

Boosting GUI Prototyping with Diffusion Models

Jialiang Wei*, Anne-Lise Courbis*, Thomas Lambolais*,
Binbin Xu*, Pierre Louis Bernard** and Gérard Dray*

*: EuroMov Digital Health in Motion, Univ Montpellier, IMT Mines Ales, Ales, France

**: EuroMov Digital Health in Motion, Univ Montpellier, IMT Mines Ales, Montpellier, France

* : firstname.lastname@mines-ales.fr ** : firstname.lastname@umontpellier.fr

Abstract—GUI (graphical user interface) prototyping is a widely-used technique in requirements engineering for gathering and refining requirements, reducing development risks and increasing stakeholder engagement. However, GUI prototyping can be a time-consuming and costly process. In recent years, deep learning models such as Stable Diffusion have emerged as a powerful text-to-image tool capable of generating detailed images based on text prompts. In this paper, we propose UI-Diffuser, an approach that leverages Stable Diffusion to generate mobile UIs through simple textual descriptions and UI components. Preliminary results show that UI-Diffuser provides an efficient and cost-effective way to generate mobile GUI designs while reducing the need for extensive prototyping efforts. This approach has the potential to significantly improve the speed and efficiency of GUI prototyping in requirements engineering.

I. INTRODUCTION

The exponential growth of mobile technology and the increasing dependence on mobile devices for various daily activities have significantly influenced the design and development of mobile applications. The mobile GUI (graphical user interface) plays a critical role in mobile applications since it is the primary means of interaction between users and their devices. A well-designed mobile UI can significantly enhance the user experience, making it simpler for users to navigate and achieve their desired tasks, resulting in increased user engagement and retention [1, 2]. Additionally, an engaging and user-friendly GUI can set an application apart from its competitors and increase its chances of success in the highly competitive mobile app market [3]. As the competition among mobile apps intensifies, it's critical for developers to create innovative and user-friendly GUI to meet users' evolving expectations.

GUI prototyping is a crucial technique that allows developers to create an initial version of a GUI design, assess its effectiveness, and refine it based on feedback from stakeholders. This technique is highly valuable in the context of requirements engineering, as it can help refine requirements, reduce development risks, and promote stakeholders' engagement [4]. Despite the benefits of GUI prototyping, it can be a time-consuming and costly process [5].

To enhance the GUI prototyping process, various approaches have been proposed, including the use of established tools that provide basic components and templates for creating GUIs. The industry has widely embraced commercial tools like Figma [6], InVision Studio [7], Adobe XD [8], Moqups

[9], Sketch [10], as well as open source tools like Pencil Project [11] to streamline the prototyping process. In recent literature, several GUI search and retrieval approaches have been proposed [12–19]. These approaches aim to provide users with the ability to search for inspiration from existing designs and reuse them to streamline the GUI prototyping process, which can reduce the time and effort spent on creating new designs.

Recently, deep learning-based text-to-image models have emerged as a promising approach to generate highly detailed and structured images based on text descriptions [20–23]. Among these models, Stable Diffusion [21] has shown impressive results in generating high-quality images from textual input. In this study, we propose a novel approach called UI-Diffuser that leverages the Stable Diffusion model to generate mobile UI designs through simple text prompts and UI components, as illustrated in Figure 1. The preliminary results of UI-Diffuser show its potential to enhance the effectiveness of mobile UI design and to decrease time consumption for prototyping, leading to cost reductions for this phase.

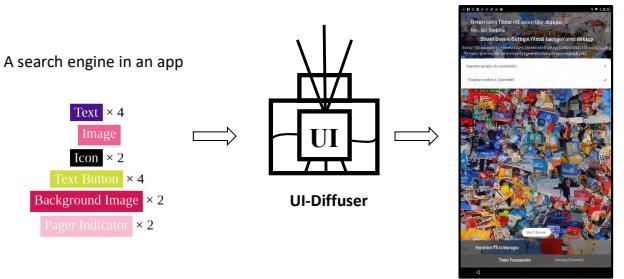


Fig. 1. UI image generation with UI-Diffuser

The rest of the paper is organized as follows: Section II presents the background of this research. Section III details the framework of UI-Diffuser. Section IV shows the samples of generated UI images and discusses their limitations. Section V concludes the paper and outlines the future directions.

