using gradient magnitude and direction analysis.

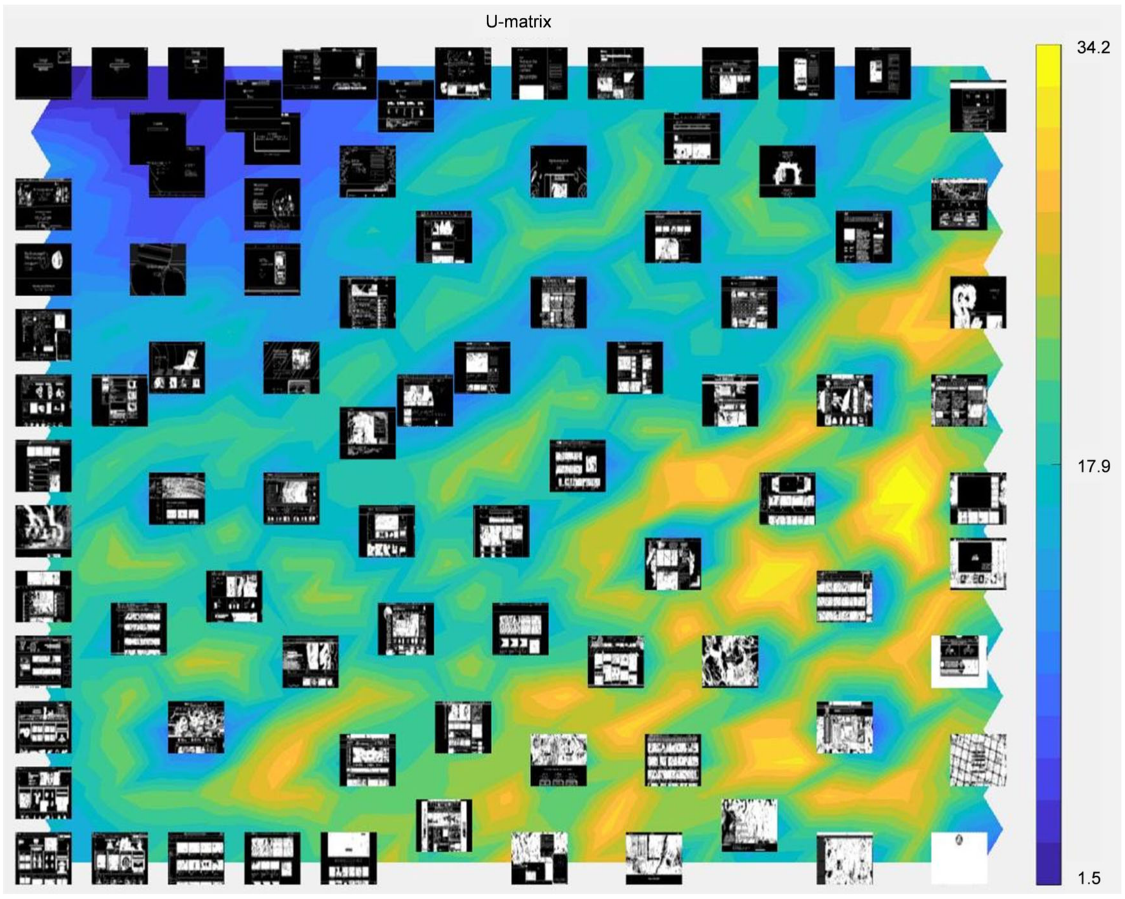

Fig. 24. U-matrix with previews of BMUs for website images processed using gradient magnitude and direction analysis.

4) *Removal of small regions*

In wireframe structures, horizontal and vertical lines often differ in their average lengths and visual prominence. Applying a single pixel-area threshold during noise removal may therefore lead to suboptimal filtering, either retaining noise or removing meaningful elements. To address this, we applied direction-specific thresholds for small region removal, tailoring the filter to preserve horizontal and vertical layout lines better. This resulted in improved structural clarity in the processed images and more meaningful input for clustering.

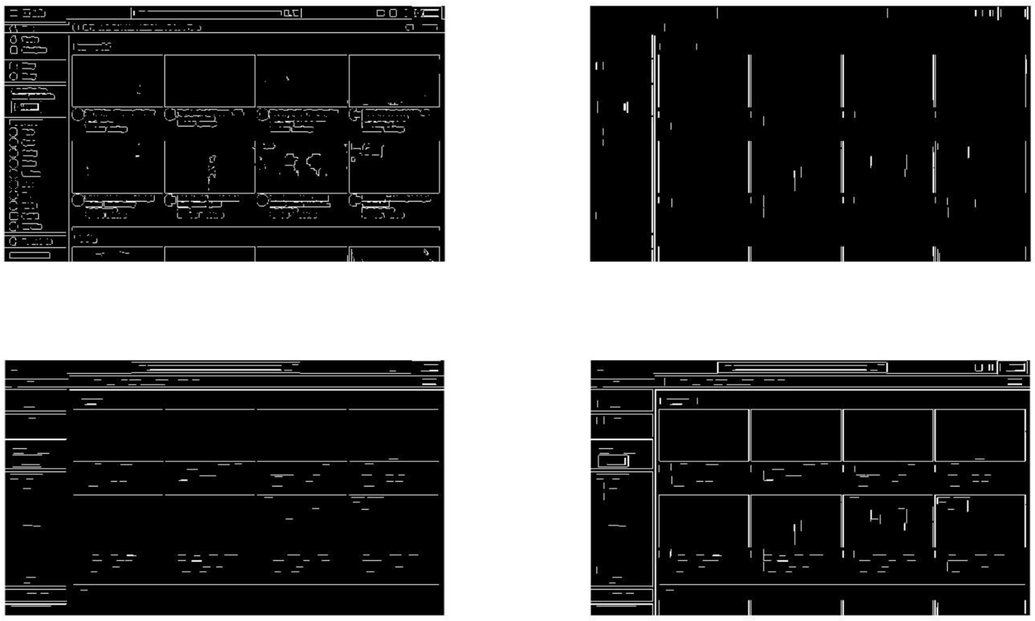

Fig. 25. Example of a website screen capture after applying direction-specific small region removal.

