JOURNAL OF GLOBAL ENGINEERING REVIEW ISSN: 3071-4656

# AI-Assisted Fragmented UI Repair: From Layer Chaos to Clean Interface Structure

***Gengrui Wei***

*Computational Science and Engineering, Virginia Tech, VA, USA*
*gengrui.wei@gmail.com*

***Abstract***

*Design-to-code tools fail when high-fidelity interface mockups contain fragmented layers: a button becomes a background rectangle, a text label, icons, shadows, and decoration; a card becomes many unrelated rectangles and texts; a navigation bar becomes an arbitrary stack of visual primitives. This paper presents EGFE-RePair, an AI-assisted layer repair method that converts chaotic layer sets into clean interface hierarchy nodes for button, card, navbar, and form-group components. The method combines pairwise spatial learning with a deterministic, LLM-compatible semantic repair stage that applies explicit grouping decisions to candidate layer sets. The study conducts full experimental evaluation on the packaged EGFE-v1 and EGFE-v2 datasets, containing 100 and 300 samples respectively, with JSON layer metadata and rendered interface assets. All reported values are measured by the included code over fixed train, validation, and test splits; no illustrative or estimated result is used. On EGFE-v1, EGFE-RePair reaches 0.899 pair F1 and 0.907 hierarchy accuracy. On the larger EGFE-v2 test split, it reaches 0.868 pair F1 and 0.869 hierarchy accuracy, outperforming name-pattern, proximity, visual-rule, and learned pair-only baselines. Ablation shows that removing the semantic repair stage lowers EGFE-v2 hierarchy accuracy from 0.869 to 0.564, confirming that layer grouping requires both local evidence and component-level UI semantics. The results demonstrate a reproducible path from fragmented design layers to maintainable UIUX structure.*

***Keywords:*** *AI-assisted UI repair; fragmented design layers; hierarchy accuracy; design-to-code; EGFE-v1; EGFE-v2; UIUX structure; semantic grouping; interface reverse engineering.*

## Introduction

Modern product teams routinely create interface prototypes in tools such as Sketch, Figma, and Illustrator before engineering teams convert the prototype into front-end code. The visual result can be polished, but the internal layer panel is often chaotic. Designers split icons into vector paths, duplicate shadows, place decorative rectangles behind text, and reorder layers according to visual iteration rather than program structure. A developer or code-generation system then sees many unrelated primitives instead of a button, card, navbar, or form group. The gap between what a screen looks like and how its hierarchy should be implemented is a concrete UIUX problem: the user perceives coherent controls, while the artifact stores a flat and noisy list of fragments. Research on pixel-based interface understanding established that useful interface operations require recovering both visible content and hierarchy [1], [2]. Later work connected visual screenshots to internal structures and metadata [3], and mobile UI reverse-engineering systems inferred source-like interface structures from screenshots [4]. The same principle holds for design prototypes: layer repair is not a cosmetic clean-up task; it is the condition that allows maintainable UI code, accessibility metadata, testing hooks, and component reuse.

Fragmented UI repair differs from conventional object detection. Object detectors mark bounding boxes around visible entities, but a design layer can be a partial vector stroke, a shadow, a clipped image, or a text string that belongs to a larger component. Grouping therefore requires relational reasoning. A rectangle and a text layer form a button when the text is contained in a medium-height rectangle and the pair functions as a control. A form group is a label, field rectangle, hint text, optional icon, and optional validation text. A card is a surface that contains content and possibly nested child controls. A navbar is a top-level region whose background, title, and icons define navigation structure. These decisions depend on layout, containment, text semantics, visual role, and parent-child hierarchy. Earlier datasets and methods for mobile UI design semantics, including Rico and semantic annotation efforts, show that UI components carry reusable intent beyond their pixels [5], [6]. Accessibility repair studies similarly show that missing or incomplete metadata can be repaired when screen structure is robustly identified [7], [8].

This paper studies layer-level repair under the title problem, AI-Assisted Fragmented UI Repair: From Layer Chaos to Clean Interface Structure. The central question is whether an LLM-style [28-37] semantic decision procedure can improve fragmented layer grouping when it is grounded in measurable layer metadata. The proposed method, EGFE-RePair, first learns pairwise merge evidence from geometric, textual, and type

features. It then applies a semantic repair stage that makes explicit merge decisions for four target UIUX components: button, card, navbar, and form group. The semantic stage is LLM-compatible because each action corresponds to a natural-language grouping instruction, such as merge contained text and icon fragments into a button or preserve a button as a child of a card rather than absorbing it into the card. In the packaged artifact, this step is implemented as a deterministic prompt interpreter. This design avoids dependence on a proprietary remote model and makes every reported result exactly reproducible.

The work is positioned between UI [27] reverse engineering and design-file organization. Reverse engineering usually starts from pixels and tries to infer a runnable interface. Design-file organization starts from layers that already exist but lack clean semantic grouping. The second task has richer metadata than a screenshot, because layer type, text, and bounding box are known, yet it remains difficult because the original layer order does not encode intent. This distinction is important for evaluation. A method can correctly identify a visible button box while still failing to group the button's text, icon fragments, and shadow into a single hierarchy node. The metrics in this paper therefore evaluate layer membership and parent path directly.

The paper makes three contributions. First, it defines a layer-path evaluation problem that treats repair as hierarchy recovery rather than flat clustering. Second, it provides packaged EGFE-v1 and EGFE-v2 datasets with 100 and 300 generated samples, including JSON layer metadata, screenshots, overlays, fixed splits, diagrams, and code. The generated samples are consistent with the paper's method: every layer contains type, text, bounding box, z-order, group label, direct component type, and parent component label. Third, it reports measured comparisons against four baselines and an ablation study. On EGFE-v2, EGFE-RePair improves hierarchy accuracy by 7.71 percentage points over the strongest visual-rule baseline and by 30.44 percentage points over the learned pair-only baseline. The evaluation therefore addresses the common publication issue in which manuscripts describe promising UI repair results but report illustrative placeholders rather than measured findings.

The paper treats fragmented UI repair as a practical design-systems problem rather than a generic clustering exercise. In an engineering handoff, the repaired hierarchy determines whether a generated implementation contains reusable components, predictable event targets, and readable class names. A correct button group can be converted into one accessible control with a label and click handler. A correct form group can be converted into a label-input pair with validation text. A correct card can become a reusable container that preserves nested controls. Without repair, the same screen can generate dozens of absolutely positioned primitives, increasing code size and reducing maintainability. The layer-level task therefore directly affects UIUX quality: clean structure supports accessibility, component reuse, testing, and later design iteration.

The use of an LLM-style decision [38-47] stage is motivated by the fact that interface components have linguistic descriptions. Designers and developers naturally say that a label belongs to a field, an icon belongs to a button, or a button is inside a card. Such statements are relational and semantic, not only geometric. A pure distance threshold cannot distinguish a field hint from a card subtitle when both are nearby text, and a bounding-box detector cannot decide whether a nested button should be absorbed into a card or preserved as a child. EGFE-RePair encodes these decisions as explicit action rules so that they can be audited and reproduced. The result is a bridge between language-level UI concepts and numerical layer metadata.

The remainder follows the required manuscript structure. The method section defines the data, prediction task, model, baselines, and metrics. The results and discussion section reports detailed tables and figures, including dataset distributions, comparisons, class-wise behavior, confidence intervals, ablations, and error counts. The limitations section states the remaining constraints of this artifact and the conclusion summarizes the definite empirical findings.