II. BACKGROUND

A. GUI Prototyping

To streamline and enhance the process of GUI prototyping, numerous strategies have been previously introduced.

Prototyping tools that provide basic components and templates are widely used in practice, such as Figma [6], InVision

Studio [7], Adobe XD [8], Moqups [9], Sketch [10], and Pencil Project [11]. However, utilizing these tools effectively requires users to possess design experience.

Some researchers introduced GUI retrieval approaches that take the sketch (GUIfetch [12], SWiRE [13], Wireframe-Based UI Design Search of Chen et al. [14]) or the screenshot (Screen2vec [15], VINS [16]) as input and find designs that are similar to the input. Although useful, these methods require an initial rudimentary GUI prototype. Moreover, while these strategies primarily aim to support the GUI design’s prototyping by presenting alternative designs, they are not ideal for interactive GUI prototyping during requirements elicitation.

GUI search engines, such as Guigle [17], Gallery D.C. [18], and RaWi [19], allow users to search for existing GUI designs and components using textual queries. Wei et al. [24] have put forth an proposition of combining app features with app UIs. While these approaches can serve as a source of inspiration and guidance, blindly copying existing designs without proper attribution or permission can result in legal issues such as copyright infringement.

GUIGAN, as introduced by Zhao et al. [25], leverages previously collected GUI components from existing mobile app to compose new designs. These new composite designs not only comply with accepted standards of GUI structure, but also cater to consumer aesthetics. However, it is important to note that this approach offers limited control over the generation process.

In summary, these existing approaches fall short in providing support to analysts during the requirements elicitation phase through GUI prototyping.

B. Image synthesis

Image synthesis is a process of creating new images from diverse forms of image descriptions, including textual descriptions, sketch images, noise, and others.

In 2014, Goodfellow et al. [26] introduced Generative Adversarial Network (GAN) as a means of generating realistic image. GANs consist of two deep neural networks: a generator and a discriminator. The generator creates new images to deceive the discriminator, which aims to distinguish real from fake images. Both networks are trained simultaneously and this process continues until the generator produces images that are indistinguishable from real images. However, the lack of diversity and the challenges associated with training GANs limit their scalability and hinder their applicability to novel domains [27].

Diffusion models (DMs) [28] are neural networks trained to denoise images blurred with Gaussian noise by learning to reverse the diffusion process. Recent studies [28] have demonstrated that DMs are capable of generating high-quality images, and possess desirable attributes such as distribution coverage, a stable training objective, and scalability. Several companies recently released their image synthesis tools based on DMs like DALL-E 2 [20], Midjourney [29], Stable Diffusion [21], and DreamBooth [30]. Despite their usefulness in generating images, existing tools for image synthesis generally

lack efficacy in generating UIs. Since 2023, some researchers utilize DMs for the generation of UI layout [31–34]. However, to the best of our knowledge, no DM-based models have been developed specifically for generating UIs.

III. UI-DIFFUSER

UI-Diffuser is a novel approach that facilitates requirements engineers in rapidly prototyping mobile app UIs through a two-step process (cf. Figure 2). In the first step, UI-Diffuser takes input UI components, such as text, buttons, and images, and generates a layout that considers the arrangement of these components (see Section III-A). In the second step, the generated layout is used to complete a mobile UI image based on the textual description provided by the user (see Section III-B). In the subsequent subsections, we will describe each step of UI-Diffuser in detail.

