Lazy Diffusion Transformer for Interactive Image Editing

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

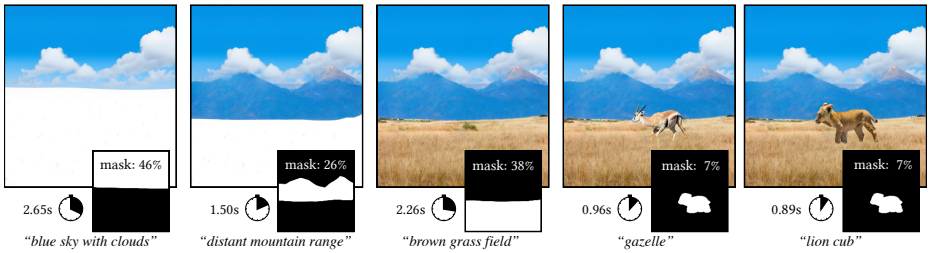

Fig. 1: Incremental image generation at 1024×1024 using *LazyDiffusion* with 20 diffusion steps. The model generates content according to a text prompt in an area specified by a mask. Each update generates *only* the masked pixels, with a runtime that depends chiefly on the size of the mask, rather than that of the image.

Abstract. We introduce a novel diffusion transformer, *LazyDiffusion*, that generates partial image updates efficiently. Our approach targets interactive image editing applications in which, starting from a blank canvas or an image, a user specifies a sequence of localized image modifications using binary masks and text prompts. Our generator operates in two phases. First, a context encoder processes the current canvas and user mask to produce a compact global context tailored to the region to generate. Second, conditioned on this context, a diffusion-based transformer decoder synthesizes the masked pixels in a "lazy" fashion, i.e., it *only* generates the masked region. This contrasts with previous works that either regenerate the full canvas, wasting time and computation, or confine processing to a tight rectangular crop around the mask, ignoring the global image context altogether. Our decoder's runtime scales with the mask size, which is typically small, while our encoder introduces negligible overhead. We demonstrate that our approach is competitive with state-of-the-art inpainting methods in terms of quality and fidelity while providing a 10× speedup for typical user interactions, where the editing mask represents 10% of the image.

1 Introduction

Diffusion models have had remarkable successes in generating high-quality and diverse images. They are the powerful engine behind exciting local image editing applications based on inpainting, where a user provides a mask and a text prompt describing a region to modify and the content to generate, respectively [33, 44, 54]. While current approaches yield impressive results, they are also slow and wasteful. Invisible to the end user, the inpainting pipeline generates an entire image and then selectively utilizes only the few pixels located within the mask, discarding all others. Although this approach is generally common in inpainting pipelines [59,60], its inefficiency is particularly pronounced with diffusion models, due to their iterative sampling procedure, precluding their usage in interactive workflows. Practitioners [38,55] save time and computation by cropping a small rectangular region around the mask, possibly downsampling for processing with the diffusion, then upsampling and blending the result to fill the hole. In doing so they compromise image quality and sacrifice the global image context, which often leads to spatially inconsistent outputs (Compare Figs. 2(a) and 2(b)).

We propose a new generative model architecture, which we call *LazyDiffusion*. Our approach, illustrated in Fig. 1, generates *partial* image updates, strictly limited to the masked region, and does so efficiently, with a cost commensurate to the mask size. Yet, its output respects the global context given by the observed canvas (Fig. 2(c)). To achieve this, our key idea is to decouple the generative process into two distinct steps. First, an encoder processes the visible canvas and mask, summarizing them into a global context code. This encoder processes the entire canvas, but it only runs once per mask. Second, conditioned on the global context and the user's text prompt, a diffusion decoder generates the next partial canvas update. This model runs many times during the diffusion process, but unlike previous works, it only operates on the masked region. Since, in practice, most updates cover small areas (10–20% of the image), this yields significant computation savings, making the editing experience more interactive.