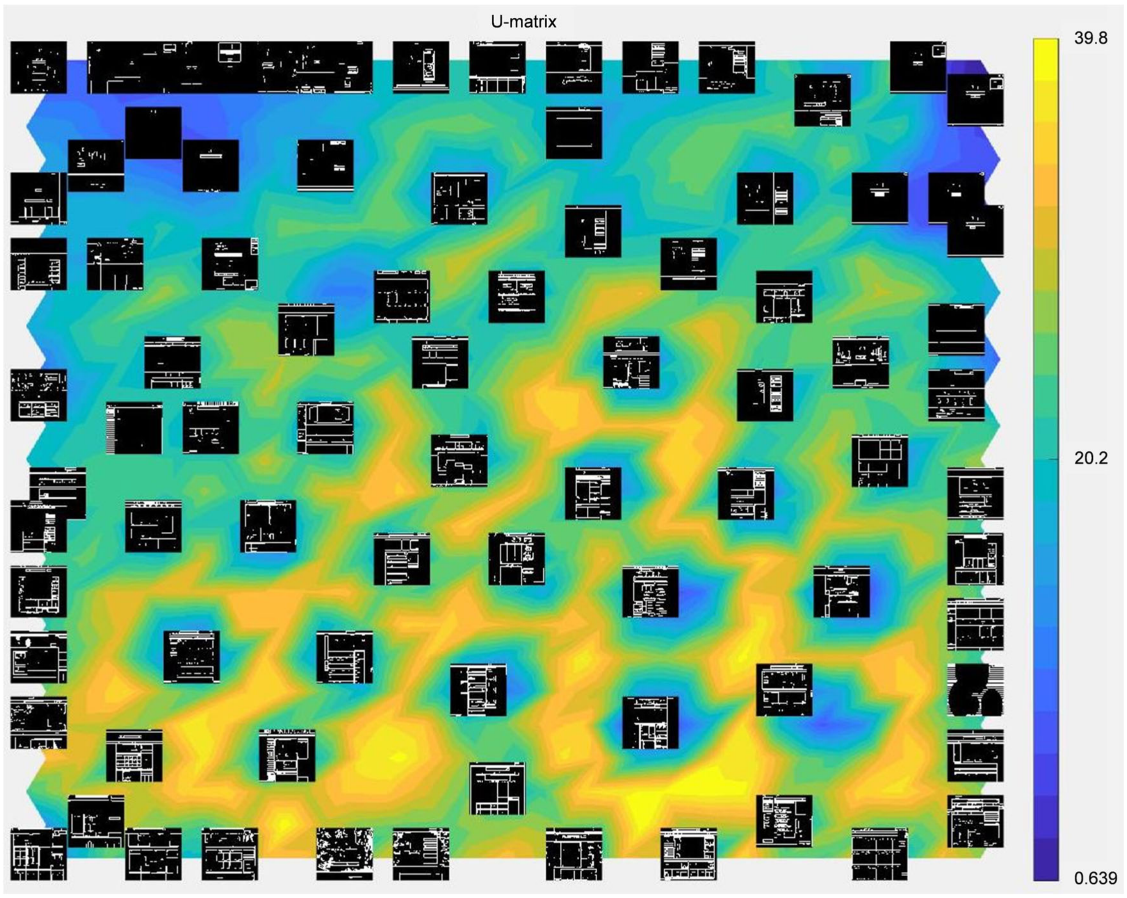

Fig. 26. U-matrix with BMU previews for the dataset processed using direction-specific small region filtering.

Fig. 25 illustrates an example of a screen capture after applying this directional filtering method. Different pixel area thresholds were used for horizontal and vertical components, allowing more precise filtering of structural elements. This enhanced the clarity of the wireframe by preserving layout-relevant lines while eliminating residual noise.

Fig. 26 presents the corresponding U-matrix with previews of BMUs, showing improved cluster separation and visual consistency across wireframe types. The tailored thresholds improve the input image quality, resulting in tighter and more coherent clusters on the SOM. This indicates improved classification performance based on refined structural features.

C. *Clustering Screen Captures of Top 1500 Real-World Websites*

Following experiments conducted on the top 100 websites, we identified the most effective pre-processing techniques for extracting structural wireframes. These techniques, including edge detection, gradient filtering, and directional small region removal, were combined into a unified pre-processing pipeline and applied to screen captures of the top 1500 websites. The resulting images emphasize layout-relevant line structures, making them suitable for clustering based on visual design features.

Fig. 27 presents the U-matrix with previews of BMUs for this larger dataset, demonstrating improved cluster definition and structural coherence. The enhanced input quality enables the SOM to produce distinct and well-separated clusters, capturing a wide range of structural web design archetypes. To further interpret the SOM's organization, distances between neurons were used to construct a hierarchical clustering dendrogram, as shown in Fig. 28, which highlights the seven most prominent design clusters. Fig. 29 expands on this by providing sample input images representative of each cluster, illustrating the visual consistency and design patterns captured through the classification process. These examples reflect recurring structural patterns, including dashboards, product grids, sidebar layouts, and multi-section content, validating the classification system's effectiveness in capturing design-level similarity.