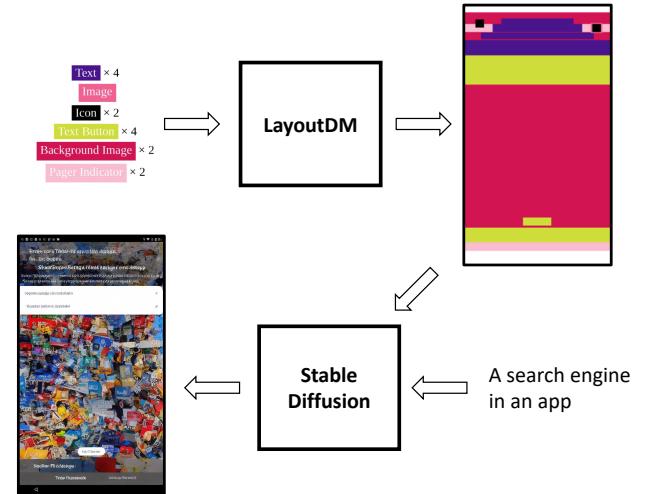


Fig. 2. Overview of UI-Diffuser

A. Layout Generation

The objective of layout generation is to create realistic graphic scenes that consist of diverse components with varying attributes, such as category, size, position, and between-component relationships [31]. This task is critical for simplifying graphic design tasks, especially for structured scenes like documents and user interfaces.

To achieve this goal, we employ LayoutDM [34] in this study. LayoutDM builds on the discrete-state space diffusion models [35, 36] and has been trained on the Rico dataset [37] – a dataset of user interface designs for mobile applications containing 25 categories of UI components, such as text button, toolbar, and icon. We will detail the Rico dataset in Section III-B2. LayoutDM allows the generation of UI layout with given conditions, such as a list of components. Figure 2 illustrates the generation of layout with a given component list.

B. UI Generation

Given the generated layout and a textual description, UI-Diffuser is able to generate UI images fitting the layout and the description.

1) *Architecture*: To generate UI images from layout and description, we utilized Stable Diffusion [21] augmented with ControlNet [38]. The proposed model workflow is illustrated in Figure 3. It contains four components: Text Encoder, Image Information Creator, Image Decoder, and ControlNet.

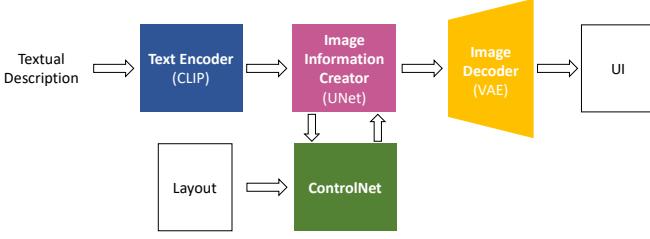


Fig. 3. Overview of UI generation model

The Text Encoder transforms raw text inputs into numerical vectors called token embeddings. CLIP [39] is a pre-trained model that has been trained on a large-scale corpus of text and images using a contrastive learning approach. CLIP is employed as text encoder due to its remarkable performance in encoding both text and images into a shared latent space.

The Image Information Creator is responsible for generating image embeddings based on the given token embeddings, which are then used by the Image Decoder to produce the final image. The *diffusion process* occurs inside this component, in a step-by-step fashion. Starting from a noisy image, each step of the *diffusion process* adds more relevant information that aligns with the input text. UNet [40] is used in this component. During the pre-training, a large number of images are blurred with Gaussian noise and the UNet is trained by denoising the image.

To support additional input conditions, such as the layout image in our case, ControlNet [38] is integrated with UNet. ControlNet is an end-to-end neural network architecture that enables the control of large image diffusion models, like Stable Diffusion, to learn task-specific input conditions. ControlNet clones the weights of the Image Information Creator into a “trainable copy”. The original Image Information Creator preserves the network capability learned from billions of images during the pre-training, while the “trainable copy” is trained on the Rico datasets to learn the conditional control. The “trainable copy” and the Image Information Creator are connected with a unique type of convolution layer called “zero convolution”. We will train the ControlNet during the fine-tuning.

Finally, the Image Decoder generates images using the image embeddings. Variational Autoencoder (VAE) [41] is a type of generative model consisting of an encoder network, which maps input images into a lower-dimensional latent space, and a decoder network, which maps the latent space

back to the original images space. The VAE encoder-decoder pair have been pre-trained on a large number of images to accurately reconstruct input images. The decoder network of VAE is used as the Image Decoder.