Our encoder and diffusion decoder operate in a latent space [44], for efficiency. Both use the transformer architecture [13, 36, 53]. The transformer architecture is particularly appealing because splitting the image into small enough patches (tokens) enables generating arbitrarily-shaped regions with minimal waste. The encoder processes the entire image and mask and produces a mask-dependent context. We keep only the context tokens corresponding to the location of the masked patches. This ensures the downstream computation only scales with the size of the masked region, and encourages the compressed context to represent the relationship of the masked region to the rest of the image. At each denoising step, the decoder only processes tokens corresponding to masked patches. While the decoder *generates* only the masked region, it *"sees"* the entire image, through the compressed context, ensuring strong coherence. The conditioning on context is efficient and adds negligible computational overhead. In contrast, previous methods [1, 44, 54] achieve spatial consistency by uniformly processing all image regions, masked or not. Figure 3 illustrates the conceptual difference between our approach and a baseline diffusion transformer.

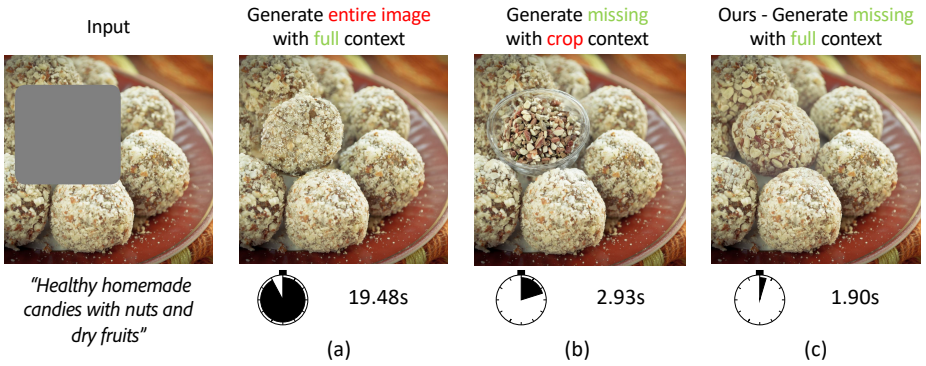

Fig. 2: Comparing inpainting approaches. (a) Most works [39, 44] generate the entire image, utilizing the full image context and fill the hole by discarding the non-masked regions. While the outcome aligns well with the image, the process is time-consuming. (b) generating only a lower resolution crop around the mask is more efficient and still seamlessly blends with nearby pixels [38, 55]. However, the inpainted content is semantically inconsistent with the overall image context. (c) our approach ensures both global consistency and efficient execution.

Our approach reduces computational cost significantly for small masks, typical in interactive editing. We achieve a speedup up to $\times 10$ over methods processing the entire image, for mask covering 10% of the image. Additionally, our model produces results of comparable quality, indicating that the compressed context is rich and expressive. In an interactive image generation context, our method amortizes the overall synthesis cost over multiple user interactions, improving interaction latency. It also amortizes the encoder cost when generating multiple updates for a given mask, using different input noise or text prompt (Fig. 1, rightmost panel).

2 Related Work

Speeding up diffusion models and inpainting. Diffusion models [21, 48, 51] are a significant breakthrough in generative modeling [2, 11, 42, 44, 45] and editing [1, 31], producing images with unparalleled quality and diversity. But they remain costly to evaluate, due to the iterative nature of their sampling process. Numerous methods have been developed to speed up their inference time by performing less denoising steps. Prominent approaches include better samplers and ODE solvers [22, 28, 29, 49], and distillation techniques [26, 30, 46, 50]. Consequently, the gap between recent one-step diffusion models [32, 39, 58] and their expensive multi-step counterparts is closing. Our approach also seeks to speed up the image synthesis process for diffusion-based models, but our contribution is largely orthogonal and can be combined with existing methods: we speed up the atomic denoising iteration, rather than performing less iterations. We achieve our speedup by generating only a subset of pixels, supporting applications per-