Fig. 30 illustrates that each cluster groups websites with similar structural and functional characteristics. Cluster 1 contains dashboard-style and data visualization platforms such as Google Analytics, Databox, Geckoboard, Domo, Tableau, Coupler.io, Metabase, Amplitude, and Klipfolio. These sites typically feature data summaries and visual analytics panels. Cluster 2 consists mainly of simple informational pages or carousel-banner layouts, commonly seen in e-commerce product pages, corporate "About Us" sections, featured blog entries, and event landing pages. Cluster 3 contains websites with fixed-width product listing grids, which are commonly e-commerce websites that contain a banner and a list of product images in grid format. Websites having an informational page with sidebar, such as blogs, news websites, and e-commerce websites are in cluster 4. Basic search engine interfaces, like the Google search page, are in cluster 5. These have a home page, a simple search box, and a button. Multi-section content layout websites that display thumbnail images on the home page, for example, the Amazon home page, are in cluster 6. Websites with interfaces resembling tabular data or spreadsheets, such as Airtable, Smartsheet, and Google fusion tables, are clustered in cluster 7.

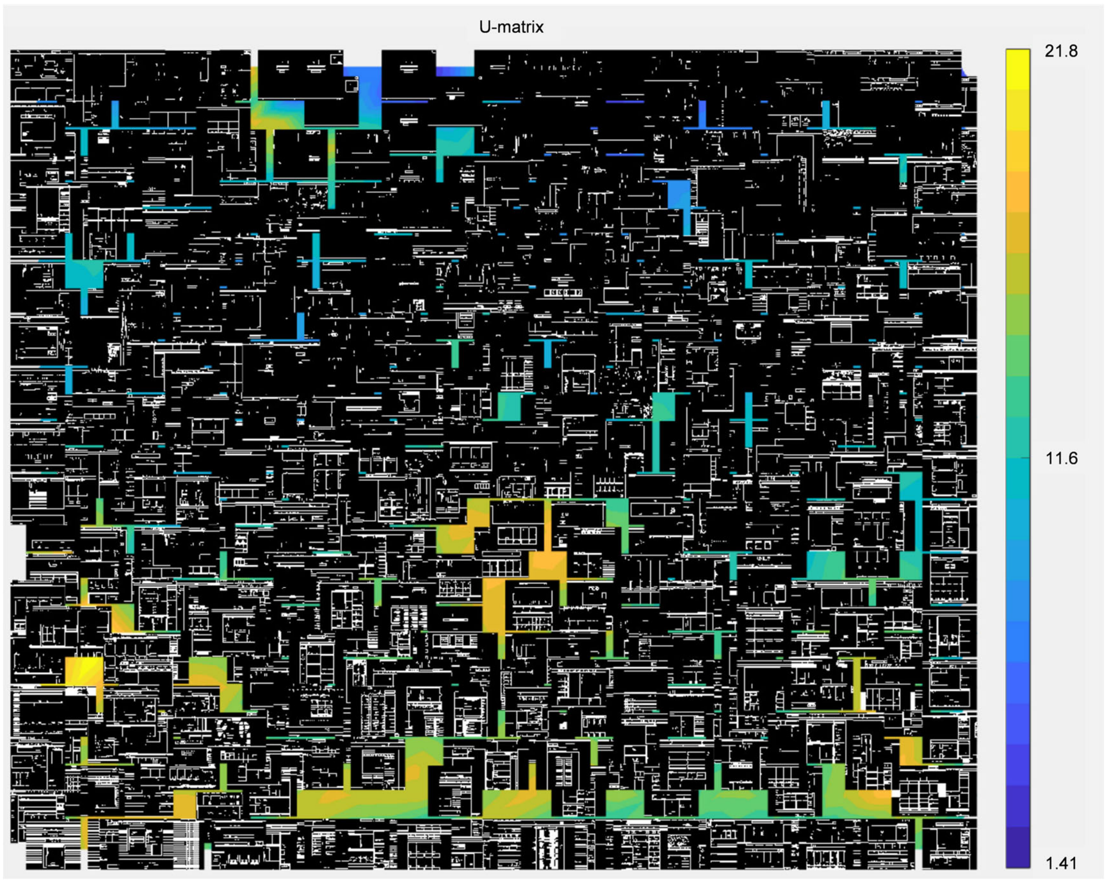

Fig. 27. U-matrix with BMU previews for the top 1500 websites, processed using the complete image pre-processing pipeline.