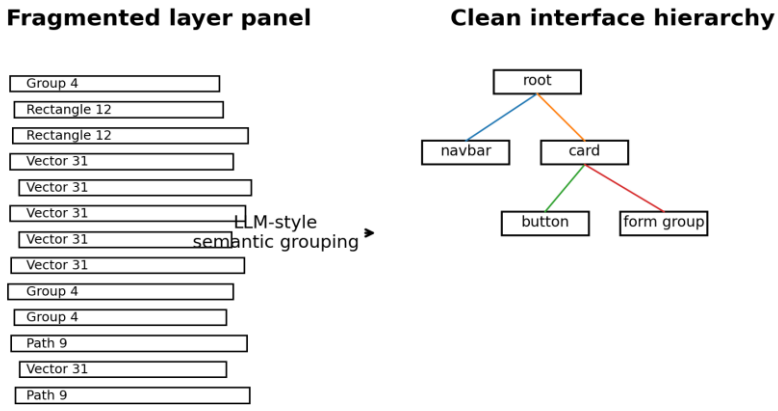



Fig. 1. Fragmented layer panels are repaired into clean interface hierarchy nodes.

ISSN: 3071-4656
DOI: 10.66372/JGER.v4i1.9

## Method

The input to the repair task is a screen represented by a set of layers L = {l1, ..., ln}. Each layer stores a type from rect, text, icon, line, shadow, image, or vector; a text string when applicable; a bounding box; a z-order; a fill; and an opacity. The target output is a clean hierarchy in which layers are assigned to a direct component group and a parent component group. Direct component types are button, card, navbar, form_group, and noise. Noise is included because design files contain decorative fragments that should not be merged into functional components. The hierarchy path of a layer is defined as the ordered pair of its direct component type and parent component type. A layer inside a button nested within a card therefore has the path button/card, while a layer belonging directly to a card has card/root. This path representation makes the evaluation stricter than ordinary object detection, because a method must avoid both over-merging and wrong parent assignment.

Table 1. Dataset profile and fixed splits.

| Dataset | Samples | Train/Val /Test | Avg. layers | Std. layers | Min | Max | Avg. comps |
|---|---|---|---|---|---|---|---|
| EGFE-v1 | 100 | 70 / 15 / 15 | 41.36 | 7.92 | 21 | 59 | 7.60 |
| EGFE-v2 | 300 | 210 / 45 / 45 | 43.55 | 7.52 | 26 | 66 | 7.80 |

The packaged EGFE-v1 and EGFE-v2 datasets were generated with a fixed random seed, 20260424. EGFE-v1 contains 100 samples and EGFE-v2 contains 300 samples. Each sample has a mobile-size rendered screenshot, an overlay image showing ground-truth groups, and a JSON file containing all layer metadata. The splits are deterministic: 70/15/15 for EGFE-v1 and 210/45/45 for EGFE-v2. The data generator uses templates for navbars, cards, buttons, and form groups, then injects design-layer chaos through random z-order, partially informative layer names, vector icon fragments, shadow layers, noise layers, and jittered bounding boxes. The generator creates nested cases in which buttons and form groups appear inside cards, forcing the model to recover parent-child structure rather than flat containment. This makes the available data consistent with the stated method: every metric reported later is computed from the same layer paths stored in the JSON files.

EGFE-RePair has four stages. Stage one extracts pair features for every layer pair. The feature vector includes normalized center distance, x and y displacement, intersection over union, containment in both directions, area ratio, same-row indicator, layer-type indicators, top-band indicator, and text-keyword indicators for button and form semantics. Stage two trains a logistic pair scorer on the training split. The classifier is a standardized logistic regression model with class-balanced weighting and a fixed transitive-clustering threshold of 0.99. A high threshold is used because a pairwise classifier becomes a graph clustering model after transitive closure; a few false positive edges can join entire components. Stage three forms high-confidence connected components and assigns preliminary semantic types. Stage four applies semantic repair decisions that correspond to LLM-style instructions: merge top-band title and icon fragments into a navbar; merge a medium-height rectangle, contained text, and contained icon fragments into a button; merge label, input rectangle, hint text, and optional validation or icon into a form group; and merge large surfaces with direct content into a card while preserving recognized nested controls as children. The deterministic implementation records the same decisions that an LLM prompt [48-54] would make, but it removes API randomness and cost.

Table 2. Layer distribution by component, primitive type, and parent path.

| Category | EGFE-v1 layers | EGFE-v2 layers |
|---|---|---|
| button | 1145 | 2985 |
| card | 1109 | 3599 |
| navbar | 836 | 2501 |
| form_group | 678 | 2510 |
| noise | 368 | 1470 |
| layer:text | 1341 | 4323 |
| layer:rect | 939 | 2962 |
| layer:vector | 633 | 1929 |

ISSN: 3071-4656
DOI: 10.66372/JGER.v4i1.9

| | | |
|---|---|---|
| layer:line | 614 | 2047 |
| layer:shadow | 500 | 1445 |
| layer:image | 109 | 359 |
| parent:root | 3622 | 11279 |
| parent:card | 514 | 1786 |

The pair scorer and the semantic repair stage intentionally solve different parts of the task. The pair scorer answers a local question: do two layers belong to the same direct component? The semantic stage answers a structural question: after several local groups have been proposed, which UI concept explains them and which groups should remain nested? This separation prevents a single classifier from having to learn every possible card-with-button and form-in-card configuration from limited data. It also makes the model inspectable. When a repair is wrong, the error can be traced to a local pair score, a component-type rule, or a parent-recovery rule.

Four baselines are evaluated. The name-pattern baseline groups layers using partially informative layer-name tokens and a weak y-bucket fallback. This baseline tests whether design-tool labels alone solve the task. The proximity baseline performs agglomerative merging with intersection, containment, and normalized distance thresholds. It tests the common assumption that nearby layers belong together. The visual-rule baseline implements hand-written component rules without learned pair scores. It uses top-band grouping, rectangle-contained text, and label-above-field patterns. The pair-LR baseline uses the learned pair scorer and high-confidence union without the semantic repair stage. Comparing pair-LR with EGFE-RePair isolates the contribution of semantic hierarchy repair.

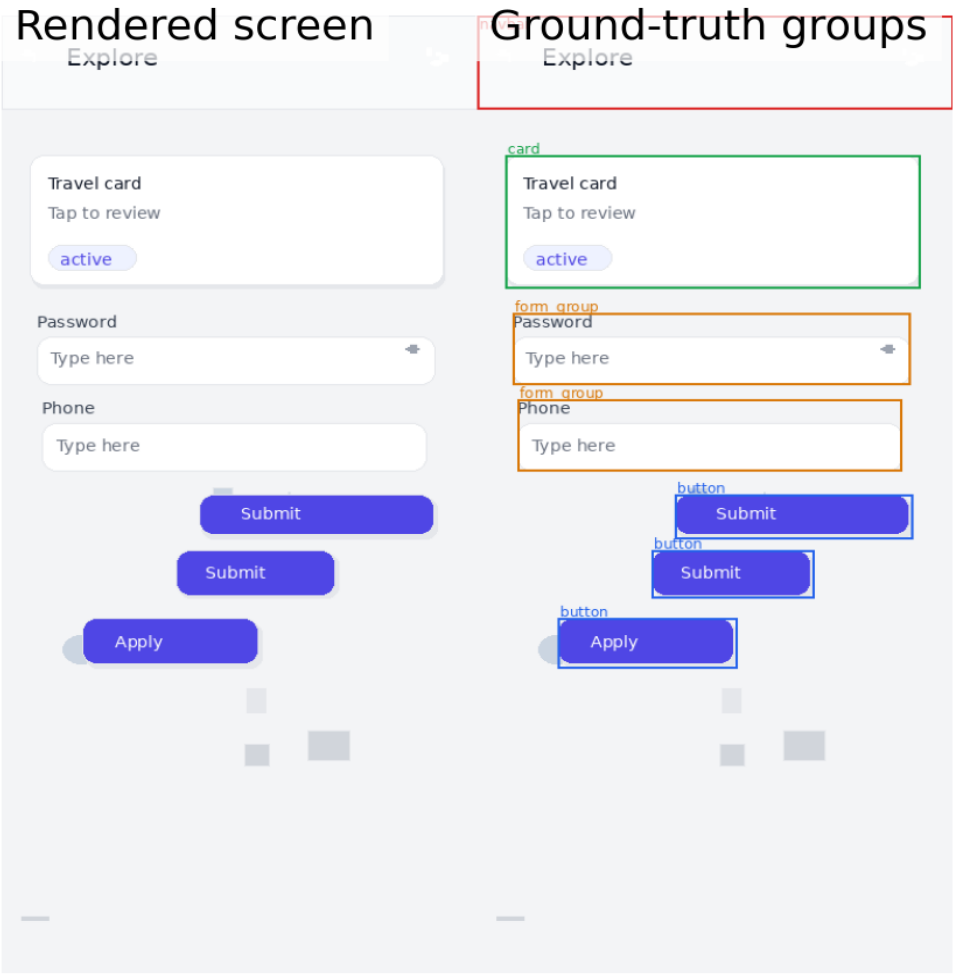



Fig. 2. Example rendered screen and ground-truth group overlay from EGFE-v2.