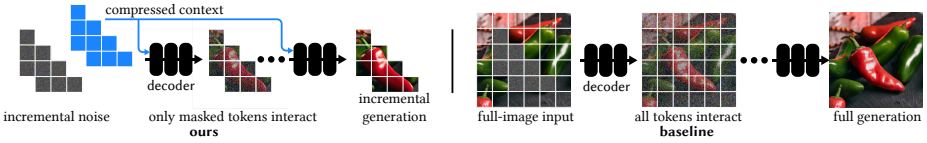

Fig. 3: Our diffusion transformer decoder (left) reduces synthesis computation using two strategies. First, we compress the image context using a separate encoder (not shown) outside the diffusion loop. Second, we only generate tokens corresponding to the masked region to generate. In contrast, typical diffusion transformers (right) [7,36] maintain tokens for the entire image throughout the diffusion process, to preserve global context. When performing inpainting, such model generates a full-size image, most of which is discarded in order to in-fill the hole region only. Existing convolutional diffusion models for inpainting [44] suffer from the same drawbacks.

forming *partial* image generation, such as inpainting. Previous works following this approach have proposed vastly different methods. Li et al. [24] reuse cached activation maps from previous generations and process only edited regions – a versatile concept that can be integrated with many existing architectures. However, the per layer and denoising step cache consumes additional GPU memory and must be recomputed when changing the text prompt. Additionally, processing only edited regions prevent high-context and long-range dependencies. CoordFill [25] do not consider diffusion models or text-based generation, but propose a pixel-wise generator architecture that uses a downsampled version of the full image for context, resulting in inferior generation quality.

Transformer-based generative models. Early transformers for image generation generate image autoregressively [8, 15, 43] in scanline order. CogView2 [12] proposes a hierarchical transformer to improve generation speed and shows application to text-guided image inpainting with rectangular masks. Later non-autoregressive models like MaskGIT [6] generate images gradually, a few tokens at a time, but they do so iteratively, generating all tokens at every iteration and discarding the unmasked ones, which is inefficient. They focus on sequential generation to improve image quality.

Our transformer-based model design is inspired by Masked Autoencoders (MAEs) [17], but we reverse their asymmetric design. Our encoder processes *all* the tokens to produce context at the masked locations, and our decoder operates on the masked tokens. Our decoder is a powerful diffusion transformer, recently proposed as an alternative to the popular UNet design [36, 52]. Most relevant to this work, DiT [36] was proposed for class-conditioned image generation and was improved in PixArt-α [7] to support text-conditioning. Our diffusion decoder is an adaptation of PixArt-α that additionally conditions on the global context produced by the encoder. Masked diffusion transformers were previously explored for representation learning [16, 56] or for minimizing training cost [61]. Our focus is on speeding inference to improve interactivity. Recent trends indicate that the transformer architecture becoming central to state-of-the-start image [14] and video generators [3], for which our method would enable faster inference and interactive applications.

Text-guided diffusion-based image editing. Text-to-image diffusion models have become the de-facto foundation for generative image editing methods. With user edits typically spatially localized, significant effort has gone into developing techniques that allow precise modifications [4, 18, 35] by selectively manipulating internal representations, *e.g.* attention maps, during the denoising process to affect only certain local regions without undesirable side-effects. Another line of work adopts the formulation of inpainting, where a mask is provided to localize the edit. Blended diffusion [1] and DiffEdit [9] use pre-trained generation models and spatially blend noised versions of the input into the gradual denoising process to enforce the preservation of unmasked regions. This indirect approach often result in artifacts, leading more recent approaches to fine-tune text-to-image models specifically for inpainting. Starting from an image generation architecture, GLIDE [33] and Stable Diffusion Inpaint [44] add mechanisms to additionally condition on the mask and masked image and fine-tune the models to predict the masked pixels. Recent advancements in this domain involve training inpainting models with object-level masks [54] rather than random ones and possibly also object-level text captions [57], mirroring real-world usage more closely. These works retrofit image generation architectures for local editing, but these models produce the full image, including regions that should not be changed. This is inefficient in time and computing resources. Our architecture efficiently performs local edits by generating only the masked region.