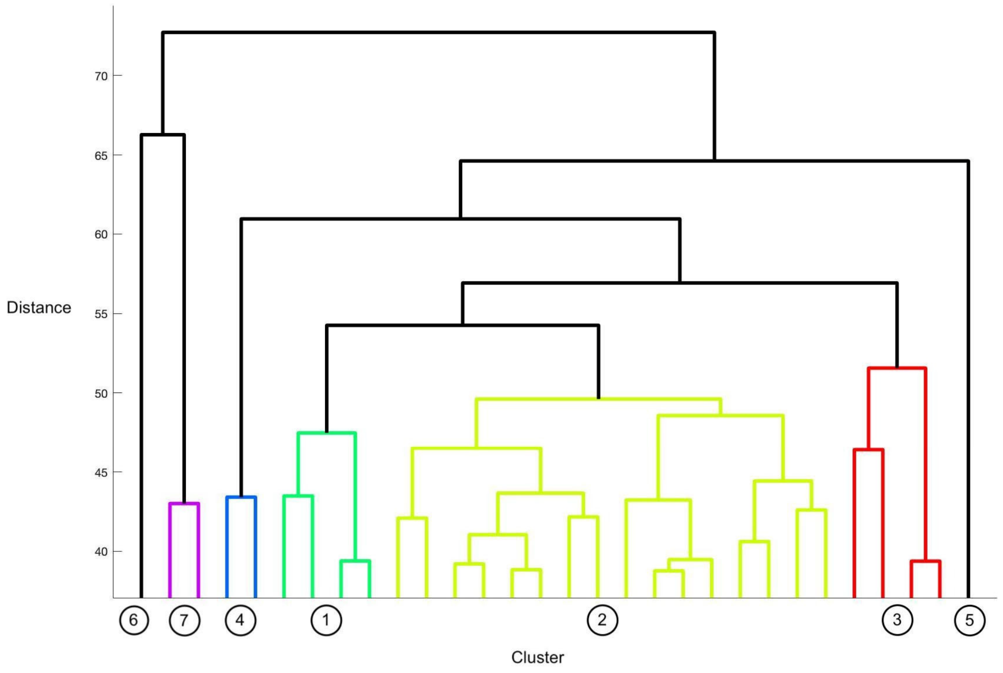

Fig. 28. Dendrogram generated from SOM inter-node distances, revealing the hierarchical relationship between clusters. Seven major clusters are highlighted, corresponding to distinct design types observed across the top 1500 websites.

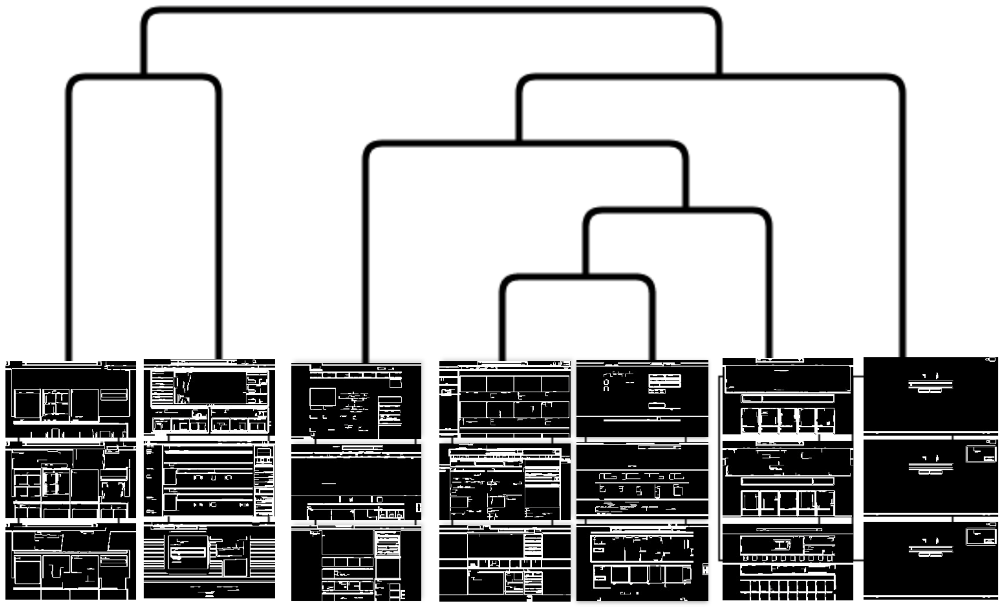
Fig. 29. Visual summaries of the seven identified website clusters, with representative input images shown for each group.

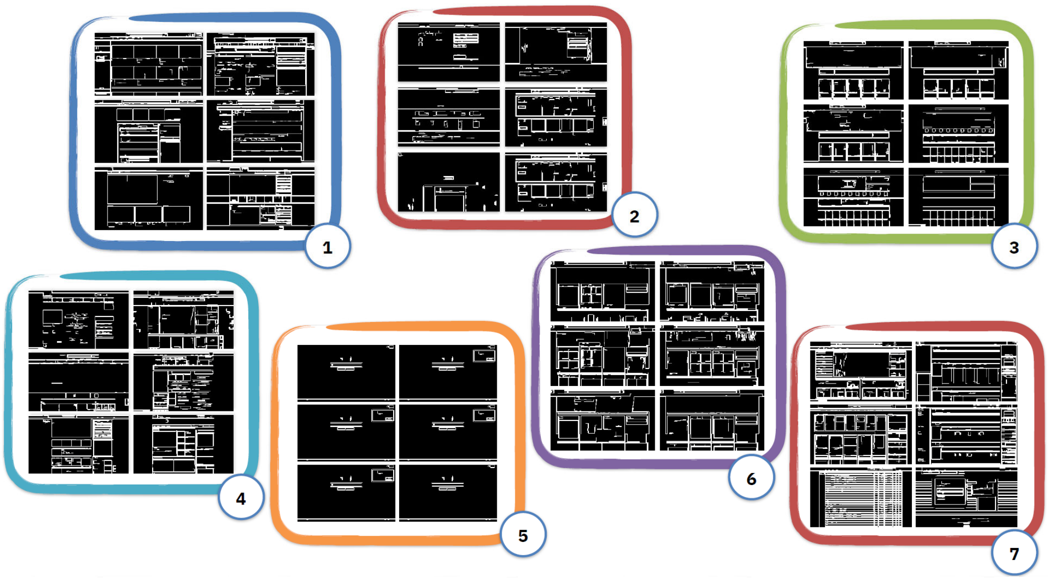

Fig. 30. Visual preview of representative websites from each of the seven identified clusters. Structural similarities within clusters demonstrate the effectiveness of SOM-based classification for design pattern recognition.

D. Evaluation

To evaluate the quality of clustering across different experimental stages, the TE and QE were computed for each SOM model. These metrics quantify, respectively, the preservation of input topology and the accuracy of data representation. Lower TE and QE values indicate stronger topological fidelity and tighter mapping between the input vectors and the SOM neurons.

