Lazy Diffusion Transformer for Interactive Image Editing – Supplementary Materials

Yotam Nitzan[1,2], Zongze Wu[1], Richard Zhang[1], Eli Shechtman[1], Daniel Cohen-Or[2], Taesung Park[1], and Michaël Gharbi[1]

[1] Adobe Research
[2] Tel-Aviv University

A Supplementary Overview

Supplementing our main paper, we include two additional files. First, we provide a **video** file comparing real-world interactions of users with *LazyDiffusion* and our baseline, *RegenerateImage*. Second, we include this PDF document containing further experiments, results, and details. In Appendix B, we conduct an ablation study, comparing our chosen architecture with possible alternatives. Then, in Appendix C, we analyze our blending approach at post-processing and extend the qualitative evaluation from the main paper. Finally, in Appendix D, we offer additional implementation details, completing the paper.

B Architecture Design and Ablation

Pivotal to our architectural design is compressing the visible context to fewer tokens and utilizing it within the diffusion decoder. In the following section, we describe the experiments leading to our eventual design.

B.1 Setting

While text-based inpainting serves as the primary application demonstrated in this paper, *LazyDiffusion* is readily applicable to a range of other local generation applications. When designing our architecture in early stages of this work, we applied our method to unconditional inpainting [15, 18] on ImageNet [2] at 256×256 resolution, as this setting demands substantially less training time and resources. We adopt the masking protocol from DeepFillV2 [18]. We use the same ViT XL/2 [3] backbone for our context encoder and adopt DiT XL/2 [10] for the diffusion transformer. Note that the PixArt-α [1] architecture, used in the main paper, is a straight-forward adaptation of DiT to support text conditioning. Consequently, the architectures we describe next can seamlessly use both as backbones.

B.2 Chosen design review

Recall that in our proposed architecture, discussed in Sec. 3, we selectively retain only encoder output tokens corresponding to the masked region, marked $\mathcal{T}_{\text{hole}}$. This ensures that downstream decoder computation scales with the mask size rather than the image size. At time t, the decoder denoises tokens $\mathcal{X}^t_{\text{hole}}$ while conditioning on the retained context tokens. We implement the conditioning by concatenating the context tokens to the noise tokens at the decoder's input. Omitted from the main paper for clarity, we prepend a linear projection layer to the diffusion transformer backbone, projecting the concatenation of tokens to the decoder's hidden dimension d. Other than the first layer, the diffusion transformer is then used *as-is* to generate $k = |\mathcal{T}_{\text{hole}}|$ tokens. Rewriting Eq. (4) from the main paper with greater detail, a single denoising step reads as

$$\mathcal{X}^{t-1}_{\text{hole}} = \text{DiT}\left(\text{linear}(\mathcal{X}^t_{\text{hole}} \oplus \mathcal{T}_{\text{hole}}); t, \mathbf{c}\right), \tag{1}$$

where $\oplus$ denotes concatenation along the hidden dimension. Transformers runtime scale quadratically with the number of tokens. Thus, the runtime of this architecture scales as $\mathcal{O}(k^2)$. In this section, we refer to this architecture as the "Concat Hidden" variant.

B.3 Alternative designs

We next describe alternative designs with the goal of ablating the two core choices – dropping visible tokens to compress context and conditioning through concatenation

Full context designs, utilizing the full set of N encoder tokens $\mathcal{T}_{\text{all}}$ as context:

- *RegenerateImage*– As described in the paper, we adapt DiT for inpainting using the GLIDE [8] conditioning approach. This model represents the common approach in local editing literature – operates on the entire canvas thus seeing the full context but also re-generating the entire image. The runtime complexity of this variant scales as $\mathcal{O}(N^2)$. Note that $N >> k$.
- *Full-Context Cross-Attention* – We add a cross-attention layer to the DiT block, between the self-attention and MLP layers. Other than the upstream activations, the cross-attention layer gets as input the *full* encoder context tokens $\mathcal{T}_{\text{all}}$. Despite "seeing" the full context, the model generates only the k masked patches. It's runtime scales as $\mathcal{O}(Nk)$.