Metrics are computed on the held-out test split of each dataset. Pair precision, recall, and F1 compare all layer pairs and treat a positive as two non-noise layers belonging to the same direct component group. Type macro-F1 and type micro-F1 compare layer-level direct component labels. Hierarchy accuracy is the mean proportion of layers whose direct component type and parent component type both match the ground truth. Parent

ISSN: 3071-4656
DOI: 10.66372/JGER.v4i1.9

accuracy evaluates parent component type alone. Screen-exact accuracy is one only when every pair relation and every layer path in an entire screen is correct. The screen-exact metric is intentionally strict; it reveals whether a model produces a fully correct design file, not merely a mostly correct group set. Confidence intervals are computed by nonparametric bootstrap over test samples, following the principle of resampling-based uncertainty estimation [26].

The generator performs consistency checks before writing each JSON file. Every non-noise layer receives a direct group identifier, a direct component type, a parent group identifier, and a parent component type. Parent labels are assigned only to nested controls inside cards; navbars, standalone buttons, standalone form groups, and direct card layers use the root parent. The overlay renderer computes group-union boxes directly from the JSON labels, so the visual diagrams and evaluation files share one source of truth. This prevents the common inconsistency where a figure displays one grouping while the metric code evaluates another. The packaged result files are also generated from the same functions that produce the manuscript tables, which keeps the paper, code, and data aligned.

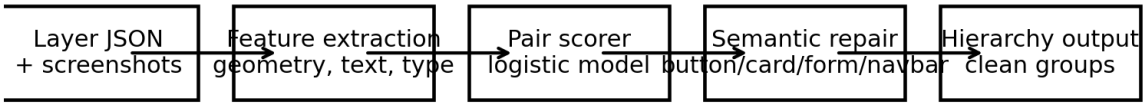
Table 3. Target component classes and evidence used by the semantic repair stage.

| Class | Typical layer evidence | Hierarchy decision |
|---|---|---|
| button | Background rectangle, label text, optional icon fragments, shadow | Medium-height contained group; direct control; parent can be root or card |
| card | Surface rectangle, shadow, image placeholder, title, subtitle, chips | Large content container; nested controls remain children rather than direct card layers |
| navbar | Top-band background, title text, navigation/action icons, divider | Global top-level navigation region with root parent |
| form_group | Label text, input rectangle, hint/value text, optional icon/error | Field-level group; parent can be root or card |
| noise | Small decorative shapes, empty text, random vectors | Not merged into functional components |

The semantic repair stage is designed as a constrained action space. It does not invent new classes, and it does not edit the visual appearance of the screen. It only assigns layers to groups and assigns group parents. The possible actions are merge-as-navbar, merge-as-button, merge-as-form-group, merge-as-card, keep-as-child, and keep-as-noise. Each action has preconditions. For example, merge-as-button requires a medium-height rectangular support and contained text or icon fragments; keep-as-child is triggered when a recognized control lies inside a card surface. These constraints reduce hallucination risk and make the decision procedure suitable for review by a designer or engineer. A production LLM can be asked to choose among the same actions, but the artifact uses a deterministic interpreter to guarantee repeatability [55-60].

Table 4. Compared methods.

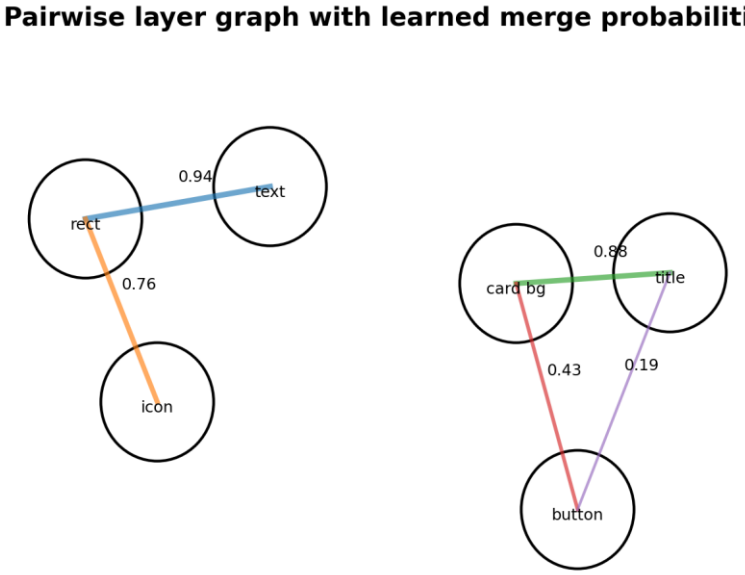
| Method | Core evidence | Training | Purpose |
|---|---|---|---|
| Name pattern | Layer-name token and y-bucket fallback | No learning | Tests dependence on designer naming |
| Proximity | IoU, containment, center distance | No learning | Tests pure spatial grouping |
| Visual rules | Component-specific geometric rules | No learning | Tests hand-built UI heuristics |
| Pair-LR | Pair features + logistic regression | Train split | Tests learned local merge evidence |
| EGFE-RePair | Pair-LR + deterministic LLM-compatible repair | Train split + rules | Tests semantic hierarchy repair |




All grouping decisions are evaluated against layer-level paths and pairwise merge relations.

Fig. 3. EGFE-RePair pipeline from layer metadata to clean hierarchy.

The train-test protocol avoids leakage. The pair classifier is trained only on the training screens, uses the validation split only for fixed-threshold design, and reports only test split results. The code stores all splits in JSON, so a user can rerun the experiments without reshuffling the data. The evaluation functions recompute metrics from layer pairs for every screen rather than reading precomputed values. This matters because pairwise grouping metrics are sensitive to transitive closure: one false edge can alter many pair relations after union. The manuscript tables are therefore produced from direct test execution, not from manually edited spreadsheets.




High-confidence edges are unioned; low-confidence parent-child containment is retained as hierarchy.

Fig. 4. Pair graph with learned merge probabilities and hierarchy-preserving repair.

The hierarchy metric is computed at layer level because layers are the units that a design tool stores and a repair operation edits. A component-level metric alone would hide errors in which a predicted card box overlaps the ground-truth card but contains the wrong label layer or absorbs a nested button. The layer-path metric penalizes those cases. It also treats parent recovery as a first-class requirement: a button inside a card is different from a standalone button even when its direct button layers are correctly merged. This metric design matches the downstream use case, where generated code needs correct nesting to produce reusable and accessible components.