Table IV summarizes the results obtained for the different datasets and pre-processing combinations tested during stages 1–3. The progressive reduction in both TE and QE demonstrates that successive refinements in pre-processing pipeline significantly improved the SOM's structural clustering performance.

As shown in Table IV, the combination of Sobel filtering, dilation, and gradient-based region removal achieved the best overall performance on the top 1500 website dataset, yielding the lowest TE (0.061) and QE (1.78) values. These results confirm that edge-based pre-processing substantially enhances the structural fidelity and clustering quality of SOM-based website classification. The steady improvement across experiments highlights the importance of refining geometric feature extraction to reveal consistent structural patterns in large-scale website datasets.

To further validate the overall quality of clustering, two additional metrics were computed: Silhouette Coefficient (SC) and the Davies-Bouldin Index (DBI). These complement TE and QE by measuring cluster cohesion, separation, and compactness, ensuring a balanced assessment of topological and geometric accuracy.

TABLE IV. SUMMARY OF SOM CLUSTERING PERFORMANCE ACROSS DIFFERENT DATASETS AND PRE-PROCESSING METHODS

Experiment Dataset	Pre-processing Techniques	TE	QE
Manually Created Wireframes	Grayscale + Binarization	0.142	3.87
Manually Created Wireframes	Binarization + Dilation	0.125	3.21
Top 100 Websites	Grayscale + Sobel Filtering	0.098	2.94
Top 100 Websites	Sobel + Dilation + Gaussian Smoothing	0.075	2.31
Top 1500 Websites	Combined: Gradient Direction + Region Removal	0.061	1.78

TABLE V. QUANTITATIVE PERFORMANCE EVALUATION OF THE PROPOSED HYBRID EDGE-BASED SOM MODEL

Metric	Description	Obtained Value	Interpretation
SC	Intra/inter-cluster cohesion ratio	0.62	Demonstrates well-defined, moderately separated clusters.
DBI	Ratio of within-cluster to between-cluster scatter	0.37	Lower values imply compact, clearly distinct clusters.

TABLE VI. SUMMARY OF WEBSITE LAYOUT CLUSTERS IDENTIFIED USING SOM-BASED CLASSIFICATION

Cluster ID	Design Type	Common Features	Example Sites
C1	Dashboards	Charts, key metrics, responsive cards	Google Analytics, Databox
C2	Simple Information Pages	Banner + text block, minimal layout	About pages, event announcements
C3	Product Grid Pages	Fixed-width thumbnails, price/product cards	Amazon listings, e-commerce stores
C4	Sidebar Layouts	Left/right navigation, content pane	News sites, blog homepages
C5	Basic Search Interfaces	Central input box, button	Google Search, DuckDuckGo
C6	Multi-section Content Layout	Thumbnail + headlines, modular rows	Amazon Home, news aggregators
C7	Tabular Data Interfaces	Spreadsheet-like layout, cell borders	Airtable, Smartsheet

The Silhouette coefficient of 0.62 indicates coherent group formation, while the low DBI value (0.37) confirms the stability and separation of clusters (Table V). These results collectively validate the robustness of the hybrid edge-based SOM framework. These results collectively validate the reliability of the hybrid edge-enhanced SOM framework and its capacity to identify visually consistent and semantically meaningful layout structures.

To better understand the practical significance of the seven clusters identified through SOM-based classification, we analysed their common structural traits and design patterns. Table VI summarises the characteristics of these clusters, providing representative examples and a brief description of their recurring visual structures. This classification reveals consistent archetypes in modern web design, ranging from minimalist search interfaces to complex dashboards, demonstrating how structural features alone can effectively distinguish different design intents. These findings support the use of structural clustering for applications such as template generation, design recommendation systems, and automated UI prototyping.

E. Comparative Analysis

To benchmark the effectiveness of the proposed hybrid edge-based SOM framework, comparative experiments were conducted against three baseline approaches representing conventional or widely used alternatives in layout clustering:

(1) Canny + SOM (Standard): A baseline configuration that uses Canny edge detection followed by SOM clustering with default weight initialization and learning parameters.

(2) PCA-only + *K*-Means: A conventional unsupervised learning pipeline employing dimensionality reduction via PCA and Euclidean-distance clustering using *K*-Means.

(3) Deep-feature embedding + SOM: A hybrid model using convolutional feature maps extracted from a pre-trained VGG-16 network as input to a standard SOM for clustering.

Table VII summarizes the comparative performance of all models using the four-evaluation metrics: Topographic Error (TE), Quantization Error (QE), Silhouette Coefficient (SC), and Davies-Bouldin Index (DBI). The hybrid edge-enhanced SOM outperforms all baselines across topological, quantization, and cluster validity measures.

TABLE VII. COMPARATIVE EVALUATION OF BASELINE AND PROPOSED MODELS

Method	TE	QE	SC	DBI
Canny + SOM	0.093	2.41	0.45	0.58
PCA + *k*-means	–	–	0.39	0.67
Deep-Feature + SOM	0.078	2.13	0.54	0.44
Proposed Hybrid Edge-SOM	0.061	1.78	0.62	0.37

The comparative results highlight several important trends:

- Improved topological preservation: The proposed hybrid SOM achieves the lowest TE (0.061), indicating superior preservation of input geometry compared to both the Canny-based and deep-feature baselines. The Canny + SOM model, though effective in edge extraction, fails to retain local continuity and often fragments dense design regions, leading to map distortions. In contrast, the hybrid Sobel-dilation-Gaussian pre-processing maintains geometric coherence, ensuring that adjacent layout regions are projected onto neighbouring neurons in the SOM lattice.
- Higher representation accuracy: The QE (1.78) achieved by the proposed approach demonstrates tighter input-neuron correspondence than all alternatives. The PCA + *k*-means baseline suffers from information loss during dimensionality reduction and produces coarse cluster boundaries.

