On Improving the Qualitative Features of the User Interface of Mobile Applications Using Machine Learning Methods

Ana Terović*, Igor Mekterović*
* University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, Croatia
terovic.ana@gmail.com igor.mekterovic@fer.hr

***Abstract*—User interfaces are among the most frequently used systems for interaction. To ease the process of creating user interfaces for designers and developers, in this paper, we aim to explore methods for improving interface design. We propose a methodology focused on improving UI through the implementation of a layout generation model. This model leverages Diffusion Layout Transformer (DLT) and is trained using comprehensive datasets such as Rico, Clay and a Huggingface dataset. The effectiveness of our model is evaluated based on its ability to generate aesthetically pleasing and functional UI layouts. We conclude the paper by discussing the implications of our findings and outlining future research directions in the automation of UI design using machine learning. The paper underscores the potential and challenges of integrating machine learning in UI design, paving the way for future advancements in automated UI layout generation.**

***Keywords*—UI, UX, Design, Machine Learning, Automated UI Generation, Layout Generation Model**

I. Introduction

User interfaces (UI) are central to interaction in the digital world, where not only the content but also the design and user experience crucially determine their overall quality. Research underscores the importance of UI design, showing its impact on user preferences [1], perceived usability [2], credibility [3], and performance [4]. Among various types of UIs, mobile interfaces are the most commonly used [5]. The visual and functional aspects of UIs are often more apparent to users than the underlying complexities, making the construction of a well-designed interface essential for a positive user experience.

Our goal is to automate enhancement of mobile UIs using machine learning, supporting designers and developers in refining existing interfaces. The integration of machine learning enables the analysis and prediction of user preferences to create more intuitive and user-friendly interfaces.

This paper comprises five sections: a review of relevant literature on UI/UX and UI enhancement methods; a detailed description of our approach, including the datasets and methodology; an analysis of our model's results followed by a discussion of our findings, their implications, and future research directions.

II. Related Work

A. Principles of UI/UX Design

User Interface (UI) and User Experience (UX) design are foundational concepts in creating digital solutions that are not only aesthetically appealing but also functionally efficient and user-friendly. The UI is the tangible bridge between the user and a digital product, be it a website, application, or any digital tool [6]. User Experience, a concept popularized by Don Norman, extends beyond the immediate interaction, encompassing all aspects of the end user's interaction with the company, its services, and its products [7]. The distinction between UI and UX is critical: while UI focuses on the momentary interactions between user and machine, UX encompasses the overall experience of using a product.

User-centered design is grounded in principles that prioritize the user's needs and experiences. Derived from experts like Jakob Nielsen [8] and Ben Shneiderman [9], these principles serve as guidelines for creating intuitive and efficient UIs. Practical guidelines rooted in Gestalt psychology further refine UI design, with concepts like alignment, composition, color usage, emphasis, the Gutenberg diagram, and Fitts's law [10].

In scientific literature, the measurement of a UI's aesthetics is most commonly defined as assessing its visual complexity. The notion of using visual complexity as a means to measure the aesthetics of a UI comes from an early paper "Modelling Interface Aesthetics" [11]. The authors break down visual complexity into smaller components like balance, symmetry, and sequence which they find all influence visual complexity.

Our study focuses on automating the evaluation and enhancement of specific UI design elements, acknowledging that the breadth of UI/UX design involves intricacies that extend beyond the scope of current automation capabilities. By balancing these principles with the capabilities of automation, we aim to develop interfaces that are technically robust and user centric.

B. Enhancement of UIs

The pursuit of automating UI design has been a focus of researchers for a long time. Initially, model-based systems like UIDE [12] and Jade [13] attempted to guide UI

generation using formal models that described tasks, data, and users. These early systems, while innovative, were often limited by their reliance on predefined templates and heuristic rules, which made them challenging to use and less effective at capturing complex design distributions. This limitation led to a pivot towards more dynamic and adaptable methods, primarily driven by advancements in deep generative models. The field has seen significant growth, particularly with the introduction of models based on Variational Autoencoders (VAE), Generative Adversarial Networks (GAN), and Transformer-based techniques.

VAEs, such as LayoutVAE [14] have shown promise in probabilistic and autoregressive layout generation. LayoutVAE uses latent variables to generate a variety of layouts from a single data point, characterized by a dual VAE structure.