Compressed context designs. Comparable to our chosen design – the following models utilize the masked tokens $\mathcal{T}_{\text{hole}}$ as context, generate only the masked region and have runtimes that scale with $\mathcal{O}(k^2)$. They differ in their mechanism to condition on the context tokens. We experiment with simple conditioning approaches that are applied near the input level. This prevents designs from being tightly coupled with the specific backbone architecture, which we anticipate would facilitate easier adaptation to future diffusion transformers.

– *Concat Length* – The sets of tokens are concatenated over the sequence length, rather than hidden dimension. This requires the two sets of tokens to have the same hidden dimension. To this end, we first linearly project the context tokens to the decoder's hidden dimension d. Formally, a single denoising step is done by

$$\mathcal{X}_{\text{hole}}^{t-1} = \text{DiT}\left([\mathcal{X}_{\text{hole}}^{t}, \text{linear}(\mathcal{T}_{\text{hole}})]; t, \mathbf{c}\right), \tag{2}$$

where $[\cdot,\cdot]$ represents the sequence-length concatenation.

– *Weighted Sum* – An additional weight $w \in \mathbb{R}^d$ is learned, and the input to DiT is a weighted sum of the two sets of tokens, formally

$$\mathcal{X}_{\text{hole}}^{t-1} = \text{DiT}\left(\mathcal{X}_{\text{hole}}^{t} + w * \text{linear}(\mathcal{T}_{\text{hole}}); t, \mathbf{c}\right). \tag{3}$$

– Compressed-Context Cross-Attention – We again add a cross-attention layer, but here it attends only to the reduced set of tokens $\mathcal{T}_{\text{hole}}$. To better resemble other designs in this category, incorporating the conditioning near the input, we add the cross-attention layer only to the first DiT block.

B.4 Configurations

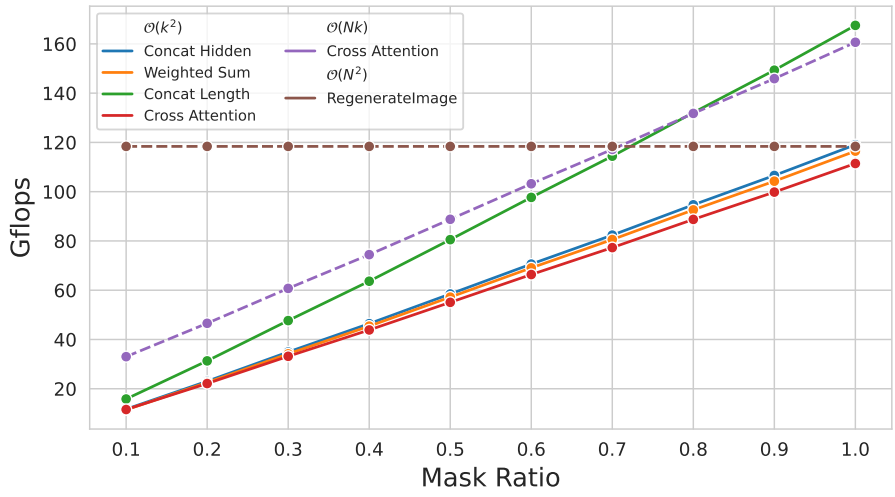
DiT's FLOPs are strongly negatively correlated with FID, across different configurations [10]. To facilitate direct comparison, we slightly adjust the XL/2 configuration for the $\mathcal{O}(k^2)$ variants so that their FLOP counts are similar. We provide the exact hyperparameters used with each variant in Tab. 1 and the resulting FLOP counts as a function of mask size are in Fig. 1a. As can be seen, *Concat Hidden*, *Weighted Sum* and the *Compressed-Context Cross-Attention* have comparable FLOPs on the entire spectrum ranging from mask ratio of 10% to 100%. For full masks, the *Concat Hidden*, *Weighted Sum* variants use 0.4% and 0.6% more FLOPs than *RegenerateImage*, respectively. This implies that our conditioning introduces negligible overhead and is well suited for using larger masks with no apparent downside. The other three variants have strictly greater FLOP counts.