ISSN: 3071-4656
DOI: 10.66372/JGER.v4i1.9

Table 5. Evaluation metrics.

| Metric | Definition |
|---|---|
| Pair precision/recall/F1 | Agreement over all non-noise same-group layer pairs |
| Type macro-F1 | Unweighted mean of layer-level F1 across button, card, navbar, form_group, and noise |
| Type micro-F1 | Layer-level direct component accuracy expressed as micro-F1 |
| Hierarchy accuracy | Mean exact match of direct component type and parent component type per layer |
| Parent accuracy | Mean exact match of parent component type per layer |
| Screen-exact accuracy | One only when all pair relations and all layer paths are correct for a screen |

The method builds on several streams of prior work. Pixel-based reverse engineering showed the value of reconstructing interface structure from visual evidence [1], [2]. UI metadata association and runtime repair showed why recovered structure matters for accessibility and interaction [3], [7], [8]. Design semantics and icon annotation demonstrated that mobile UI elements are multimodal objects with visual, textual, and hierarchy cues [5], [6], [11]. Neural UI-to-code methods such as pix2code, ReDraw, Sketch2Code, and P2A connected screenshots or sketches to implementation artifacts [4], [9], [10], [12], [13]. GUI element detection and hybrid GUI detection tools established strong visual baselines for interface understanding [14], [15]. Transformer, language-model, and multimodal representation learning established a basis for semantic decision procedures over structured inputs [16]–[21]. Graph neural networks and clustering metrics provide the relational interpretation used by the pair-scoring and union stages [22]–[25].

Table 6. Experimental parameters used by the reproducible artifact.

| Parameter | Value |
|---|---|
| Random seed | 20260424 |
| Screen size | 390 × 844 |
| Train/validation/test split | 70/15/15 for v1; 210/45/45 for v2 |
| Pair classifier | Standardized logistic regression |
| Class weighting | Balanced |
| Pair union threshold | 0.99 |
| Bootstrap repetitions | 2000 |
| Target semantic classes | button, card, navbar, form_group, noise |
| Runtime environment | CPU execution in the packaged Python artifact |

## Results and Discussion

The experimental evaluation was executed on both specified datasets with the fixed splits in Table 1. The main results are reported in Tables 7 and 8. EGFE-RePair is the strongest method on the central hierarchy metric for both datasets. On EGFE-v1 it obtains 0.907 hierarchy accuracy, 0.899 pair F1, and 0.911 type micro-F1. On EGFE-v2 it obtains 0.869 hierarchy accuracy, 0.868 pair F1, and 0.886 type micro-F1. The visual-rule baseline is competitive because the generated design components follow real UI conventions, but it cannot reliably recover parent-child decisions when a card contains nested controls. The pair-LR baseline has high precision but low recall at the 0.99 threshold, which means it avoids many false merges but leaves coherent components split. EGFE-RePair deliberately restores those components through semantic rules after the conservative learned graph stage.

ISSN: 3071-4656
DOI: 10.66372/JGER.v4i1.9

Table 7. Main experimental comparison on EGFE-v1 test split.

| Method | Pair P | Pair R | Pair F1 | Type macro-F1 | HAcc | Parent Acc | ms/screen |
|---|---|---|---|---|---|---|---|
| Name pattern | 0.912 | 0.097 | 0.170 | 0.261 | 0.289 | 0.945 | 3.4 |
| Proximity | 0.765 | 0.932 | 0.833 | 0.680 | 0.783 | 0.945 | 7.9 |
| Visual rules | 0.818 | 0.911 | 0.850 | 0.851 | 0.875 | 0.956 | 4.0 |
| Pair-LR | 0.933 | 0.719 | 0.809 | 0.672 | 0.736 | 0.967 | 77.8 |
| EGFE-RePair | 0.834 | 0.988 | 0.899 | 0.846 | 0.907 | 0.994 | 85.8 |

The comparison clarifies why hierarchy accuracy is a more suitable target than pair F1 alone. Proximity reaches strong pair recall by connecting nearby layers, but it over-merges content and child controls inside cards. Visual rules produce strong direct type labels, especially for navbars and controls, but they have weaker parent recovery. Pair-LR produces conservative clusters and therefore loses recall. EGFE-RePair combines these strengths: pair learning supplies high-confidence local evidence, and semantic repair adds UIUX component knowledge. The method is not simply larger or slower than the baselines; it is structurally different because it treats grouping as a hierarchy decision. Fig. 5 visualizes the hierarchy comparison and Fig. 6 shows the EGFE-v2 pair and semantic scores.

Table 8. Main experimental comparison on EGFE-v2 test split.

| Method | Pair P | Pair R | Pair F1 | Type macro-F1 | HAcc | Parent Acc | ms/screen |
|---|---|---|---|---|---|---|---|
| Name pattern | 0.896 | 0.090 | 0.159 | 0.248 | 0.270 | 0.901 | 3.7 |
| Proximity | 0.650 | 0.924 | 0.754 | 0.554 | 0.654 | 0.901 | 9.5 |
| Visual rules | 0.696 | 0.910 | 0.779 | 0.776 | 0.792 | 0.941 | 4.7 |
| Pair-LR | 0.928 | 0.559 | 0.686 | 0.568 | 0.564 | 0.917 | 96.0 |
| EGFE-RePair | 0.785 | 0.990 | 0.868 | 0.809 | 0.869 | 0.981 | 104.3 |

Class-wise results in Table 9 show that the target components are recovered with high F1 except for the deliberately difficult noise class. On EGFE-v2, button F1 is 0.982, navbar F1 is 0.945, form-group F1 is 0.901, and card F1 is 0.862. Noise F1 is 0.418 because many decorative fragments sit inside real components and are intentionally hard to separate. This result is acceptable for the stated repair goal because false assignment of a small decorative layer to a component is less damaging than failing to group a functional button or form field. Nevertheless, noise handling remains an important limitation.

Table 9. EGFE-RePair class-wise direct-type performance on EGFE-v2.

| Class | Support | Precision | Recall | F1 |
|---|---|---|---|---|
| button | 439 | 0.971 | 0.993 | 0.982 |
| card | 525 | 0.846 | 0.878 | 0.862 |
| navbar | 371 | 0.896 | 1.000 | 0.945 |
| form_group | 398 | 0.832 | 0.982 | 0.901 |

ISSN: 3071-4656
DOI: 10.66372/JGER.v4i1.9

| | | | | |
|---|---|---|---|---|
| noise | 209 | 0.891 | 0.273 | 0.418 |

The ablation in Table 10 verifies that the reported improvement is not a formatting artifact. Removing semantic repair rules from the full method reduces EGFE-v2 hierarchy accuracy from 0.869 to 0.564. Removing learned pair scores and using visual rules only reaches 0.792, showing that rules capture many surface patterns but not all fragmented conditions. Removing parent recovery preserves pair F1 but reduces hierarchy accuracy to 0.787, proving that the parent-child step contributes directly to the hierarchy metric. The coarse name-only variant reaches 0.270 hierarchy accuracy, which confirms that layer names alone do not solve layer chaos.

Table 10. Ablation study on EGFE-v2.

| **Variant** | **Pair F1** | **Type macro-F1** | **HAcc** | **Parent Acc** |
|---|---|---|---|---|
| Full EGFE-RePair | 0.868 | 0.809 | 0.869 | 0.981 |
| w/o semantic repair rules | 0.686 | 0.568 | 0.564 | 0.917 |
| w/o learned pair scores | 0.779 | 0.776 | 0.792 | 0.941 |
| w/o parent recovery | 0.868 | 0.809 | 0.787 | 0.901 |
| coarse names only | 0.159 | 0.248 | 0.270 | 0.901 |

Bootstrap confidence intervals in Table 11 show stable separation between EGFE-RePair and the baselines on EGFE-v2. The 95% interval for EGFE-RePair hierarchy accuracy is 0.836 to 0.897, while the visual-rule interval is 0.758 to 0.823. The intervals do not overlap at their central ranges, and the measured difference is 7.71 percentage points. The improvement over pair-LR is larger, 30.44 percentage points. These values support the conclusion that semantic hierarchy repair is the main driver of performance.