3 Method

Our goal is to develop an efficient diffusion generator for text-guided image editing, whose generation cost scales with the size of the region to generate, and which can incorporate the context of the entire image for a fixed, small fraction of its total cost. Starting from an image $I \in \mathbb{R}^{h \times w \times 3}$, the user specifies the region to be edited with a binary mask $M \in \{0,1\}^{h \times w}$ and text prompt $\mathbf{c}$, indicating where and what content to generate. A mask value 1 specifies a hole to inpaint, and 0 for context pixels to not touch. Unless stated otherwise, we use images of $h = w = 1024$ resolution.

Following standard practice, we operate in latent space [44], a compressed version of the RGB domain (§ 3.1). Observing that the iterative diffusion process is the computational bottleneck in state-of-the-art generators, our generator has a novel asymmetric encoder-decoder transformer architecture, as illustrated in Fig. 4. The encoder (§ 3.2) compresses and summarizes the whole image context and is only run once. The decoder (§ 3.3) is a transformer-based diffusion denoiser that is iteratively run, but only on the masked area. As such, computation cost and latency are proportional to the number of pixels to synthesize, rather than the entire canvas [1, 54, 57]. This significantly reduces computation since, for most edits, the masks are small.

3.1 Latent space processing

Following previous works of Latent Diffusion Models (LDM) [44], our model operates in an intermediate latent space of $8\times$ lower resolution with $c = 4$ chan-

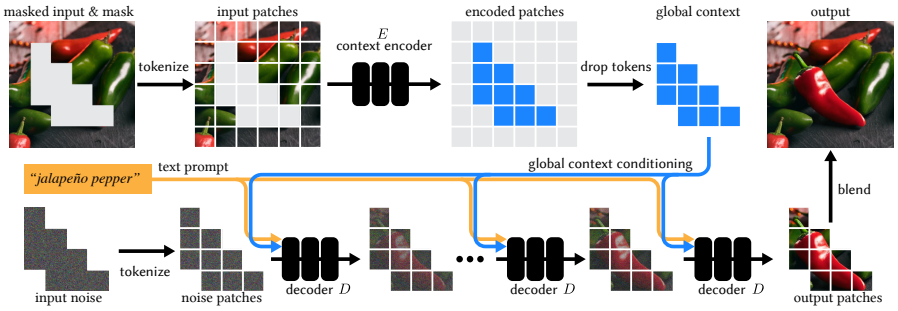

Fig. 4: Overview. To generate an incremental image update, our algorithm takes as input a user mask and a text prompt. **(top)** We start by transforming the visible pixels and binary mask into patches, and pass them to a vision transformer (ViT) encoder. We then drop all tokens, except those corresponding to the hole region; this is our global context. **(bottom)** To generate the missing pixels, we initialize a set of noise patches corresponding to the masked region and pass them through a diffusion transformer model for several denoising iterations, until we obtain denoised patches. Unlike previous works [7, 36], which process the entire image, our diffusion transformer only processes the patches required to cover the missing region. We train our encoder and diffusion decoder jointly using a diffusion denoising objective on the missing patches. The generated patches are then blended back into the missing region to produce the final output. Our model operates in a pretrained latent image space [44], but we illustrate our pipeline with RGB images for simplicity.

nels, which reduces computation without significantly impacting visual quality. We use the pretrained latent VAE of Stable Diffusion [44], denoting the encoder and decoder $\mathcal{E}$ and $\mathcal{D}$, respectively. We encode the masked image as our latent input [54]:

$$Z = \mathcal{E}\left(I \odot (1 - M)\right) \quad \in \mathbb{R}^{\frac{h}{8} \times \frac{w}{8} \times c}, \tag{1}$$

where $\odot$ represents element-wise multiplication across the spatial dimensions.