GANs have also been pivotal, with models like LayoutGAN [15] using a GAN framework to synthesize semantic and geometric properties of scene elements. Its capability to handle class probabilities and geometric parameters allows them to create diverse and precise layouts.

The use of Transformers in layout generation marks another significant advancement. Due to its self-attention mechanism it allows processing of complex spatial relationships in layout design. The Layout Transformer [16] specializes in generating structured layouts, utilizing the decoder block of the Transformer model to efficiently generate layouts of arbitrary lengths.

Recent models like LayoutGAN++ [17] and GUILGET [18] represent the fusion of Transformer and GAN frameworks, emphasizing their ability to handle diverse inputs and focus on semantic relationships between elements. The diffusion layout transformer (DLT) [19] and LayoutDM [20] introduce novel approaches by employing joint continuous-discrete and modality-wise discrete diffusion processes, respectively, for layout generation. These models exemplify the evolving complexity and sophistication in automated UI design, progressively inferring noiseless layouts from initial noisy inputs.

III. Experiment Methodology

In this work, we want to improve the design of user interfaces by using machine learning methods. Our idea is to generate alternate, better versions of an existing design.

To generate a new design we have decided to use generative methods. We will be using layout generation methods, specifically we will be following the work of [19] by implementing the DLT model. A more in-depth explanation on how layout generation methods and DLT work can be found in following sections.

Our layout generation pipeline shown in Fig. 1 takes an image of an existing UI and its descriptive features as an input. They are fed to the main part of the pipeline – the layout generation model. The model then produces several alternate versions of the UI as an output. Afterward, using certain UI metrics, we will evaluate proposed alternate versions of the original UI. Only the ones that have improved the original UI based on the selected and relevant metrics will be kept and propose them to the user as the final output of the pipeline.

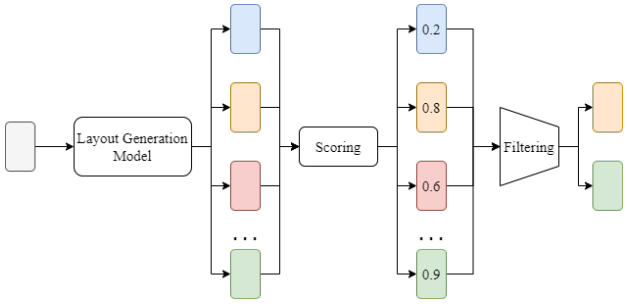

Fig. 1. Diagram representation of the pipeline

A. Layout Generation

To generate multiple versions of an UI, we have chosen to work with layout generation models. The main idea behind layout generation is to produce a new version of a layout by trying to determine the position of elements on the screen. Since we want to produce new layouts from preexisting ones, the look of the preexisting layout will determine the constraints for the newly generated layout. Specifically, we will be using an implementation of a conditional layout generative model called Joint Discrete-Continuous Diffusion Layout Transformer, DLT [19].

B. Diffusion Process

Diffusion models are a type of generative AI that are able to generate images. Their pipeline is split into two main parts: the forward and backward diffusion process. In the forward process the model gradually adds Gaussian noise to the data, making it in the end undecipherable. Then, in the backward process the model tries to learn how to decipher the image by removing the previously added noise.

More specifically, the forward process starts by sampling a data point, x_0 from the real distribution $x_0 \sim q(x)$. Then using a Markov chain, slowly through T steps, random Gaussian noise is added to the sample. For each step there is a noisy sample x_t, producing a sequence, $x_1, ..., x_T$. Approximate posterior distribution is given by (1).

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} \mathcal{N}\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I\right) \quad (1)$$

β is the variance schedule which defines the step size. As the step t becomes bigger, the original sample x_0 is no longer distinguishable. When $T \to \infty$ meaning x_0 is x_T, the sample x_T is transformed to pure Gaussian noise.

In order to run the backward diffusion process we need to learn a model p_0 to approximate conditional probabilities $q(x_{t-1}|x_t)$. If we can sample from $q(x_{t-1}|x_t)$, we can recreate a true sample of the target distribution from

Gaussian noise. Training starts with pure Gaussian noise $p(xT) := \mathcal{N}(XT, 0, I)$ where the model learns the joint distribution p_0

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \tag{2}$$

by maximizing the likelihood of training data

$$L_{\text{VLB}} = -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0) \leq \mathbb{E}_{q(\mathbf{x}_{0:T})} \Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} \Big]. \tag{3}$$

The loss function is trying to make the probability distribution P similar to reference distribution Q. Meaning, the model is trying to learn the distribution of the sample distribution. This way, the model is able to reverse the forward diffusion process and recreate true samples from Gaussian noise.