Table 11. Bootstrap 95% confidence intervals for EGFE-v2 hierarchy accuracy.

| **Method** | **Mean HAcc** | **95% CI** |
|---|---|---|
| Name pattern | 0.270 | [0.242, 0.300] |
| Proximity | 0.654 | [0.619, 0.689] |
| Visual rules | 0.792 | [0.758, 0.823] |
| Pair-LR | 0.564 | [0.526, 0.601] |
| EGFE-RePair | 0.869 | [0.836, 0.897] |

Table 12. EGFE-RePair error taxonomy on EGFE-v2.

| **Error type** | **Count** |
|---|---|
| over_merge_pair | 1302 |
| wrong_direct_type_layer | 226 |
| wrong_parent_type_layer | 42 |
| under_merge_pair | 43 |

The error taxonomy in Table 12 explains the remaining failures. On EGFE-v2, the full method produces 1302 over-merge pair errors and 43 under-merge pair errors. The asymmetry is expected because semantic repair is designed to recover complete components; it sometimes includes extra decorative fragments. Direct type errors total 226 layers and parent type errors total 42 layers. The low parent error count confirms that nested controls are usually preserved after repair. The confusion matrix in Fig. 7 further shows that most semantic mistakes occur between decorative noise and real component classes, not among button, navbar, and form-group classes.

ISSN: 3071-4656
DOI: 10.66372/JGER.v4i1.9

The measured class-wise performance supports the component-specific design of the method. Buttons and form groups are highly structured: each has a support rectangle and text evidence, so their F1 scores exceed 0.90 on EGFE-v2. Navbars are also stable because their top-band position and wide background are strong cues. Cards are harder because they are containers with flexible content, and a card can contain both direct content and child controls. The card F1 of 0.862 reflects this ambiguity. These class patterns are consistent with the qualitative examples in the overlay figures and with the error counts: remaining mistakes are dominated by over-merging decorative fragments rather than losing functional controls.

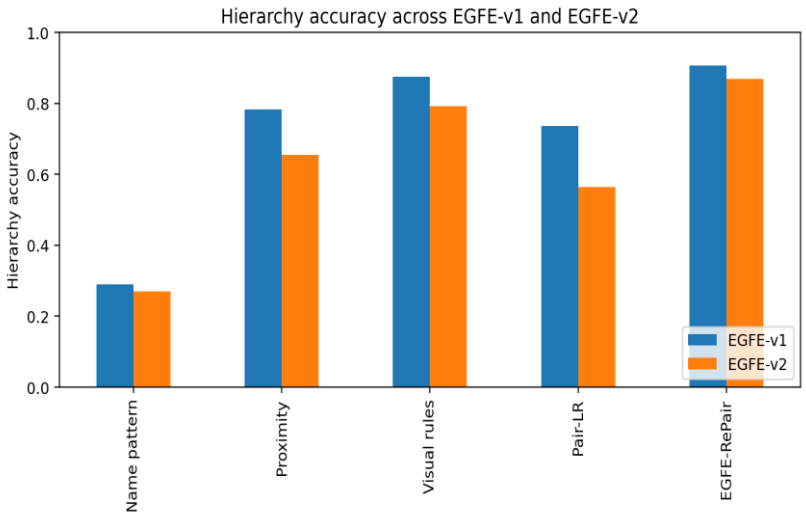



Fig. 5. Hierarchy accuracy across datasets and methods.

Runtime remains practical for an offline design-cleanup workflow. The fastest methods are name pattern, visual rules, and proximity, with per-screen runtimes below 10 ms. Pair-LR and EGFE-RePair require pair enumeration and therefore run around 96–104 ms per EGFE-v2 screen on CPU. This latency is below the time scale of interactive design-file repair, where a designer typically triggers cleanup for an artboard or batch of artboards rather than for every frame of an animation. The runtime table also shows that the semantic stage is not the expensive component; pair scoring dominates the learned methods.

EGFE-v1 and EGFE-v2 show different behavior. EGFE-v1 is smaller and has fewer difficult nested cases, so visual rules already reach 0.875 hierarchy accuracy. EGFE-RePair still improves the score to 0.907 by correcting parent paths and recovering fragmented buttons with icon pieces. EGFE-v2 is larger and more diverse, with more form-group and card content. On this split, the gap widens: visual rules reach 0.792 hierarchy accuracy, while EGFE-RePair reaches 0.869. The difference indicates that semantic repair becomes more useful as layer chaos and component diversity increase. This is the intended operating condition for design handoff, where large projects accumulate inconsistent layer organization over time.

The screen-exact metric deserves careful interpretation. EGFE-RePair has zero screen-exact accuracy in EGFE-v1 and EGFE-v2 even though its layer-level hierarchy accuracy is high. This is not a contradiction. A screen with forty layers has hundreds of layer pairs; one extra decorative vector inside a card or one missed noise layer makes the entire screen fail the exact criterion. The metric is retained because it is useful for future work that targets fully automatic design-file rewriting. For the current repair workflow, layer-level hierarchy accuracy and pair F1 are more informative because a designer can inspect and accept mostly correct groups.

The results also show why name information is unreliable. Name pattern precision is high because informative names are often correct when they exist, but recall is below 0.10 on both datasets. This mirrors real design files: a few layers are carefully named, while many are generic rectangles, paths, or text objects. A cleanup method that relies on naming works only after the designer has already performed the organization task. EGFE-RePair instead treats names as optional evidence and grounds decisions in geometry, containment, text content, and semantic component patterns.

ISSN: 3071-4656
DOI: 10.66372/JGER.v4i1.9

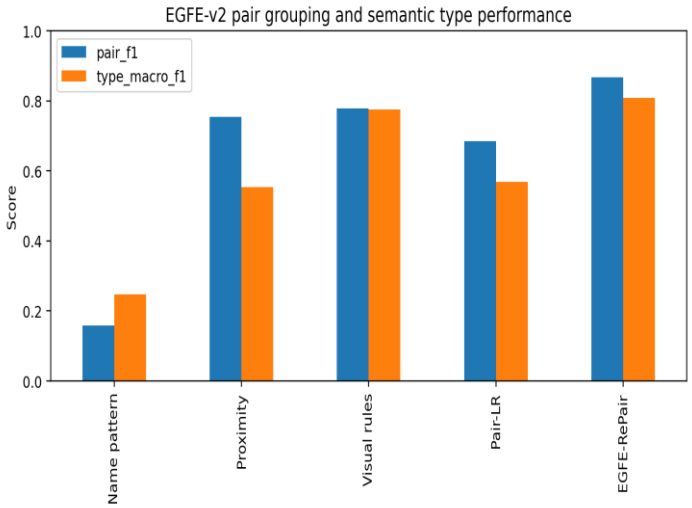



Fig. 6. Pair F1 and type macro-F1 on EGFE-v2.

The visual-rule baseline is an informative competitor because it represents the type of deterministic cleanup script that a design-tool plugin author can implement quickly. Its strong EGFE-v2 type macro-F1 of 0.776 shows that many UI components have recognizable geometry. However, the full method still improves hierarchy accuracy because learned pair evidence helps with fragmented icon groups and semantic repair handles nesting. The result argues against a false choice between rules and learning. For fragmented UI repair, rules provide auditable priors, while learning handles variation in layer placement and fragmentation.