3.2 Global context encoder

Encoder E processes the *whole* image, with the goal of efficiently encoding the information given by the visible region, so that a downstream decoder can synthesize a visually consistent output with the context. Our encoder E is a Vision Transformers (ViT) [13]. To produce tokens, we first downsample the mask M using a learned convolution layer to match the latent spatial dimensions, as done by Wang et al. [54]. Then, we concatenate the downsampled mask and latent code Z along the channel dimensions and and divide them into 4×4 patches, with an overlap of 1 on each side. This yields $N = 64 \times 64 = 4096$ patches. Then, following standard practice, we linearly embed each patch and add positional embedding [53]. Finally, the tokens are passed through the transformers and produce a new set of N tokens. In summary, the encoder transforms the input Z and M into a set of N tokens of dimension $d = 1152$.

$$\mathcal{T}_{\text{all}} = \{\tau_1, \tau_2, \ldots, \tau_N\} = E(Z, M),\ \tau_i \in \mathbb{R}^d. \tag{2}$$

Token dropping. The set of output tokens contain information regarding the whole image, but using them all would cause downstream computation to scale with respect to the input size. *Can we instead keep only a subset of tokens, that would hold the information needed for generation?*

As the self-attention layers in the encoder transformer enable all the tokens to interact, each individual token has the potential to encode the relevant context of the whole image. As such, we discard the tokens corresponding to the visible region, keeping the ones corresponding to the hole. Dropping tokens outside the mask creates an information bottleneck that encourages E to summarize the input context in a compact set of tokens and ensures the downstream computation only scales with the size of the masked area, since the decoder will thus only process tokens covering the hole. The tokens should also represent the relevant information for the given location; previous works visualizing transformers [5] suggest that this location information can be preserved. Patches with partial holes are also included, and the visible pixels in those patches are blended in at the output step. Formally, we maxpool mask M to a 64×64 map and vectorize into a set $\{m_i\}_{i=1}^{4096}$, where $m_i \in \{0, 1\}$.

$$\mathcal{T}_{\text{hole}} = \{\tau_i \mid m_i = 1\} \subseteq \mathcal{T}_{\text{all}}. \tag{3}$$

The remaining set of $N_{\text{hole}} \leq N$ tokens form our compressed global context. This design, along other architectural choices, are evaluated in the supplemental.

3.3 Incremental diffusion decoder

We synthesize the missing pixels, using a transformer-based diffusion decoder D [7, 36]. Rather than keeping a set of N tokens representing the whole image, we start with N_{hole} tokens corresponding to the hole, $\mathcal{X}_{\text{hole}} = \{\mathbf{x}_i\}$. The diffusion process creates time-conditioned tokens $\mathcal{X}^t_{\text{hole}} = \{\mathbf{x}^t_i\}$, where $t \in [0, ..., T]$, starting at time T with features drawn from a unit Gaussian. The decoder progressively denoises these tokens, conditioned on the T5-encoded text prompt $\mathbf{c}$ [41] and the global context produced by the encoder $\mathcal{T}_{\text{hole}}$:

$$\mathcal{X}^{t-1}_{\text{hole}} = D\left(\mathcal{X}^t_{\text{hole}} \oplus \mathcal{T}_{\text{hole}}; t, \mathbf{c}\right), \tag{4}$$

where $\oplus$ denotes concatenation along the hidden dimension of corresponding elements in each set. We find this conditioning mechanism superior to several alternatives analyzed in Appendix B.