B.5 Results

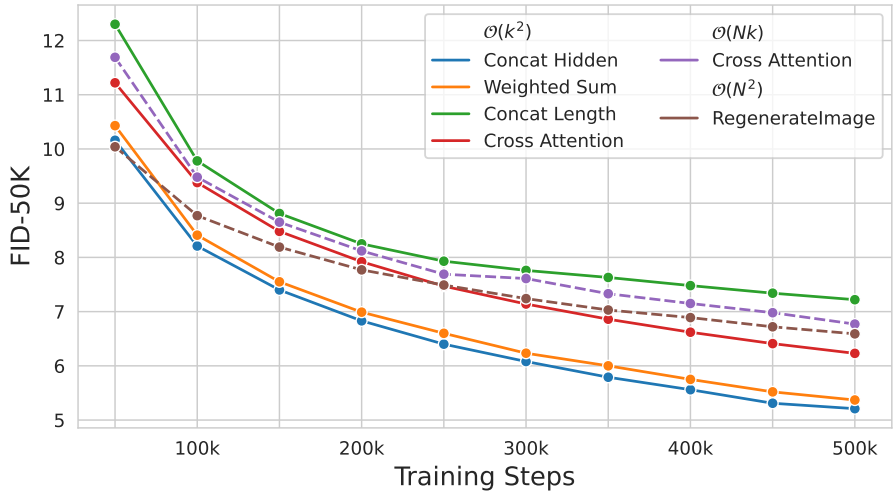
We track the FID [5] scores across 500K training iterations for all decoder designs and present the results in Fig. 1b.

Initially, we observe that "Concat Hidden" and "Weighted Sum" notably outperform all other variants. We attribute this superior performance to the explicit one-to-one context provided by these approaches. In both cases, each noise token is directly conditioned on the corresponding context token. In contrast, other methods require the decoder to extract context from a set of encoder tokens, which appears to be more challenging despite the use of positional embedding and more expressive mechanisms such as cross-attention.

Furthermore, we note that the more computationally intensive baselines, which leverage additional context, do not yield better results. Specifically, in



(a)



(b)

Fig. 1: Comparing the various architecture designs in terms of (a) FLOPs and (b) quality, measured via FID [5]. Solid lines represent variants of our approach – the encoder outputs a compressed context and the decoder generates only the masked region. Dashed lines represent mechanisms in which the decoder is conditioned on the full image context and either generates the masked region or the entire image. The latter is the approach taken by existing inpainting approaches [14]. The runtime complexities of different approaches is noted in the legend. As can be seen, conditioning each generated token directly on its corresponding compressed context token, as done for the "Concat Hidden" and "Weighted Sum" variants, leads to superior performance, despite using fewer FLOPs than competing approaches.

Table 1: Hyperparameters configuration for all architecture designs. Starting from DiT's XL/2 configuration, we slightly adapt the hyperparameters to ensure FLOP counts of $\mathcal{O}(k^2)$ are comparable.

Runtime Complexity	Model	Layers	Hidden Dimension
$\mathcal{O}(k^2)$	Concat Hidden	28	1152
	Weighted Sum	28	1152
	Concat Length	24	1024
	Cross Attention	26	1152
$\mathcal{O}(Nk)$	Cross Attention	28	1152
$\mathcal{O}(N^2)$	*RegenerateImage*	28	1152

the two cross-attention variants, the one that uses compressed context is superior to the one using full context. Our attempts to improve the performance of the *RegenerateImage* baseline by using a context encoder and a "Concat Hidden" based conditioning were futile; only dropping the visible context tokens was effective. We speculate that incorporating the full context imposes additional complexity on the decoder's task. In comparison, with *LazyDiffusion*, the information bottleneck encourages the context to be expressive but selective, allowing the decoder to "concentrate" on synthesis only.