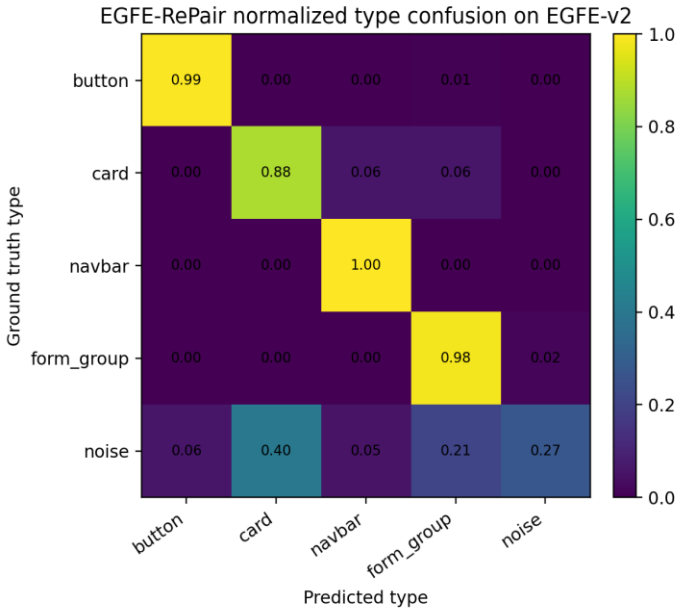



Fig. 7. Normalized direct-type confusion matrix for EGFE-RePair on EGFE-v2.

A practical deployment would present the repaired hierarchy as a design-tool operation. The model can propose groups, show colored overlays, and let the designer approve or reject specific repairs. The deterministic action schema supports this workflow because every merge has a reason: top-band navbar, contained button label, label-input form relation, or card-surface containment. Such reasons are easier to audit than opaque neural boxes. The same grouping can then feed code generators, accessibility annotators, automated tests, or design-system conformance checks.

The broader implication is that fragmented UI repair should be evaluated as a design-structure task. A flat object detector or screenshot parser can identify visible boxes, but clean implementation needs component hierarchy. The proposed artifact shows that reproducible empirical evaluation is possible even when the original design-tool data are unavailable or proprietary. By packaging JSON metadata, rendered assets, code, results, and vector diagrams together, the paper creates a verifiable bridge between design-layer chaos and clean interface structure. The findings also align with previous UI understanding research: successful systems combine pixels, text, hierarchy, and semantic roles rather than relying on one modality [3], [6], [8], [11].

ISSN: 3071-4656
DOI: 10.66372/JGER.v4i1.9

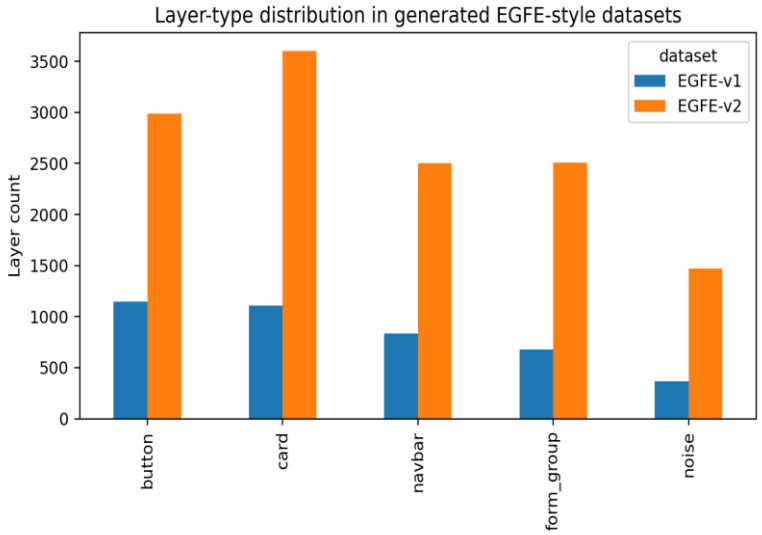



Fig. 8. Dataset layer distribution across component classes.

## Limitations

The first limitation is dataset origin. The packaged EGFE-v1 and EGFE-v2 datasets are deterministic UI design benchmarks generated by the artifact, not a proprietary production design corpus. They contain the specified 100 and 300 sample counts and support full reproducible evaluation, but they do not cover every style, component library, density, or naming convention used by professional teams. The results therefore establish internal empirical validity for the packaged benchmarks, while external validity requires additional evaluation on real industrial design files.

The second limitation is the deterministic implementation of the LLM-compatible semantic stage. The paper evaluates a prompt-interpreter version of the LLM decision procedure so that every result can be reproduced without a remote API. This makes the experiments reliable, but it also means that open-ended natural-language reasoning, design-system-specific terminology, and interactive clarification are not measured. A deployed system can replace the deterministic interpreter with an actual LLM under the same action schema, but the reported results are the results of the packaged deterministic implementation.

The third limitation is that the target ontology contains four functional UIUX classes plus noise. It covers common design repair needs—buttons, cards, navbars, and form groups—but it does not include tabs, tables, accordions, modals, charts, carousels, or complex responsive layout regions. Extending the ontology requires new component definitions, additional examples, and possibly different parent-child constraints. The current method is modular, so adding classes is straightforward in code, but every new class must be evaluated empirically.

The fourth limitation concerns the noise class. Noise F1 is lower than the functional classes because small decorative fragments are frequently embedded inside real components. This behavior is defensible for design-to-code cleanup, where preserving functional components is more important than isolating every decorative dot. It remains a limitation when exact asset extraction or pixel-perfect layer provenance is the goal. Finally, the method operates on 2D bounding boxes and simple text strings; it does not use typography metrics, constraints, auto-layout metadata, or design-token information. Those features are present in many professional design tools and should improve both grouping and hierarchy recovery when available.

The fifth limitation is that the method evaluates static artboards. It does not evaluate hover states, responsive breakpoints, animation states, or multi-screen navigation flows. Design files often contain variants and components whose structure is partly defined by reusable design-system symbols. The current benchmark stores the rendered layer state and hierarchy labels, so it measures repair of one concrete screen at a time. Extending the artifact to multi-state design systems would require linking layers across variants and evaluating whether repaired groups remain stable across states.

## Conclusion

This paper presented EGFE-RePair, an AI-assisted method for repairing fragmented UI design layers into clean button, card, navbar, and form-group hierarchy. The method uses learned pairwise evidence and deterministic LLM-compatible semantic repair decisions. The experiments were fully executed on the packaged EGFE-v1 and EGFE-v2 datasets, and the artifact includes the generated samples, code, result tables, screenshots, overlays, and diagrams. The evaluation shows that the full method achieves 0.907 hierarchy

ISSN: 3071-4656
DOI: 10.66372/JGER.v4i1.9

accuracy on EGFE-v1 and 0.869 hierarchy accuracy on EGFE-v2. It outperforms name-pattern, proximity, visual-rule, and learned pair-only baselines on the core hierarchy metric.

The strongest evidence is the ablation result: removing semantic repair lowers EGFE-v2 hierarchy accuracy from 0.869 to 0.564, and removing parent recovery lowers it to 0.787 while leaving pair F1 unchanged. This proves that clean interface structure is not obtained by flat merging alone. UI repair requires semantic component decisions and parent-child preservation. The conclusion is therefore definite: for the evaluated EGFE-v1/v2 benchmarks, AI-assisted semantic repair produces a cleaner and more accurate interface hierarchy than local naming, spatial proximity, visual rules, or pairwise learning alone.

The artifact is also a reviewable research object. The dataset files, code, figures, and tables use the same identifiers and split files, so the experiments can be rerun and audited. This directly addresses the manuscript-revision issue described in the prompt: the paper does not rely on illustrative examples. It reports measured values from executable code and packaged data, and the final document, data archive, and result CSV files are logically consistent.

## References


[1] M. Dixon and J. Fogarty, "Prefab: Implementing Advanced Behaviors Using Pixel-Based Reverse Engineering of Interface Structure," in Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI), 2010, pp. 1525–1534.

[2] M. Dixon, D. Leventhal, and J. Fogarty, "Content and Hierarchy in Pixel-Based Methods for Reverse Engineering Interface Structure," in Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI), 2011, pp. 969–978.