2) *Dataset for fine-tuning*: The fine-tuning of ControlNet [38] requires a dataset including input images, conditioning images and text prompts. In the context of UI generation, this dataset should consist of UI screenshots, wireframes that depict the page layout, and textual descriptions of the screenshots. To this end, we leveraged the Rico dataset [37], which provides the first two elements: the screenshots and wireframes. And we generated the textual descriptions of these screenshots using XUI [42].

The Rico dataset [37] is one of the largest mobile app design datasets to date, encompassing design data from more than 9.3k Android applications across 27 categories. This dataset provides access to the visual, textual, structural, and interactive design properties of more than 66k distinct UI screens. In this work, we utilized the screenshots, wireframes, and hierarchies of the Rico dataset.

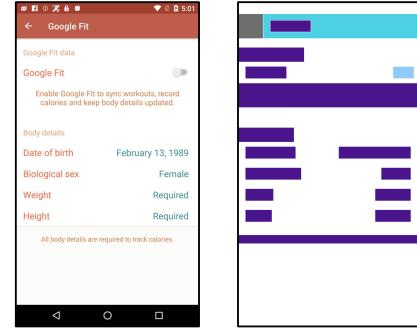


Fig. 4. Example of a screenshot and its wireframe from Rico dataset

To prepare the Rico dataset for fine-tuning the Stable Diffusion model, a preprocessing pipeline was applied to its screenshots and wireframes. The pipeline includes the following steps:

- remove the screenshots and wireframes on landscape. For this preliminary work, we focus on portrait UI design.
- resize the screenshots and wireframes to 288x512. The original size of the images is 1080x1920 or 540x960. However, the Stable Diffusion model we use accept only images with height/width less than 512 as its input during the training process.

The Rico dataset does not provide the textual description of the screenshots. For this reason, we used XUI [42] model to generate the descriptions. XUI is a tool to generate automatically informative description of a given UI screenshot. It takes the screenshots, wireframes, and hierarchies of the UIs as input, and generate their natural language descriptions. For instance, the generated description of Figure 4 is “*That screen maybe is a list screen. You may see a list of elements, typically arranged in rows. You may notice a text ubicated at the center area.*”. During the training, we randomly replace 50% text prompts with a default prompt (“A nice screenshot

TABLE I
SAMPLES OF GENERATED UI IMAGES

Components and Descriptions	Generated Layouts	Generated UIs				
<ul style="list-style-type: none"> Text × 2 Image Text Button × 4 Input × 2 <p>A login page with input fields.</p>						
<ul style="list-style-type: none"> Text × 5 Image × 7 Icon × 2 <p>A tutorial app having text components.</p>						
<ul style="list-style-type: none"> Text × 4 Image Icon × 2 Text Button × 4 Background Image × 2 Pager Indicator × 2 <p>A gallery page of an app.</p>						
<ul style="list-style-type: none"> Text Icon Text Button × 2 Input <p>A maps app.</p>						
<ul style="list-style-type: none"> Text Image × 2 Text Button Advertisement Pager Indicator <p>A mediaplayer app.</p>						
<ul style="list-style-type: none"> Text × 2 Image × 3 Text Button × 2 <p>A profile app with a big image.</p>						

of a mobile app”). This facilitates ControlNet’s capability to better recognize semantic contents from the wireframes. This is primarily because, in cases where the prompt lacks sufficient information for the Stable Diffusion model, the model may rely more heavily on the semantics of the input wireframes to compensate for the lack of meaningful guidance from the prompt.

The processed screenshots and wireframes from Rico dataset and the generated textual descriptions of XUI are then used for the subsequent fine-tuning.

3) *Fine-tuning procedures*: During the fine-tuning procedures, the parameters of Text Encoder, the Image Information Creator, and the Image Decoder are all locked. The update is only applied to the parameters of ControlNet.

Our model are implemented with the Diffusers library¹. We employed the “runwayml/stable-diffusion-v1-5” checkpoint² and the “llyasviel/sd-controlnet-seg” checkpoint³ from HuggingFace. We trained the model for 1 epoch on the processed Rico dataset with a batch size of 4, equivalent to about 16,000 steps. We use AdamW for optimization with a learning rate of $1e^{-5}$. The training was performed using Nvidia Tesla T4 with 16GB VRAM.