Blending. The final tokens $\mathcal{X}^0_{\text{hole}}$ are mapped back into the latent image domain using a linear layer, and the inverse of the patch-splitting procedure to obtain a partial latent image $\hat{Z}_{\text{hole}} \in \mathbb{R}^{\frac{h}{8} \times \frac{w}{8} \times c}$. The missing tokens, corresponding to visible pixels, are left uninitialized with zeros. We combine this output with the visible latent, using pointwise masking, to obtain the final latent composite:

$$\hat{Z} = (1 - M) \odot Z + M \odot \hat{Z}_{\text{hole}}. \tag{5}$$

Finally, this is decoded by the latent decoder to produce the final RGB image $\hat{I} = \mathcal{D}(\hat{Z})$.

These decoded results occasionally contains faintly visible seams. Previous works performing inpainting with latent diffusion models observed this phenomenon and addressed it with a dedicated latent decoder [62]. As their decoder is computationally intensive, we opt to use a simple Poisson blending postprocessing step [37] in RGB space. We discuss this challenge in greater length in the supplemental.

Training and implementation details. The encoder and decoder models are trained jointly, end-to-end, to reconstruct masked (latent) pixels, using the Improved DDPM objective [34]. For the decoder, we adopt the PixArt-α [7] architecture, and add a single layer to support our conditioning on context. We initialize all shared layers from the public PixArt-α checkpoint to benefit from their pretraining. The encoder on the other hand, is trained from scratch. We train our model for 100,000 iterations on 56 NVIDIA A100 GPUs, using the AdamW optimizer [27], with a constant learning rate 2×10^{-5}, weight decay set to 3×10^{-2} and global batch size of 224. We use $T = 1000$ diffusion steps during training. We generate our results using the Improved DDPM sampler [34] with 50 steps, unless specified otherwise, and set the classifier-free guidance scale to 4.5. All running times are measured on a single A100 GPU. We provide further details in Appendix D.

4 Experiments

4.1 Experimental setup

The main paper primarily focuses on a text-conditioned setting, as do the experiments that follow. However, our approach is versatile and can be applied in other use cases as well. In the early stages of this research, we primarily explored unconditional inpainting on the ImageNet dataset [10], which are detailed in Appendix B.

Dataset. We train our model at 1024×1024 resolution on an internal dataset containing 220 million high-quality images, covering a wide variety of objects and scenes. We produce masks and text prompts in a process similar to that proposed by Xie et al. [57]. Specifically, we use an entity segmentation model [40] to segment all objects in an image and then caption each entity with BLIP-2 [23]. To simulate the rough and inaccurate masks created by users, we randomly dilate the entity mask (see Appendix D for details). During training, we randomly sample triplets of image, mask, and caption.

Baselines. We compare *LazyDiffusion* with two inpainting baselines (already shown in Fig. 2), which we refer to as *RegenerateImage* and *RegenerateCrop*. *RegenerateImage*, is the approach found in most academic works [39, 44, 54, 57], and operates on the entire image. *RegenerateCrop*, used in popular software frameworks [38, 55], operates on a tight square crop around the masked region. The crop is first resized to a fixed low-resolution before processing and is upsampled back afterwards. Both approaches generate as many pixels as their input

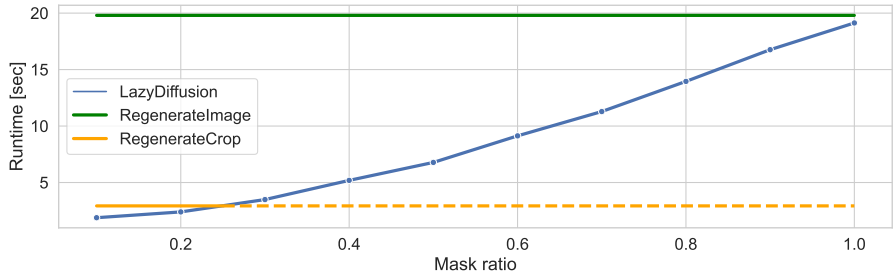

Fig. 5: Comparing *LazyDiffusion*'s runtime to that of baselines regenerating the entire 1024×1024 image or a smaller 512×512 crop around the mask. *LazyDiffusion* is consistently faster than *RegenerateImage*, especially for small mask ratios typical to interactive edits, reaching a speedup of $10\times$. Similarly, *LazyDiffusion* is faster than *RegenerateCrop* for mask ratios $< 25\%$. For masks greater than that (dashed), *RegenerateCrop* is technically faster but generates in low-resolution and naively upsamples to match the desired resolution, harming image quality.