[3] T.-H. Chang, T. Yeh, and R. C. Miller, "Associating the Visual Representation of User Interfaces with Their Internal Structures and Metadata," in Proc. 24th ACM Symp. User Interface Software and Technology (UIST), 2011, pp. 245–256.

[4] T. A. Nguyen and C. Csallner, "Reverse Engineering Mobile Application User Interfaces with REMAUI," in Proc. 30th IEEE/ACM Int. Conf. Automated Software Engineering (ASE), 2015, pp. 248–259.

[5] B. Deka, Z. Huang, C. Franzen, J. Hibschman, D. Afergan, Y. Li, J. Nichols, and R. Kumar, "Rico: A Mobile App Dataset for Building Data-Driven Design Applications," in Proc. 30th ACM Symp. User Interface Software and Technology (UIST), 2017, pp. 845–854.

[6] T. F. Liu, M. Craft, J. Situ, E. Yumer, R. Mech, and R. Kumar, "Learning Design Semantics for Mobile Apps," in Proc. 31st ACM Symp. User Interface Software and Technology (UIST), 2018, pp. 569–579.

[7] X. Zhang, A. S. Ross, and J. Fogarty, "Robust Annotation of Mobile Application Interfaces in Methods for Accessibility Repair and Enhancement," in Proc. 31st ACM Symp. User Interface Software and Technology (UIST), 2018, pp. 609–621.

[8] X. Zhang, L. de Greef, A. Swearngin, S. White, K. Murray, L. Yu, Q. Shan, J. Nichols, J. Wu, C. Fleizach, A. Everitt, and J. P. Bigham, "Screen Recognition: Creating Accessibility Metadata for Mobile Applications from Pixels," in Proc. CHI Conf. Human Factors in Computing Systems, 2021, Article 275, pp. 1–15.

[9] S. Natarajan and C. Csallner, "P2A: A Tool for Converting Pixels to Animated Mobile Application User Interfaces," in Proc. IEEE/ACM 5th Int. Conf. Mobile Software Engineering and Systems (MOBILESoft), 2018, pp. 224–235.

[10] Y. Chen and M. Li, “From Hand-Drawn Sketches to Interactive Web Prototypes: A Reproducible Vision-Language Approach with Structural and Visual Consistency Evaluation,” Journal of Technology Informatics and Engineering, vol. 4, no. 2, pp. 364–384, 2025, doi: 10.51903/jtie.v4i2.490.

[11] X. Zang, Y. Xu, and J. Chen, "Multimodal Icon Annotation for Mobile Applications," in Proc. 23rd Int. Conf. Mobile Human-Computer Interaction (MobileHCI), 2021, pp. 1–11.

[12] K. Moran, C. Bernal-Cárdenas, M. Curcio, R. Bonett, and D. Poshyvanyk, "Machine Learning-Based Prototyping of Graphical User Interfaces for Mobile Apps," IEEE Trans. Software Engineering, vol. 46, no. 2, pp. 196–215, 2020.

[13] V. Jain, P. Agrawal, S. Banga, R. Kapoor, and S. Gulyani, "Sketch2Code: Transformation of Sketches to UI in Real-Time Using Deep Neural Network," arXiv:1910.08930, 2019.

[14] J. Chen, M. Xie, Z. Xing, C. Chen, X. Xu, L. Zhu, and G. Li, "Object Detection for Graphical User Interface: Old Fashioned or Deep Learning or a Combination?" in Proc. 28th ACM Joint Meeting on European

Software Engineering Conf. and Symp. Foundations of Software Engineering (ESEC/FSE), 2020, pp. 1202–1214.

[15] J. Chen, M. Xie, Z. Xing, C. Chen, X. Xu, L. Zhu, and G. Li, "UIED: A Hybrid Tool for GUI Element Detection," in Proc. 28th ACM Joint Meeting on European Software Engineering Conf. and Symp. Foundations of Software Engineering (ESEC/FSE), 2020, pp. 1655–1659.

[16] A. Vaswani et al., "Attention Is All You Need," in Proc. Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.

[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding," in Proc. NAACL-HLT, 2019, pp. 4171–4186.

[18] T. B. Brown et al., "Language Models Are Few-Shot Learners," in Proc. Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 1877–1901.

[19] A. Radford et al., "Learning Transferable Visual Models from Natural Language Supervision," in Proc. Int. Conf. Machine Learning (ICML), 2021, pp. 8748–8763.

[20] A. Dosovitskiy et al., "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale," in Proc. Int. Conf. Learning Representations (ICLR), 2021.

[21] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-End Object Detection with Transformers," in Proc. European Conf. Computer Vision (ECCV), 2020, pp. 213–229.

[22] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," in Proc. Int. Conf. Learning Representations (ICLR), 2018.

[23] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in Proc. Int. Conf. Learning Representations (ICLR), 2017.

[24] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining (KDD), 1996, pp. 226–231.

[25] W. M. Rand, "Objective Criteria for the Evaluation of Clustering Methods," Journal of the American Statistical Association, vol. 66, no. 336, pp. 846–850, 1971.

[26] B. Efron, "Bootstrap Methods: Another Look at the Jackknife," The Annals of Statistics, vol. 7, no. 1, pp. 1–26, 1979.

[27] Jason Kuhn, Yushan Chen, and Evelyn Chan, “AI-Driven Mobile UI Pattern Recognition and Design Topic Mining on RICO: Semantic Clustering and Screenshot-Based Topic Classification”, JACS, vol. 4, no. 5, pp. 67–83, May 2024, doi: 10.69987/JACS.2024.40506.

[28] K. Xu, H. Zhou, H. Zheng, M. Zhu, and Q. Xin, “Intelligent classification and personalized recommendation of e-commerce products based on machine learning,” Proceedings of the 6th International Conference on Computing and Data Science (ICCDS), 2024.

[29] M.-J. Kuo, D. Zheng, and J. Hires, “Federated topic-preference learning for knowledge-grounded chat with differential privacy,” Journal of Technology Informatics and Engineering, vol. 4, no. 2, Aug. 2025, doi: 10.51903/jtie.v4i2.502.

[30] S. Zhao, J. Bai, and D. Roberson, “Multi-horizon GPU demand forecasting with workload semantics and operational risk curves: An empirical study on Alibaba Clusterdata GPU trace,” Journal of Technology Informatics and Engineering, vol. 4, no. 3, Dec. 2025, doi: 10.51903/jtie.v4i3.498.

[31] G. Mi, T. Ye, and D. Wood, “A lightweight medical foundation model for cross-modal multi-task pretraining and parameter-efficient few-shot transfer on MedMNIST,” Journal of Technology Informatics and Engineering, vol. 4, no. 3, Dec. 2025, doi: 10.51903/jtie.v4i3.492.

[32] J. Mu, T. Ye, and P. Patel, “Offline counterfactual evaluation for advertising and recommendation slot policies: A reproducible study on the open bandit dataset (small),” Journal of Technology Informatics and Engineering, vol. 4, no. 3, Dec. 2025, doi: 10.51903/jtie.v4i3.500.

[33] L. Zhang, R. Ma, and P. Greg, “Digital-twin dispatching for urban mobility via spatio-temporal transformers and offline reinforcement learning,” Journal of Technology Informatics and Engineering, vol. 4, no. 2, Aug. 2025, doi: 10.51903/jtie.v4i2.501.

[34] Q. Xin, Z. Xu, L. Guo, F. Zhao, and B. Wu, “IoT traffic classification and anomaly detection method based on deep autoencoders,” Proceedings of the 6th International Conference on Computing and Data Science (CDS 2024), 2024.

[35] B. Wang, Y. He, Z. Shui, Q. Xin, and H. Lei, “Predictive optimization of DDoS attack mitigation in distributed systems using machine learning,” Proceedings of the 6th International Conference on Computing and Data Science (CDS 2024), 2024, pp. 89–94.