IV. DEMO AND DISCUSSION

Table I illustrates some samples of UI images generated by UI-diffuser. Given UI components and a brief description, UI-Diffuser generates UIs with various designs that roughly meet the requirements. UI-Diffuser can produce a UI image within a matter of seconds. Compared to traditional prototyping, UI-Diffuser can generate GUI prototypes at a much quicker rate.

At first glance, the UI images generated by UI-diffuser appear to be of high quality. However, upon closer examination, some details are missing. It’s important to note that the generated UIs may not always conform to the components category, as illustrated by the five UIs in the fifth row of Table I, where the ”advertisement” component at the bottom of the layout is disregarded. Additionally, certain generated UIs may not meet aesthetic standards.

Consequently, the current UIs produced by UI-Diffuser may be more suitable for inspiring UI designers than serving as fully functional UI prototypes.

V. CONCLUSION AND ROADMAP

This paper presents UI-Diffuser, a GUI prototyping tool that utilizes layout components and simple text prompts to produce mobile UI designs. Through a demo, we demonstrate that UI-Diffuser can generate UI images that align the given components and textual descriptions, highlighting the potential advantages of UI-Diffuser in GUI prototyping.

To advance this research, we intend to perform a comprehensive evaluation of the UI-Diffuser by investigating three critical factors: the aesthetics of the generated UIs, their compatibility with UI components, and their compatibility

with textual descriptions. The aesthetics evaluation will be carried out manually. We shall rate the aesthetics based on a predetermined set of criteria. As for the compatibility with UI components, we will manually assess the number of correctly generated UI components. Finally, we will use CLIPScore [43] to calculate the compatibility of the descriptions and their generated UIs.

Moreover, we propose enhancing UI-Diffuser from three aspects:

Developing a dataset with high-quality screenshot descriptions: the quality of the image descriptions within a training dataset has a significant impact on the performance of Stable Diffusion. In this work, we used XUI [42] to generate screenshots’ descriptions. Although XUI is a valuable tool for generating descriptions of screenshots, it only categorizes screenshots into roughly 20 categories, which is insufficient in depicting the numerous functions of modern mobile apps. Moreover, the descriptions generated by XUI lack sufficient detail in describing the UI components. To address these limitations, we plan to investigate alternative UI image captioning tools that can generate more comprehensive UI descriptions.

Cropping components from generated UIs: while generated UI images can inspire requirements engineers in GUI prototyping, they are usually not editable or directly reusable. To overcome this limitation, we take inspiration from Kolthoff et al. [19] and propose cropping each component of the generated UI image based on its absolute position in the generated layout image. The GUI components may overlap with each other, leading to a blank space of the lower component when cropping the top component. In such cases, the blank space can be filled with the top-ranked RGB color from the color histogram of the lower component. The cropped components can be then reused in further prototyping.

Generating code from generated UIs: the ability to generate code from UI designs can significantly accelerate the development of application prototypes. As the layout images generated by UI-Diffuser contain the components’ category as well as their size and position, it is possible to generate corresponding code [44]. We intend to develop a GUI code generator that compromises two steps: (i) extract location and size of each components from generated layout image, and the style from generated UI image, (ii) generate GUI code for each kind of components according to its attributes.

We believe that exploring the potential of Diffusion Models for generating UIs is a promising research direction, as it can significantly improve the speed and efficiency of GUI prototyping in requirements engineering.

REFERENCES

- [1] Q. Chen, C. Chen, S. Hassan, Z. Xing, X. Xia, and A. E. Hassan, “How Should I Improve the UI of My App?: A Study of User Reviews of Popular Apps in the Google Play,” *ACM Transactions on Software Engineering and Methodology*, vol. 30, no. 3, pp. 1–37, 2021.
- [2] S. Hassan, C. P. Bezemer, and A. E. Hassan, “Studying Bad Updates of Top Free-to-Download Apps in the Google Play Store,” *IEEE Transactions on Software Engineering*, vol. 46, no. 7, pp. 773–793, 2020.