The DLT model performs both continuous and discrete diffusion as we need to determine both the coordinates of the elements on the layouts and to what category they belong to. Following sections look at some details important for both continuous and discrete diffusion.

1) Continuous diffusion

The model applies the diffusion process on a set of all bounding boxes in the layout, $x_0 \in \mathcal{R}^{N\times 4}$, where N is the maximal number of components. The model $\mathbf{F}^c_\theta(x_t, c, y_t)$ predicts x_0, as previously described, while also carrying the class information y_t. The final continuous diffusion loss, $\mathbf{L}_{bbox}$, is reweighted and simplified, as formulated by (4).

$$\mathbf{L}_{\text{bbox}} = \mathbb{E}_{(x_0,y_0)\sim q(x_0,y_0|c,t),t\in[0,1]} \left[\left\| \vec{F}^\theta_c(\vec{x}_t, c, \vec{y}_t) - \vec{x}_0 \right\|^2 \right]. \tag{4}$$

2) Discrete diffusion

For a class assignment, we need a discrete problem setting with K categories, $y \in \{1, ..., K\}^N$. For the forward denoising process, we need to use a Markov transition matrix $[Q_t]_{i,j} = q^d(y_t = i|y_{t-1} = j)$ with an absorbing state as the K-th category [Mask]. For $i < K$, the chosen matrix is:

$$[[Q_t]_{i,j} = \begin{cases} \beta_{i,j} & \text{when } i = j \\ 1 - \beta_i K \text{ ([Mask])} & \text{when } i \neq j \end{cases} \tag{5}$$

and for $i = K$:

$$[Q_t]_{K,j} = \begin{cases} 1 & \text{when } j = K \\ 0 & \text{when } j \neq K \end{cases} \tag{6}$$

If the sample is labeled as [Mask], it will not be changed. However, if it still holds a component class label, then with probability $1 - \beta$ it will be changed to [Mask], and with probability β it will remain the same. When $T \to \infty$, class labels will approximate $y_T = [Mask]^N$, meaning they will all be masked. In the reverse process,

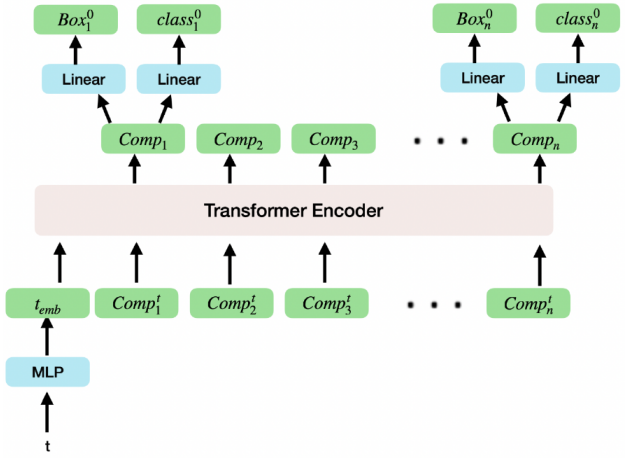

Fig. 2. DLT model scheme [19]

the model $\mathbf{F}^d_\theta(x_t, c, y_t)$ predicts class probabilities of components $\mathbf{P}^\theta(y_0|y_t, x_t, c)$. The regular cross-entropy loss is a reweighted function and formulated by (7).

$$\mathbf{L}_{cls} = \mathbb{E}_{(y_0,x_0)\sim q(y_0,x_0|c),t\in[0,1]} \mathbf{CE}(\mathbf{F}^\mathbf{d}_\theta(x_t, c, y_t), y_0) \tag{7}$$

3) Combined diffusion process

The final layout diffusion model samples both x_0 and y_0 by applying the continuous and discrete process independently. Hence the model is trained with a combined objective:

$$\mathbf{L}_{model} = \lambda_1 * \mathbf{L}_{bbox} + \lambda_2 * \mathbf{L}_{cls}. \tag{8}$$

C. Model

The full DLT model is shown in Figure 2. The model $\mathbf{F}_\theta$ uses a multi-layer Transformer encoder to learn the sample distribution. The input of the transformer is embedded time-step t, t_{emb} and a set of embedding vectors of components where each represents one of the components of a layout, $Comp^t_n$. The time embedding t_{emb} is a result of a multi-layer perceptron (MLP) network. The transformer encoder outputs an embedded representation of components, $Comp_n$. Then, these embeddings are fed to a Linear layer representing the box and class head. The output are values of bounding box coordinates and classes of components.