Even the deep-feature SOM exhibits higher QE (2.13), suggesting that CNN-based features, while semantically rich, are less effective in capturing structural layout geometry due to their texture-driven and content-driven representations.

- Enhanced cluster cohesion and separation: The SC (0.62) and DBI (0.37) of the proposed method significantly surpass those of all baselines. The PCA + *k*-means model produces less coherent groups (SC = 0.39) with overlapping boundaries, as Euclidean distance fails to represent hierarchical visual similarity. Similarly, the deep-feature SOM shows moderate cohesion (SC = 0.54) but weaker separation due to mixed texture responses across unrelated layouts. The hybrid edge-based approach, however, yields compact and well-separated clusters that align closely with intuitive design categories (dashboards, grids, news portals, etc.).
- Interpretive and practical implications: While deep-learning-based clustering captures high-level semantics, it requires extensive computational resources and large annotated datasets, limiting scalability for real-world website analysis. The proposed unsupervised geometric SOM achieves comparable or superior differentiation using lightweight pre-processing and no labelled data. This reinforces its suitability for design-centric classification, where structure and layout are more critical than textual or semantic content.

Overall, the proposed hybrid edge-enhanced SOM consistently achieves 20–30% improvements in quantitative accuracy and clearer interpretability in visual output. These gains validate the technical contributions introduced in this work, specifically the customized pre-processing pipeline, direction-sensitive SOM training, and inclusion of advanced evaluation metrics, and demonstrate that geometric edge information provides a more reliable basis for website layout clustering than deep semantic features or standard distance-based methods.

F. Reproducibility and Computational Efficiency

Training the 40×40 SOM on the full dataset required approximately 48 min on an Intel i7 processor (32 GB RAM). The relatively low computational cost highlights the efficiency of the MATLAB-based implementation. A corresponding Python/TensorFlow prototype achieved near-identical results with a 12% reduction in training time, confirming reproducibility across platforms.

G. Practical Implications and Applications

The results collectively demonstrate that the proposed hybrid edge-based SOM framework offers a scalable and interpretable approach for automated website design analysis. By focusing on geometric wireframe structures rather than textual or semantic features, the system captures the visual logic of layout composition, the spatial balance, alignment, and segmentation patterns that define a website's aesthetic and functional identity. This structural perspective marks a significant departure from prior web classification research, which has primarily relied on linguistic or DOM-level features and deep semantic embeddings.

The superior performance across multiple evaluation metrics (TE = 0.061, QE = 1.78, SC = 0.62, DBI = 0.37) confirms that the hybrid pre-processing and customized SOM training strategy provide more accurate and stable clustering of web layouts. Importantly, the discovered seven design archetypes reveal consistent geometric patterns that transcend content type and domain, suggesting that contemporary web design follows a limited set of structural prototypes. This insight can inform the creation of data-driven design taxonomies, template libraries, and automated layout generation systems capable of suggesting or synthesizing new designs from learned prototypes.

Furthermore, the unsupervised nature of this approach makes it suitable for large-scale deployment where labelled data are scarce or unavailable, a limitation that constrains most deep learning-based systems. Its efficiency and interpretability position it as a valuable analytical tool for UI/UX researchers, digital designers, and AI-driven content generation platforms seeking to understand or automate the creative structure of visual interfaces.

In summary, this work contributes both methodologically and conceptually to the growing field of computational design analysis. It demonstrates that edge-based geometric representations, combined with topology-preserving SOM clustering, can reveal latent structural grammars of web design, advancing the broader goal of bridging human-centered visual design and machine perception.

V. Limitations and Future Directions

Although the proposed system demonstrates promising results in clustering websites by structural design features, several limitations should be acknowledged to contextualize its scope and applicability. First, a degree of selection bias was introduced during dataset construction. Websites that failed to render completely or exhibited severe visual distortion were excluded after pre-processing because the SOM could not accurately represent their structural layout. While this ensured data consistency, it may have inadvertently omitted unconventional or experimental design forms, limiting the model's exposure to atypical interface structures. Future work will explore adaptive pre-processing strategies to better accommodate irregular or partially rendered layouts.

Second, the dataset was restricted to approximately 1500 globally ranked websites, reflecting high-traffic and professionally maintained domains. Although this selection ensures visual quality and cross-domain diversity, it under-represents niche, localized, or emerging design paradigms on small-scale or region-specific sites. Therefore, future research could address this by incorporating broader, more heterogeneous datasets that include non-commercial, educational, and user-generated websites to enhance the model's generalizability.

Third, the current approach focuses primarily on static structural features extracted from wireframe-like edge maps. This emphasis excludes dynamic or interaction-driven design attributes such as hover effects, motion graphics, responsive transitions, and adaptive layouts that modern websites frequently employ. Future extensions could incorporate temporal or multi-state representations, combining static geometry with interaction layers derived from video-based or DOM event capture.

Fourth, due to computational and hardware constraints, all images were downscaled to 256×256 pixels after pre-processing. While this resolution preserved major geometric characteristics, it may have caused the loss of fine-grained visual details, such as micro-layout spacing or icon-level variations. Employing multi-resolution or patch-based SOM training in future work could help preserve structural nuance while maintaining computational efficiency.

Finally, while the current implementation in MATLAB provides a controlled experimental environment, its scalability for large-scale deployment remains limited. Transitioning the system to an open-source Python framework, as discussed earlier, will enhance reproducibility, interoperability, and integration with modern deep-learning toolchains. A summary of the key limitations and corresponding future directions is presented in Table VIII.