¹<https://github.com/huggingface/diffusers>

²<https://huggingface.co/runwayml/stable-diffusion-v1-5>

³<https://huggingface.co/llyasviel/sd-controlnet-seg>

- [3] K. Moran, B. Li, C. Bernal-Cárdenas, D. Jelf, and D. Poshyvanyk, “Automated reporting of GUI design violations for mobile apps,” in *40th International Conference on Software Engineering*, 2018, pp. 165–175.
- [4] T. R. Silva, J.-L. Hak, and M. A. Winckler, “A Review of Milestones in the History of GUI Prototyping Tools,” *INTERACT 2015 Adjunct Proceedings: 15th IFIP TC.13 International Conference on Human-Computer Interaction*, pp. 267–279, 2015.
- [5] S. ul Arif, Q. Khan, and S. Gahyyur, “Requirements Engineering Processes, Tools/Technologies, & Methodologies,” *International Journal of Reviews in Computing*, vol. 2, pp. 41–56, 2010.
- [6] “Figma,” <https://www.figma.com>, accessed: 2023-04-02.
- [7] “Invision studio,” <https://support.invisionapp.com>, accessed: 2023-04-02.
- [8] “Adobe xd,” <https://helpx.adobe.com/support/xd.html>, accessed: 2023-04-02.
- [9] “Moqups,” <https://moqups.com>, accessed: 2023-04-02.
- [10] “Sketch,” <https://www.sketch.com>, accessed: 2023-04-02.
- [11] “Pencil project,” <https://github.com/evolus/pencil>, accessed: 2023-04-02.
- [12] F. Behrang, S. P. Reiss, and A. Orso, “GUIfetch: Supporting app design and development through GUI search,” *Proceedings - International Conference on Software Engineering*, pp. 236–246, 2018.
- [13] F. Huang, J. F. Canny, and J. Nichols, “SWiRE: Sketch-based User Interface Retrieval,” *Conference on Human Factors in Computing Systems - Proceedings*, pp. 1–10, 2019.
- [14] J. Chen, C. Chen, Z. Xing, X. Xia, L. Zhu, J. Grundy, and J. Wang, “Wireframe-based UI Design Search through Image Autoencoder,” *ACM Transactions on Software Engineering and Methodology*, vol. 29, no. 3, 2020.
- [15] T. J. J. Li, L. Popowski, T. M. Mitchell, and B. A. Myers, “Screen2vec: Semantic embedding of GUI screens and GUI components,” *Conference on Human Factors in Computing Systems - Proceedings*, 2021.
- [16] S. Bunian, K. Li, C. Jemmali, C. Harteveld, Y. Fu, and M. S. Seif El-Nasr, “VINS: Visual Search for Mobile User Interface Design,” in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI ’21. New York, NY, USA: Association for Computing Machinery, 2021.
- [17] C. Bernal-Cárdenas, K. Moran, M. Tufano, Z. Liu, L. Nan, Z. Shi, and D. Poshyvanyk, “Guigle: A GUI search engine for android apps,” *Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion, ICSE-Companion 2019*, pp. 71–74, 2019.
- [18] S. Feng, C. Chen, and Z. Xing, “Gallery D.C.: Auto-created GUI Component Gallery for Design Search and Knowledge Discovery,” in *Proceedings - International Conference on Software Engineering*, vol. 1, no. 1. Association for Computing Machinery, 2022, pp. 80–84.
- [19] K. Kolthoff, C. Bartelt, and S. P. Ponzetto, “Data-driven prototyping via natural-language-based GUI retrieval,” *Automated Software Engineering*, vol. 30, no. 1, p. 13, 2023.
- [20] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, “Hierarchical Text-Conditional Image Generation with CLIP Latents,” 2022. [Online]. Available: <http://arxiv.org/abs/2204.06125>
- [21] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-Resolution Image Synthesis with Latent Diffusion Models,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2022-June, pp. 10 674–10 685, 2022.
- [22] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi, “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding,” in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 36 479–36 494.
- [23] A. Q. Nichol and P. Dhariwal, “Improved Denoising Diffusion Probabilistic Models,” in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 2021, pp. 8162–8171.