D. Training and Testing Datasets

For our layout generation model, it is essential to have information about the coordinates of UI components and their respective class types. To facilitate this, we identified three distinct datasets for comparison, Rico, Clay and Huggingface datasets. These datasets share a common structure: they include images of mobile applications' user interfaces, along with associated bounding boxes and classifications for the elements depicted in these images.

The RICO dataset [21], with over 66k Android app UI images, is extensive but presents challenges like complex view hierarchies and inconsistent element types. We have preprocessed it, focusing on the top 13 classes and improving bounding box accuracy. The Google Clay Dataset [22], a refined version of RICO, offers streamlined classification with 27 UI elements and more accurate bounding boxes. Additionally, the Huggingface Dataset, featuring more contemporary UIs than the Rico and Clay, simplifies its classification to six element types but risks overgeneralization and faces challenges due to its smaller size and high element density per screen.

E. Experiment settings

We set up experiment hyperparameters by following the original DLT paper [19] where they used only Rico for UI layout generation. Training was done over 800 epochs with the batch size set to 16. Learning rate started at 1e-4 and a cosine scheduler was used to adjust it over the epochs. Depending on the dataset, there were different number of class components. The transformer encoder was trained with four layers, eight attention heads, and a latent dimension of 512. For loss hyperparameters, $\lambda_1 = 5$ and $\lambda_2 = 1$ were used. After testing the number of diffusion steps for both continuous and discrete diffusion problems, it was found that $T = 100$ and $T = 10$, respectively, were the best performing options. During training, we split 85% of the datasets for training, 5% for validation, and 10% for testing. RICO and Clay dataset had a maximum of 10 components on a single layout, while Huggingface dataset could have up to 25.

When generating new alternate versions of an existing UI, we sampled $n = 20$ predictions from the model.

IV. Results and Discussion

A. Metrics

Evaluation of the layout generation model was done using 4 standard metrics used in this domain: Intersection over Union (IoU), Frechet Inception Distance (FID), alignment, and overlap. IoU computes the average area of overlap between any two bounding boxes in a layout. Meaning, it is a score we can use in reference to how diverse our results are. FID captures the similarity of generated data to real ones in the feature space. It measures the distributional distance of the generated layout to the real layout. These metrics were chosen since they are the most frequently used in layout generation and UI scoring research [17], [19], [20]. Consequently, the theory behind them is such that it makes sense that they would help detect good UI design.

B. Results

For the Rico dataset, out of 42,080 newly generated UI images, 455 (1.08%) of them were able to improve one of the metrics. For Clay this was 330 out of 23,720 (1.39%) and 190 out of 7,720 (2.46%) for the Huggingface dataset.

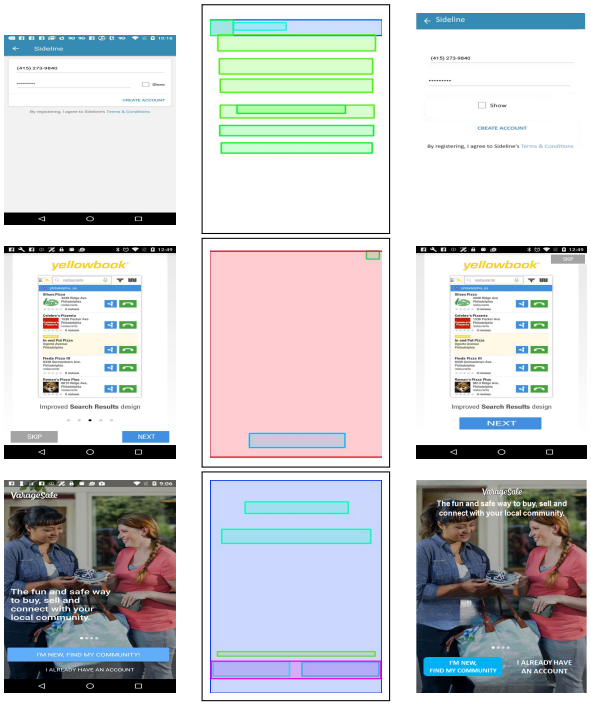

Fig. 3. Visualization of three good results. From left to right: (1) original UI, (2) generated layout, and (3) rendered image of the UI with a new layout.