contains (whether full-canvas or local crop), unlike *LazyDiffusion* that generates only masked patches.

To ensure a fair comparison, we utilize the PixArt-α architecture for both approaches. Since there is currently no publicly available PixArt-based inpainting models, we design and train them ourselves. We adapt PixArt for inpainting using the same procedure employed to transform Stable Diffusion [44] from generation to inpainting. Specifically, we incorporate the GLIDE [33] conditioning mechanism, where the generator operates on 9 latent channels: four channels for the latent being denoised, four channels representing the latent of the masked input image, and the last channel containing a downsampled version of the mask. We train two PixArt models at 1024×1024 and 512×512 for *RegenerateImage* and *RegenerateCrop*, respectively.

We also compare with Stable Diffusion variants of these two approaches for reference: SDXL [39] operates on the entire 1024×1024 image, while SD2-crop [44] operates on a 512×512 crop. It is important to note that these models utilize different architectures and were trained on different datasets, and hence are not directly comparable. We include them in this comparison only as references for state-of-the-art quality.

4.2 Inference time

We illustrate the overall runtime of all methods in Fig. 5. The baselines run is constant time, as they operate on fixed size tensors derived from the fixed input size – full canvas for *RegenerateImage* and a fixed-size crop for *RegenerateCrop*. In contrast, *LazyDiffusion*'s runtime scales with the mask size, because our decoder processes tensors with dimensions proportional to the masked region. This leads to significant speedups for small masks, typical of interactive editing applications. For example, with a mask covering 10% of the image our

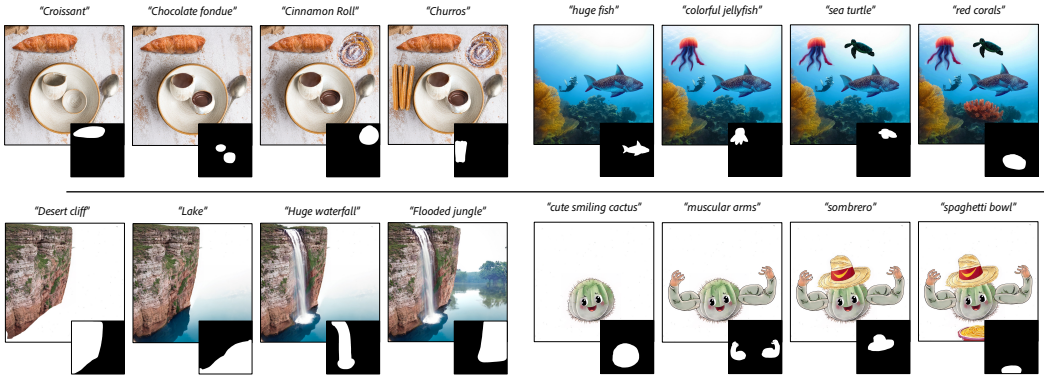

Fig. 6: Progressive image editing (top) and image generation (bottom) using *LazyDiffusion*. Each panel illustrates a generative progression compared to the preceding state of the canvas to its left. *LazyDiffusion* markedly accelerates local image edits (approximately $\times 10$), rendering diffusion models more apt for user-in-the-loop applications.

model achieves a $\times 10$ speedup over *RegenerateImage*. Similarly, *LazyDiffusion* is also faster than *RegenerateCrop* for masks smaller than 25%. At mask ratio 25%, both methods generate the same number of pixels and have comparable running times. For larger masks, *RegenerateCrop* is faster but generates low-resolution crops and naively upsamples to native resolution, reducing sharpness. Additionally, *RegenerateCrop* often fails to produce outputs that are consistent with the region outside the mask, as we discuss below (Sec. 4.4).