- [24] J. Wei, A.-L. Courbis, T. Lambalais, P. L. Bernard, and G. Dray, “Towards Boosting Requirements Engineering of a Health Monitoring App by Analysing Similar Apps: A Vision Paper,” in *2022 IEEE 30th International Requirements Engineering Conference Workshops (REW)*, 2022, pp. 75–80.
- [25] T. Zhao, C. Chen, Y. Liu, and X. Zhu, “GUIGAN: Learning to generate GUI designs using generative adversarial networks,” *Proceedings - International Conference on Software Engineering*, pp. 748–760, 2021.
- [26] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.
- [27] P. Dhariwal and A. Nichol, “Diffusion Models Beat GANs on Image Synthesis,” in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 8780–8794.
- [28] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” *Advances in Neural Information Processing Systems*, vol. 33, no. NeurIPS 2020, pp. 6840–6851, 2020.
- [29] “Midjourney,” <https://www.midjourney.com>, accessed: 2023-04-02.
- [30] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman, “DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation,” 2022. [Online]. Available: <http://arxiv.org/abs/2208.12242>
- [31] M. Hui, Z. Zhang, X. Zhang, W. Xie, Y. Wang, and Y. Lu, “Unifying Layout Generation with a Decoupled Diffusion Model,” 2023. [Online]. Available: <http://arxiv.org/abs/2303.05049>
- [32] C.-Y. Cheng, F. Huang, G. Li, and Y. Li, “PLay: Parametrically Conditioned Layout Generation using Latent Diffusion,” 2023. [Online]. Available: <http://arxiv.org/abs/2301.11529>
- [33] E. Levi, E. Brosh, M. Mykhailych, and M. Perez, “DLT: Conditioned layout generation with Joint Discrete-Continuous Diffusion Layout Transformer,” 2023. [Online]. Available: <http://arxiv.org/abs/2303.03755>
- [34] N. Inoue, K. Kikuchi, E. Simo-Serra, M. Otani, and K. Yamaguchi, “LayoutDM: Discrete Diffusion Model for Controllable Layout Generation,” 2023. [Online]. Available: <http://arxiv.org/abs/2303.08137>
- [35] S. Gu, D. Chen, J. Bao, F. Wen, B. Zhang, D. Chen, L. Yuan, and B. Guo, “Vector Quantized Diffusion Model for Text-to-Image Synthesis,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 10 686–10 696, 2022.
- [36] Jacob Austin, Daniel D. Johnson, Jonathan H. Daniel Tarlow, and Rianne van den Berg, “Structured Denoising Diffusion Models in Discrete State-Spaces,” *Advances in Neural Information Processing Systems*, no. NeurIPS, 2021.
- [37] 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,” *UIST 2017 - Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pp. 845–854, 2017.
- [38] L. Zhang and M. Agrawala, “Adding Conditional Control to Text-to-Image Diffusion Models,” 2023. [Online]. Available: <http://arxiv.org/abs/2302.05543>
- [39] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, “Learning Transferable Visual Models From Natural Language Supervision,” in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 2021, pp. 8748–8763.
- [40] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.
- [41] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, no. MI, pp. 1–14, 2014.
- [42] L. A. Leiva, A. Hota, and A. Oulasvirta, “Describing UI Screenshots in Natural Language,” *ACM Transactions on Intelligent Systems and Technology*, vol. 14, no. 1, pp. 1–28, 2023.
- [43] J. Hessel, A. Holtzman, M. Forbes, R. Le Bras, and Y. Choi, “CLIP-Score: A Reference-free Evaluation Metric for Image Captioning,” *EMNLP 2021 - 2021 Conference on Empirical Methods in Natural Language Processing, Proceedings*, no. 2014, pp. 7514–7528, 2021.
- [44] D. De Souza Baule, C. G. Von Wangenheim, A. Von Wangenheim, and J. C. Hauck, “Recent progress in automated code generation from gui images using machine learning techniques,” *Journal of Universal Computer Science*, vol. 26, no. 9, pp. 1095–1127, 2020.