Interestingly, in the text-conditioned setting, *LazyDiffusion* is not superior in terms of quality to *RegenerateImage*. This disparity might be explained by the lower level context required for unconditional inpainting, which primarily involves continuing surrounding textures, compared to the semantic context required for generating novel objects.

B.6 Implementation details

We train and sample all models with the EDM [6] diffusion formulation. We use Stable Diffusion's [14] public latent VAE. We train the encoder and decoder jointly from scratch, on 8 NVIDIA A100 GPUs, using global batch size of 256, using the AdamW [7] optimizer with constant learning rate of 10^{-4}. We sample using 40 denoising steps and classifier-free guidance scale of 4.0. Other details are the same as in the text-conditioned setting and are detailed in the main paper or in Appendix D.

C Additional Experiments and Results

C.1 Blending

LazyDiffusion generates only the masked regions of the latent image. To achieve the final desired results, these regions must be composited with the visible image regions and decoded into an image. Initially, we naively blend the generated latent with the latent of the input image, as described in Eq. (5) in the main

paper. However, we observe that passing the blended latent through the latent decoder $\mathcal{D}$ occasionally results in poorly harmonized images, characterized by faintly visible seams between the generated and visible regions. This phenomenon was previously noted by Zhu et al. [19] when performing local editing with Stable Diffusion [14]. It is conjectured that the latent encoding loses subtle color information, hindering image harmonization. In response, Zhu et al. proposed an alternative latent decoder that additionally conditions on the masked input image $I \odot (1-M)$ itself and is also significantly larger. Specifically, their decoder runs for 800ms, 4.5× longer than the "vanilla" Stable Diffusion latent decoder.

In our experiments, we find that simply performing Poisson blending [11] in pixel space achieves comparable results, while running only for 35ms on average. Therefore, we introduce a Poisson blending post-processing step to our pipeline. We demonstrate the harmonization issue and compare the two approaches in Fig. 2.

C.2 Additional Results

In Figs. 3 and 4, we extend Fig. 7 of the main paper and provide more qualitative samples comparing *LazyDiffusion* with the four baselines – *RegenerateCrop*, SD2-crop, *RegenerateImage* and SDXL. We find that *LazyDiffusion* is mostly comparable to *RegenerateImage* and SDXL even when inpainting objects that require high semantic context, despite using a compressed context and running up to 10× faster.

Finally, in Figs. 5 and 6 we provide a non-curated set of results, with masks and text prompts produced automatically by the segmentation and captioning models. The main challenge we observe from these results is that the model partially ignores the text when it conflicts with the shape of the mask. For example, the hamburger in Fig. 5 is generated without a hat.

D Additional Details

Evaluation. We compute FID [5] using clean-fid [9]. For CLIPScore [4], we report the "local" version that takes as input a crop around the generated object and the local text, describing the object. This approach was previously advocated by Wang et al. [16] and is more suitable for image inpainting than using the full image and text caption for the entire image.

Architecture. As described in the main paper, we initialize our decoder with PixArt-α's publicly released weights. Our decoder has an additional linear layer, introduced in Appendix B.2, that projects the concatenation of context and noise tokens to the decoder's hidden dimension d. We initialize this layer such that it outputs the noise tokens in its input and ignores the context. This ensures that at initialization, if given a full mask and thus operates on all tokens, our results are exactly equivalent to PixArt-α's.