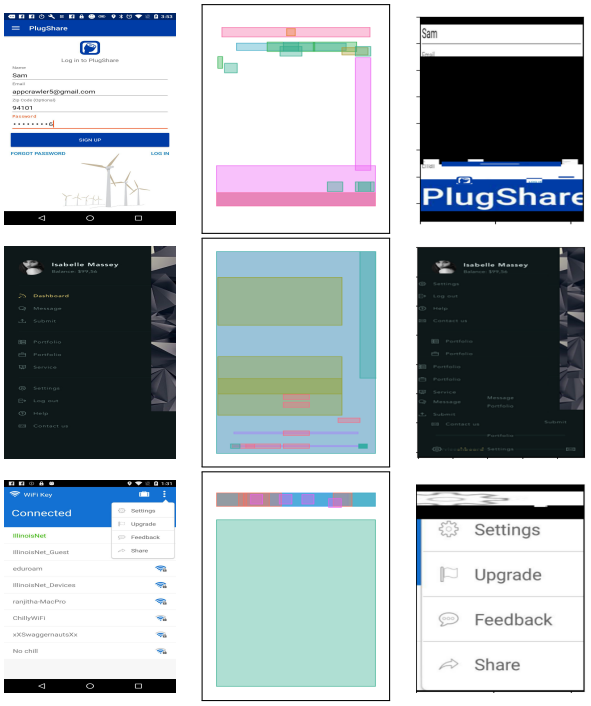

Fig. 4. Visualization of three bad results. From left to right: (1) original UI, (2) generated layout, and (3) rendered image of the UI with a new layout.

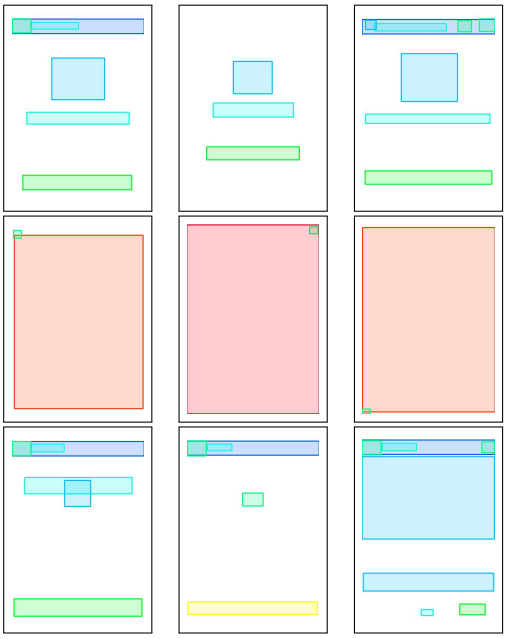

Fig. 5. Visualization of trends seen in results. Each image depicts a generated layout from a different example. From top to bottom: (1) centralized and aligned elements of similar size, (2) moving the button to different corners, and (3) top menu bar well organized.

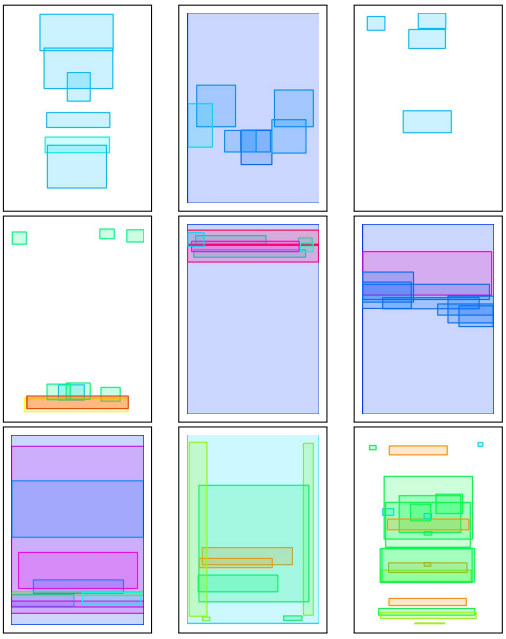

Fig. 6. Visualization of some of the models shortcomings. Each image depicts a generated layout from a different example. From top to bottom: (1) inability to handle same element type in the center of the screen, (2) grouping difficulties, and (3) poorer results with more elements.