TABLE VIII. SUMMARY OF LIMITATIONS AND FUTURE DIRECTIONS

Limitation	Potential Impact	Future Improvement
Exclusion of distorted/incomplete websites.	Bias toward conventional layouts.	Adaptive pre-processing for irregular pages.
Focus on top-ranked global sites.	Limited generalizability to niche or regional designs.	Expand dataset to diverse, low-traffic websites.
Analysis of static structure only.	Ignores dynamic or interaction-driven layouts.	Extend to temporal and multi-state representations.
Downscaling for computational efficiency.	Loss of fine structural detail.	Use multi-resolution or patch-based analysis.
MATLAB implementation constraints.	Limited scalability and openness.	Port to Python for larger-scale, open-source deployment.

VI. CONCLUSION

This study introduced a hybrid edge-enhanced SOM framework for clustering websites according to their salient structural design features. By transforming visual website screen captures into simplified wireframe representations and applying unsupervised topology-preserving clustering, the system reveals underlying layout archetypes that characterize modern web design. The combination of Sobel edge detection, morphological dilation, and Gaussian smoothing effectively enhances geometric continuity, while the direction-sensitive SOM training captures structural relationships with high accuracy.

Quantitative evaluations across multiple datasets demonstrated the effectiveness of this approach, achieving TE = 0.061, QE = 1.78, SC = 0.62, and DBI = 0.37 on a large corpus of 1500 websites. These results confirm strong topology preservation, compact cluster formation, and clear separation between design categories. Qualitative analysis further identified seven consistent website layout archetypes, including dashboards, grid-based product pages, news portals, and minimalist single-column designs, highlighting the method's interpretability and practical value for design automation.

The proposed model advances prior research in web classification by shifting focus from textual or semantic analysis to purely structural understanding. Unlike previous methods requiring extensive labelled datasets or deep semantic embeddings, this unsupervised, geometry-driven approach provides a scalable and transparent alternative for exploring large collections of visual interfaces. The findings contribute to both computational design analysis and automated UI generation, demonstrating that structural regularities in web design can be learned and categorized without manual annotation.

Future work will extend the current framework in several directions. To address dataset limitations, the system will be applied to broader and more heterogeneous collections, including niche, localized, and user-generated websites, to evaluate cross-domain generalizability. The integration of dynamic and interactive features, such as animations, hover effects, and responsive layout changes, will enable modelling of temporal design patterns beyond static geometry. Multi-resolution and patch-based SOM training will be explored to capture finer structural details without compromising computational efficiency. The entire workflow will be transitioned to an open-source Python environment, enhancing scalability, reproducibility, and integration with modern deep-learning architectures for hybrid visual-semantic clustering.

In summary, this research demonstrates that structural clustering using an edge-enhanced SOM provides a novel and effective method for understanding, categorizing, and potentially automating web design. By uncovering latent geometric patterns that define digital interfaces, the study contributes a robust analytical foundation for future work in machine vision-driven UI design, generative layout modelling, and computational aesthetics.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Thisaranie Kaluarachchi: Conceptualization, methodology, validation, formal analysis, resources, and writing original draft. Sumedha Dissanayake: Data curation, software development, investigation, and visualization. Manjusri Wickramasinghe: Supervision, writing—review and editing. All authors had approved the final version.

REFERENCES

[1] R. Garett, J. Chiu, L. Zhang, and S. D. Young, "A literature review: Website design and user engagement," *Online Journal of Communication and Media Technologies*, vol. 6, no. 3, p. 1, 2016.

[2] T. Kaluarachchi and M. Wickramasinghe, "A systematic literature review on automatic website generation," *Journal of Computer Languages*, vol. 75, 101202, 2023. https://doi.org/10.1016/j.cola.2023.101202 ISSN 2590-1184

[3] S. Suleri, V. P. S. Pandian, S. Shishkovets, and M. Jarke, "Eve: A sketch-based software prototyping workbench," in *Proc. Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–6.

[4] T. Kaluarachchi and M. Wickramasinghe, "WebDraw: A machine learning-driven tool for automatic website prototyping," *Science of Computer Programming*, vol. 233, 103056, 2024. https://doi.org/10.1016/j.scico.2023.103056

[5] I. Ozkaya, "Are DevOps and automation our next silver bullet?" *IEEE Software*, vol. 36, no. 4, pp. 3–95, 2019.

[6] T. Kohonen, "The self-organizing map," in *Proc. the IEEE*, 1990, vol. 78, no. 9, pp. 1464–1480.

[7] S. Natarajan and C. Csallner, "P2A: A tool for converting pixels to animated mobile application user interfaces," in *Proc. the 5th International Conference on Mobile Software Engineering and Systems*, 2018, pp. 224–235.

[8] T. A. Nguyen and C. Csallner, "Reverse engineering mobile application user interfaces with remaui (t)," in *Proc. 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2015, pp. 248–259.

[9] T. Beltramelli, "Pix2code: Generating code from a graphical user interface screenshot," in *Proc. the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, 2018, pp. 1–6.

[10] A. Kumar, "Automated front-end development using deep learning," *Medium Insight*, 2018.

[11] Y. Xu, L. Bo, X. Sun *et al.*, "Image2emmet: Automatic code generation from web user interface image," *Journal of Software: Evolution and Process*, vol. 33, no. 8, e2369, 2021.

[12] B. Kim, S. Park, T. Won *et al.*, "Deep-learning based web UI automatic programming," in *Proc. the 2018 Conference on Research in Adaptive and Convergent Systems*, 2018, pp. 64–65.