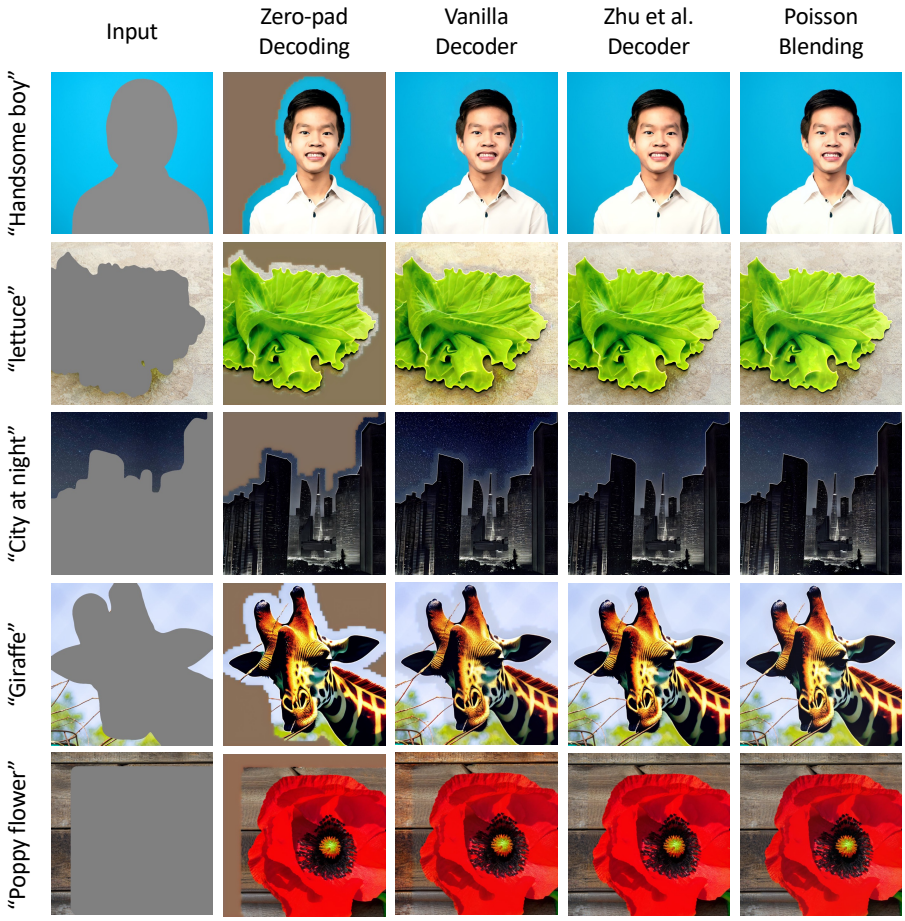


Fig. 2: From partial latent generation to inpainted image. The "zero-pad decoding" column is produced by decoding the incremental generation with zero padding, demonstrating the object in isolation. To produce the desired composited image, we blend the incremental generation with the latent input. This occasionally leads to visible seams and lack of color harmonization as seen in the "vanilla decoder" column. This issue can be solved using the latent decoder proposed by Zhu et al. [20] or with Poisson blending [11]. We recommend zooming in to better view the seams or lack thereof.

Fig. 3: Comparing inpainting results on objects that require modest context, similar to Fig. 7(Top). All models usually produce reasonably good results. Occasionally, SDXL [12] and SD2 [14] do not generate anything – a result of their usage of random masks rather than object-level masks [16, 17].

Fig. 4: Comparing inpainting results on objects that have close semantic relationship with the observed canvas, similar to Fig. 7(Bottom). Approaches that only process a crop may generate objects that appear reasonable on their own but lack coherence within the broader context of the image. In contrast, *LazyDiffusion* produces results comparable to those produces by methods regenerating the entire image. Occasionally, *LazyDiffusion* does not fully utilize the visible context. For instance, our "sushi" result accurately depicts the orange wrap and sesame seeds on top, consistent with other sushi in the roll, but it features a different filling.

Fig. 5: A random set of results produced by *LazyDiffusion*. For each input we produce three outputs from different random seeds.