Rico and Clay had similar findings. With more complex UIs, the model aimed for a balanced distribution of larger elements around the screen's center, often avoiding clustering. There is a noticeable pattern in element sizing: advertisements tend to be enlarged into larger rectangles, while text buttons become narrower. Text at the bottom usually retain their size and position. The most persistent issue observed was element overlap. This was particularly noticeable in UIs with numerous elements or those with clustered layouts. Occasionally, the model established a uniform size for all elements on a screen, leading to a predominance of either long narrow rectangles or nearly square shapes. This highlights the model's approach to UI design, though it also underscores the challenge of managing element overlap in dense layouts.

The Huggingface dataset was significantly smaller than Rico and Clay datasets, resulting in the model's poorer results. There are much fewer trends to be noticed and a lot less symphony between components.

C. Discussion

Our model's outputs reveal a promising ability to generate alternate versions of existing UIs as depicted in Fig. 3 and Fig. 5. These proposed layouts tend to establish more common alignment points and centralize elements, often resulting in designs with increased white space and uniformly distributed elements of similar size. These characteristics theoretically contribute to reduced visual complexity, which is widely regarded as a beneficial attribute in UI design. However, despite these theoretical advantages, our findings indicate that genuinely visually appealing generated UIs are exceedingly rare some of which can be seen in Fig. 4 and Fig. 6. This rarity is so pronounced that it becomes difficult to attribute their occurrence to any specific factor within our model's framework.

A critical phase in our process is the transition from layout generation to the actual rendering of the UI image. This step involves reshaping and repositioning original components according to the new layout predictions. While the layouts provided insights into element positioning, the rendered images highlighted the limitations of our approach. Representing a UI through just bounding boxes and component classes oversimplifies the rich and nuanced visual identities inherent in UI elements. For instance, a button labeled "+" in the bottom right corner has a specific function and established position, which our model often overlooked. Similarly, repositioning elements like exit or next buttons, marked "x" or ">", respectively, disregards their conventional placement familiar to users. These examples highlight the complexity of element positioning and sizing, which extends beyond what our current approach can adequately capture.

The methodology of this work was chosen based on significant research into the automation of UI/UX improvement. However, despite selecting what appeared to be a promising method, we have obtained poor results. While we did find some instances of UI improvement among our results, their scarcity cannot be overlooked. Upon re-examining the original papers, we noticed that they showcased only good examples without mentioning of their rarity, leading us to question the representativeness of these results for the overall model. Another way to evaluate the model's performance was by focusing on the metrics. We were able to replicate results from [19], and the chosen

method established benchmarks for IOU, alignment, and FID metrics. Despite achieving metrics comparable to established benchmarks, this did not necessarily translate into the creation of aesthetically pleasing or functionally superior UIs. Hence, we urge the field to strive for a more transparent and accurate depiction of outcomes.

The datasets used in our study also pose significant challenges. While they provide basic information on element position, size, and type, they often inaccurately represent the actual UI. The critical need for high-quality, extensive datasets in UI and UX optimization is evident. Our datasets each had notable shortcomings, the biggest one being the size. A larger and more diverse dataset, encompassing a broader range of class types and adequately representing each, might enable a more effective application of layout generation for UI improvement.

A step in the right direction of improving UIs with machine learning could be by using image diffusion models. As discussed, in order to generate visually appealing UIs, the model needs more information and should develop a more complex representation of the problem. Instead of representing the UIs with bounding boxes, class types and coordinates, perhaps detailed text descriptions could offer more insight. A recent paper utilized text descriptions of UIs to help their diffusion model learn how to generate a UI [23]. Such models, which incorporate richer contextual information, might offer a more effective representation of UIs.

V. Conclusion

This paper represents a step towards understanding and enhancing user interfaces through the lens of machine learning. Our research has demonstrated the potential of using layout generation diffusion models to create alternative and improved versions of existing user interfaces. This new approach in the field of UI design shows signs of becoming a significant tool for interface enhancement. However it is also apparent that the field is at it's beginnings.

The challenge of accurately and effectively representing and enhancing a UI might require a more nuanced and comprehensive approach than currently employed methods. As the saying goes "an image is worth a thousand words", so perhaps trying to somehow represent an image of an UI in anything less than a thousand words is too simple of an approach. Future research in this area would benefit from exploring more intricate approaches, taking into account the multifaceted nature of user interface design and user experience.