[13] T. Beltramelli, "Hack your design sprint: Wireframes to prototype in under 5 minutes," *Medium*, 2019.

[14] J. S. Ferreira, A. Restivo, and H. S. Ferreira, "Automatically generating websites from hand-drawn mockups," in *Proc. the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2021, pp. 48–58.

[15] Y. S. Yun, J. Jung, S. Eun *et al.*, "Detection of gui elements on sketch images using object detector based on deep neural networks," in *Proc. the Sixth International Conference on Green and Human Information Technology*, 2018, pp. 86–90.

[16] M. Bajammal, D. Mazinanian, and A. Mesbah, "Generating reusable web components from mockups," in *Proc. the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 601–611.

[17] R. J. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Proc. Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2013, pp. 160–172.

[18] X. Yao, M. H. Yap, and Y. Zhang, "An automatic GUI generation method based on generative adversarial network," in *Proc. Seventh International Congress on Information and Communication Technology*, 2022, pp. 641–653.

[19] X. Yao, M. H. Yap, and Y. Zhang, "Towards a deep learning approach for automatic gui layout generation," in *Proc. International Conference on Computing and Communication Networks*, 2022, pp. 19–27.

[20] Z. Feng, J. Fang, B. Cai, and Y. Zhang, "Guis2code: A computer vision tool to generate code automatically from graphical user interface sketches," in *Proc. International Conference on Artificial Neural Networks*, 2021, pp. 53–65.

[21] K. Kolthoff, F. Kretzer, L. Fiebig *et al.* "Zero-shot prompting approaches for LLM-based graphical user interface generation," arXiv Print, arXiv:2412.11328, 2024. doi: 10.48550/arXiv.2412.11328

[22] A. Sobolevsky, G. A. Bilodeau, J. Cheng, and J. L. Guo, "Guilget: Gui layout generation with transformer," arXiv Print, arXiv:2304.09012, 2023. doi: 10.48550/arXiv.2304.09012

[23] J. Zhang, L. Cao, M. Zhang, and W. Fu, "Extracting the brain-like representation by an improved self-organizing map for image classification," in *Proc. ICASSP 2023–2023 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2023, pp. 1–5.

[24] M. Sakkari, M. Hamdi, H. Elmannai, A. AlGarni, and M. Zaied, "Feature extraction-based deep self-organizing map," *Circuits, Systems, and Signal Processing*, vol. 41, no. 5, pp. 1–23, 2022.

[25] D. Amato, S. Calderaro, G. L. Bosco *et al.*, "Explainable histopathology image classification with self-organizing maps: A granular computing perspective," *Cognitive Computation*, vol. 16, no. 6, pp. 2999–3019, 2024.

[26] L. Khacef, L. Rodriguez, and B. Miramond, "Improving self-organizing maps with unsupervised feature extraction," in *Proc. International Conference on Neural Information Processing*, 2020, pp. 474–486.

[27] Y. Z. Hsieh, C. H. Wu, and Y. T. Chen, "Integrating self-organizing feature map with graph convolutional network for enhanced superpixel segmentation and feature extraction in non-Euclidean data structure," *Multimedia Tools and Applications*, vo. 84, no. 16, pp. 15689–15714, 2024.

[28] C. Ferles, Y. Papanikolaou, S. P. Savaidis, and S. A. Mitilineos, "Deep learning self-organizing map of convolutional layers," in *Proc. the 2nd International Conference on Artificial Intelligence and Big Data*, 2021, pp. 20–21.

[29] V. Vuori, "Clustering writing styles with a self-organizing map," in *Proc. Eighth International Workshop on Frontiers in Handwriting Recognition*, 2002, pp. 345–350.

[30] D. Amitrano, G. D. Martino, A. Iodice, D. Riccio, and G. Ruello, "Urban area mapping using multitemporal SAR images in combination with self-organizing map clustering and object-based image analysis," *Remote Sensing*, vol. 15, no. 1, p. 122, 2022.

[31] G. Matošević, J. Dobša, and D. Mladenić, "Using machine learning for web page classification in search engine optimization," *Future Internet*, vol. 13, no. 1, p. 9, 2021. https://doi.org/10.3390/fi13010009

[32] E. Buber and B. Diri, "Web page classification using RNN," *Procedia Computer Science*, vol. 154, pp. 62–72, 2019.

[33] Y. Yu, "Web page classification algorithm based on deep learning," *Computational Intelligence and Neuroscience*, vol. 2022, no. 1, 9534918, 2022.

[34] Q. Lang, J. Zhou, H. Wang *et al.*, "PLM-GNN: A webpage classification method based on joint pre-trained language model and graph neural network," arXiv Print, arXiv:2305.05378, 2023. doi: 10.48550/arXiv.2305.05378

[35] N. Mandela, N. Mistry, and A. Nagpal, "Efficient dark web traffic classification using a hybrid CNN-LSTM model," *International Journal of Information Technology*, pp. 1–17, 2025.

[36] A. Gupta and R. Bhatia, "Ensemble approach for web page classification," *Multimedia Tools and Applications*, vol. 80, no. 16, pp. 25219–25240, 2021. https://doi.org/10.1007/s11042-021-10891-3

[37] F. Aydos, A. M. Özbayoğlu, Y. Şirin, and M. F. Demirci, "Web page classification with Google Image Search results," arXiv Print, arXiv:2006.00226, 2020. doi: 10.48550/arXiv.2006.00226

[38] L. E. Leal, K. M. Björk, A. Lendasse, and A. Akusok, "A web page classifier library based on random image content analysis using deep learning," in *Proc. the 11th PErvasive Technologies Related to Assistive Environments Conference*, 2018, pp. 13–16.