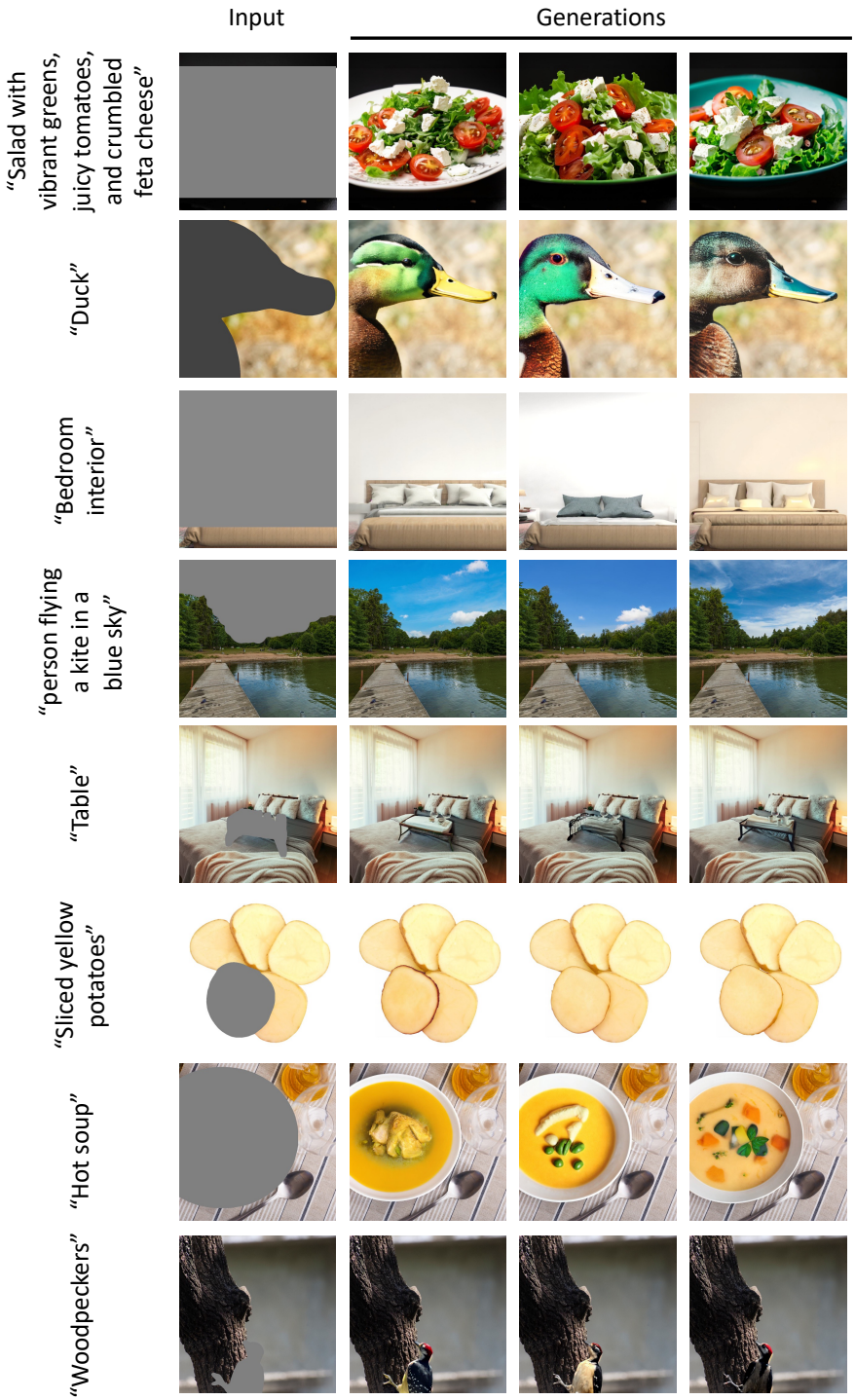


Fig. 6: A random set of results produced by *LazyDiffusion*. For each input we produce three outputs from different random seeds.