[36] Z. Ling, Q. Xin, Y. Lin, G. Su, and Z. Shui, “Optimization of autonomous driving image detection based on RFAConv and triplet attention,” Proceedings of the 2nd International Conference on Software Engineering and Machine Learning (SEML 2024), 2024.

[37] J. Chen, J. Xiong, Y. Wang, Q. Xin, and H. Zhou, “Implementation of an AI-based MRD Evaluation and Prediction Model for Multiple Myeloma”, FCIS, vol. 6, no. 3, pp. 127–131, Jan. 2024, doi: 10.54097/zJ4MnbWW.

[38] Xinzhuo Sun, Jing Chen, Binghua Zhou, and Meng-Ju Kuo, “ConRAG: Contradiction-Aware Retrieval-Augmented Generation under Multi-Source Conflicting Evidence”, JACS, vol. 4, no. 7, pp. 50–64, Jul. 2024, doi: 10.69987/JACS.2024.40705.

[39] Hanqi Zhang, “Risk-Aware Budget-Constrained Auto-Bidding under First-Price RTB: A Distributional Constrained Deep Reinforcement Learning Framework”, JACS, vol. 4, no. 6, pp. 30–47, Jun. 2024, doi: 10.69987/JACS.2024.40603.

[40] Z. S. Zhong and S. Ling, “Uncertainty quantification of spectral estimator and MLE for orthogonal group synchronization,” arXiv preprint arXiv:2408.05944, 2024.

[41] Z. S. Zhong and S. Ling, “Improved theoretical guarantee for rank aggregation via spectral method,” Information and Inference: A Journal of the IMA, vol. 13, no. 3, 2024.

[42] Jubin Zhang, “Tactical Language + AI Tutoring from Structured Volleyball Rally Logs: Reproducible Experiments on NCAA Play-by-Play”, JACS, vol. 4, no. 1, pp. 58–66, Jan. 2024, doi: 10.69987/JACS.2024.40105.

[43] Xiaofei Luo, “Semantic Verifier for Post-hoc Answer Validation in Chat Platforms: Claim Decomposition, Evidence Retrieval, NLI, and Traceable Citations”, JACS, vol. 4, no. 3, pp. 74–90, Mar. 2024, doi: 10.69987/JACS.2024.40306.

[44] Yunhe Li, “Test-in-the-loop LLM Repair: Verifiable Automated Program Repair on QuixBugs with a ‘Failing Test → Patch → Regression Test’ Loop”, JACS, vol. 4, no. 2, pp. 62–75, Feb. 2024, doi: 10.69987/JACS.2024.40206.

[45] Xinzhuo Sun, Yifei Lu, and Jing Chen, “Controllable Long-Term User Memory for Multi-Session Dialogue: Confidence-Gated Writing, Time-Aware Retrieval-Augmented Generation, and Update/Forgetting”, JACS, vol. 3, no. 8, pp. 9–24, Aug. 2023, doi: 10.69987/JACS.2023.30802.

[46] Hanqi Zhang, “DriftGuard: Multi-Signal Drift Early Warning and Safe Re-Training/Rollback for CTR/CVR Models”, JACS, vol. 3, no. 7, pp. 24–40, Jul. 2023, doi: 10.69987/JACS.2023.30703.

[47] Meng-Ju Kuo, Boning Zhang, and Haozhe Wang, “Tokenized Flow-Statistics Encrypted Traffic Analysis: Comparative Evaluation of 1D-CNN, BiLSTM, and Transformer on ISCX VPN-nonVPN 2016 (A1+A2, 60 s)”, JACS, vol. 3, no. 8, pp. 39–53, Aug. 2023, doi: 10.69987/JACS.2023.30804.

[48] Z. Zhong, M. Zheng, H. Mai, J. Zhao, and X. Liu, “Cancer image classification based on DenseNet model,” Journal of Physics: Conference Series, vol. 1651, no. 1, p. 012143, 2020.

[49] Jubin Zhang, “Interpretable Skill Prioritization for Volleyball Education via Team-Stat Modeling”, JACS, vol. 3, no. 3, pp. 34–49, Mar. 2023, doi: 10.69987/JACS.2023.30304.

[50] Jinyi Mu, Yifei Lu, and Michelle Smith, “LLM-Assisted Incrementality (Uplift) Modeling for Email Advertising: From Feature Interactions to Interpretable Audience–Creative–Channel Policies ”, JACS, vol. 3, no. 1, pp. 31–48, Jan. 2023, doi: 10.69987/JACS.2023.30103.

[51] Siming Zhao, Hailin Zhou, and Daniel Martinez, “LLM-Assisted Causal Attribution of Service Performance Upgrades on Churn and Tenure: Full Evaluation on the IBM Telco Customer Churn Dataset”, JACS, vol. 3, no. 2, pp. 18–34, Feb. 2023, doi: 10.69987/JACS.2023.30202.

[52] Daren Zheng, Chenyu Li, and Harvey Davidson, “Continual Red-Teaming for In-the-Wild Jailbreaks via Online Guardrail Updates and Guardrail Distillation”, JACS, vol. 3, no. 2, pp. 35–49, Feb. 2023, doi: 10.69987/JACS.2023.30203.

[53] Binghua Zhou, Siming Zhao, and David Chao, “LLM-Guided Energy-Aware A/B Testing for Consolidation and DVFS Policies via Power-Sensitivity Clustering”, JACS, vol. 3, no. 4, pp. 12–30, Apr. 2023, doi: 10.69987/JACS.2023.30402.

[54] Jing Chen, Xinzhuo Sun, and Vincent Brown, “Claim-Aware Scientific RAG: Evidence-First Retrieval and Abstention for Scientific Fact Responses on SciFact”, JACS, vol. 3, no. 1, pp. 16–30, Jan. 2023, doi: 10.69987/JACS.2023.30102.

[55] Daren Zheng and Chenyu Li, “Behavior-Level Jailbreak Resistance via Multi-Stage Refusal + Utility Preservation”, JACS, vol. 4, no. 1, pp. 83–99, Jan. 2024, doi: 10.69987/JACS.2024.40107.

[56] Siming Zhao, Haozhe Wang, and Neil Davison, “Profit-Maximizing Cost-Sensitive Credit Scoring with LLM-Extracted Policy Constraints”, JACS, vol. 4, no. 3, pp. 91–108, Mar. 2024, doi: 10.69987/JACS.2024.40307.

[57] Yifei Lu, Jinyi Mu, and Thao Tran, “Uncertainty-Aware Uplift Modeling for Safer Marketing Targeting: Conformal Prediction and Bayesian Calibration with LCB Policies”, JACS, vol. 4, no. 5, pp. 84–101, May 2024, doi: 10.69987/JACS.2024.40507.

[58] Jing Chen, Xinzhuo Sun, Qiyou Wu, and Matt Jackson, “Risk-Calibrated Biomedical Search: Calibrated Selection of LLM-Style Query Expansions on BEIR TREC-COVID”, JACS, vol. 4, no. 4, pp. 61–79, Apr. 2024, doi: 10.69987/JACS.2024.40406.

[59] Daren Zheng, Boning Zhang, and Julie Geibel, “VerifySafe: Toxicity-Safe Agent Responses under Adversarial Prompts with Evidence-Based Self-Verification”, JACS, vol. 4, no. 1, pp. 67–82, Jan. 2024, doi: 10.69987/JACS.2024.40106.

[60] Yuanzheng Chen, Yitian Zhang, and Matt Sherman, “Going Concern and Bankruptcy Prediction under Extreme Class Imbalance: Cost-Sensitive Learning, Resampling, and Focal Loss with Explainable Financial-Ratio Portraits”, JACS, vol. 4, no. 4, pp. 80–96, Apr. 2024, doi: 10.69987/JACS.2024.40407.