This work contributes to the broader understanding of how machine learning can be effectively integrated into the field of UI design. As technology continues to advance and user expectations evolve, we hope that the insights and methodologies presented in this paper will help gain an understanding of what is needed for advancing the automation of user interface design.

References

[1] K. O. A. N. Tuch, J. a. Bargas-Avila, "Symmetry and aesthetics in website design: It's a man's business," *Comput. Human Behav.*, 2010.

[2] J. S. A. Sonderegger, "The influence of design aesthetics in usability testing: effects on user performance and perceived usability." *Appl. Ergon.*, 2010.

[3] M. C. F. Alsudani, "The effect of aesthetics on web credibility," *British Computer Society*, 2009.

[4] D. R. S. C. Salimun, H. C. Purchase, "The effect of aesthetically pleasing composition on visual search performance," *Conf. Human-Computer Interact.*, 2010.

[5] statcounter. (2024) Desktop vs mobile vs tablet market share worldwide. [Online]. Available: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet

[6] W. C. Wang and L. Moutinho, *Human-Computer Interface in Marketing*. John Wiley Sons, Ltd, 1 2015. [Online]. Available: http://doi.wiley.com/10.1002/9781118785317.weom090132

[7] D. Norman and J. Nielsen, "The definiton of user experience (ux)," 2021. [Online]. Available: https://www.nngroup.com/articles/definition-user-experience/

[8] W. Lidwell, K. Holden, and J. Butler, *Univerzalna načela dizajna*. Rockport Publishers, Inc., 2010.

[9] B. Shneiderman, "The eight golden rules of interface design," 2016. [Online]. Available: https://www.cs.umd.edu/users/ben/goldenrules.html

[10] N. Gaal, "Ux psychology go hand in hand— how gestalt theory appears in ux design," 2017. [Online]. Available: https://uxdesign.cc/ux-psychology-go-hand-in-hand-how-gestalt-theory-appears-in-ux-design-18b7273

[11] D. C. L. Ngo, L. S. Teo, and J. G. Byrne, "Modelling interface aesthetics," *Information Sciences*, vol. 152, pp. 25–46, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025502004048

[12] W. C. Kim and J. D. Foley, "Providing high-level control and expert assistance in the user interface presentation design," ser. CHI '93. New York, NY, USA: Association for Computing Machinery, 1993, p. 430–437. [Online]. Available: https://doi.org/10.1145/169059.169346

[13] B. V. Zanden and B. A. Myers, "Automatic, look-and-feel independent dialog creation for graphical user interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '90. New York, NY, USA: Association for Computing Machinery, 1990, p. 27–34. [Online]. Available: https://doi.org/10.1145/97243.97248

[14] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2022.

[15] J. Li, J. Yang, A. Hertzmann, J. Zhang, and T. Xu, "Layoutgan: Generating graphic layouts with wireframe discriminators," 2019.

[16] K. Gupta, J. Lazarow, A. Achille, L. Davis, V. Mahadevan, and A. Shrivastava, "Layouttransformer: Layout generation and completion with self-attention," 2021.

[17] J. Li, J. Yang, A. Hertzmann, J. Zhang, and T. Xu, "Layoutgan: Synthesizing graphic layouts with vector-wireframe adversarial networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, pp. 1–1, 01 2020.

[18] A. Sobolevsky, G.-A. Bilodeau, J. Cheng, and J. L. C. Guo, "Guilget: Gui layout generation with transformer," 2023.

[19] E. Levi, E. Brosh, M. Mykhailych, and M. Perez, "Dlt: Conditioned layout generation with joint discrete-continuous diffusion layout transformer," 2023.

[20] S. Chai, L. Zhuang, and F. Yan, "Layoutdm: Transformer-based diffusion model for layout generation," 2023.

[21] 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 *Proceedings of the 30th Annual Symposium on User Interface Software and Technology*, ser. UIST '17, 2017.

[22] G. Li, G. Baechler, M. Tragut, and Y. Li, "Learning to denoise raw mobile ui layouts for improving datasets at scale," 2022.

[23] J. Wei, A.-L. Courbis, T. Lambolais, B. Xu, P. L. Bernard, and G. Dray, "Boosting gui prototyping with diffusion models," in *2023 IEEE 31st International Requirements Engineering Conference (RE)*. IEEE, Sep. 2023. [Online]. Available: http://dx.doi.org/10.1109/RE57278.2023.00035