Data. As discussed in the paper, we adopt a data processing pipeline similar to that of SmartBrush [17]. Specifically, our masks are originally produced by an entity segmentation model [13] and are dilated to simulate the rough and inaccurate masks created by users. First, with probability of 20% we replace the segmentation mask with a rectangular mask corresponding to a bounding box. Regardless, we dilate the mask by first performing Gaussian Blurring and thresholding the output. The size of the Gaussian kernel is sampled uniformly from $[\text{image size}/15, \text{image size}/5]$ and its standard deviation along X and Y is sampled uniformly and independently from $[3, 17]$. The threshold is sampled uniformly from $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$.

References

1. Chen, J., Yu, J., Ge, C., Yao, L., Xie, E., Wu, Y., Wang, Z., Kwok, J., Luo, P., Lu, H., Li, Z.: Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis (2023) 1
2. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009) 1
3. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020) 1
4. Hessel, J., Holtzman, A., Forbes, M., Bras, R.L., Choi, Y.: Clipscore: A reference-free evaluation metric for image captioning. arXiv preprint arXiv:2104.08718 (2021) 6
5. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems **30** (2017) 3, 4, 6
6. Karras, T., Aittala, M., Aila, T., Laine, S.: Elucidating the design space of diffusion-based generative models. Advances in Neural Information Processing Systems **35**, 26565–26577 (2022) 5
7. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017) 5
8. Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., Chen, M.: Glide: Towards photorealistic image generation and editing with text-guided diffusion models. arXiv preprint arXiv:2112.10741 (2021) 2
9. Parmar, G., Zhang, R., Zhu, J.Y.: On buggy resizing libraries and surprising subtleties in fid calculation. arXiv preprint arXiv:2104.11222 (2021) 6
10. Peebles, W., Xie, S.: Scalable diffusion models with transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4195–4205 (2023) 1, 3
11. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. In: ACM SIGGRAPH 2003 Papers, pp. 313–318 (2003) 6, 7
12. Podell, D., English, Z., Lacey, K., Blattmann, A., Dockhorn, T., Müller, J., Penna, J., Rombach, R.: Sdxl: Improving latent diffusion models for high-resolution image synthesis. arXiv preprint arXiv:2307.01952 (2023) 8

13. Qi, L., Kuen, J., Wang, Y., Gu, J., Zhao, H., Torr, P., Lin, Z., Jia, J.: Open world entity segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence (2022) 12
14. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10684–10695 (2022) 4, 5, 6, 8
15. Suvorov, R., Logacheva, E., Mashikhin, A., Remizova, A., Ashukha, A., Silvestrov, A., Kong, N., Goka, H., Park, K., Lempitsky, V.: Resolution-robust large mask inpainting with fourier convolutions. arXiv preprint arXiv:2109.07161 (2021) 1
16. Wang, S., Saharia, C., Montgomery, C., Pont-Tuset, J., Noy, S., Pellegrini, S., Onoe, Y., Laszlo, S., Fleet, D.J., Soricut, R., et al.: Imagen editor and editbench: Advancing and evaluating text-guided image inpainting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18359–18369 (2023) 6, 8
17. Xie, S., Zhang, Z., Lin, Z., Hinz, T., Zhang, K.: Smartbrush: Text and shape guided object inpainting with diffusion model. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 22428–22437 (2023) 8, 12
18. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Free-form image inpainting with gated convolution. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4471–4480 (2019) 1
19. Zhu, Z., Feng, X., Chen, D., Bao, J., Wang, L., Chen, Y., Yuan, L., Hua, G.: Designing a better asymmetric vqgan for stablediffusion (2023) 6
20. Zhu, Z., Feng, X., Chen, D., Bao, J., Wang, L., Chen, Y., Yuan, L., Hua, G.: Designing a better asymmetric vqgan for stablediffusion. arXiv preprint arXiv:2306.04632 (2023) 7