While there are additional networks in the pipeline, the diffusion decoder is the only component running multiple times, and thus dominates the runtime. Notably, our context encoder adds a 73ms overhead, which is dwarfed by the cost of the diffusion loop. The latent encoder and decoder take 97ms and 176ms, respectively, and the T5 text encoder 21ms. These are shared by all methods.

Scaling laws. Our method essentially reduces the cost of each denoising iteration at the price of a small overhead for the context encoder, to balance quality with context retention. As a result, our performance gains are most striking for high diffusion step counts (typically correlated with higher image quality), and smaller mask sizes (most frequent in interactive applications). A single evaluation of our decoder takes 374ms to generate *full* image, but only 28ms for 10% masks — a $\times 13.4$ speedup, greater than the encoder's overhead. So, our method remains beneficial for few-step [50], or even one-step models [58]. We expect the performance gains provided by our strategy to be even more striking on costlier applications like high-resolution image editing, or video synthesis [3].

4.3 Progressive generation

Diffusion models are challenging to integrate into interactive pipelines due to their high latency. There exists an abundance of research on broadly accelerating diffusion models [28,50,58], but in the context of this study, we highlight that

individuals often tackle tasks incrementally, executing operations progressively and concentrating on local adjustments one at a time—whether it involves adding or removing objects, refining, or retrying previous attempts. *LazyDiffusion* significantly accelerates such local operations, making it well-suited for interactive pipelines with a user-in-the-loop.

In Fig. 6, we showcase a couple of iterations using *LazyDiffusion* for both image editing and image generation, starting from a blank canvas. Furthermore, we attach a supplemental video that showcases authentic user interactions with both *LazyDiffusion* and our *RegenerateImage* baseline, highlighting the discernible difference in running time between the two.

4.4 Inpainting quality

A distinctive feature of *LazyDiffusion* is its utilization of a compressed global context to aid inpainting. In contrast, *RegenerateImage* utilizes the complete global context, while *RegenerateCrop* relies on the context provided by pixels neighboring the mask. We now compare the results produced by these approaches.

For quantitative evaluation, we report zero-shot FID [20] and CLIPScore [19], which estimate similarity to real images and text-image alignment, respectively. Additionally, we include scores for SDXL [39] and SD2-crop [44]. Despite not being directly comparable, because they use different architectures and training data, they serve as references for state-of-the-art quality. In Table 1, we report mean scores over a random sample of 10,000 images drawn from OpenImages [47]. Notably, text-image alignment (CLIP) remains unaffected by the mechanism to use image context. On the FID metric, *LazyDiffusion* exhibits only a marginal increase compared to *RegenerateImage* (%4) and performs significantly better than *RegenerateCrop* (%26).

Table 1: Quantitative comparison of our method with the three baselines. We report zero-shot FID [20] and CLIPScore [19] on $10k$ images images from OpenImages [47]. Scores of SD2-crop [44] and SDXL [39] are not directly comparable and provided only for reference.

Method	CLIP Score (↑)	FID (↓)
SD2-crop	0.21	6.95
SDXL	0.21	6.88
RegenerateCrop	0.19	9.35
RegenerateImage	0.19	7.38
LazyDiffusion (Ours)	0.19	7.70

We show qualitative comparisons in Fig. 7. Our examination reveals a significant discrepancy in the performance of models regenerating a crop – *RegenerateCrop* and SD2-crop. In many instances, inpainting involves generating an object that is visually independent of other concepts in the image, such as adding a side of fries next to a burger. Here, models operating on a tight crop can produce reasonable-looking objects and seamlessly blend them with the surrounding

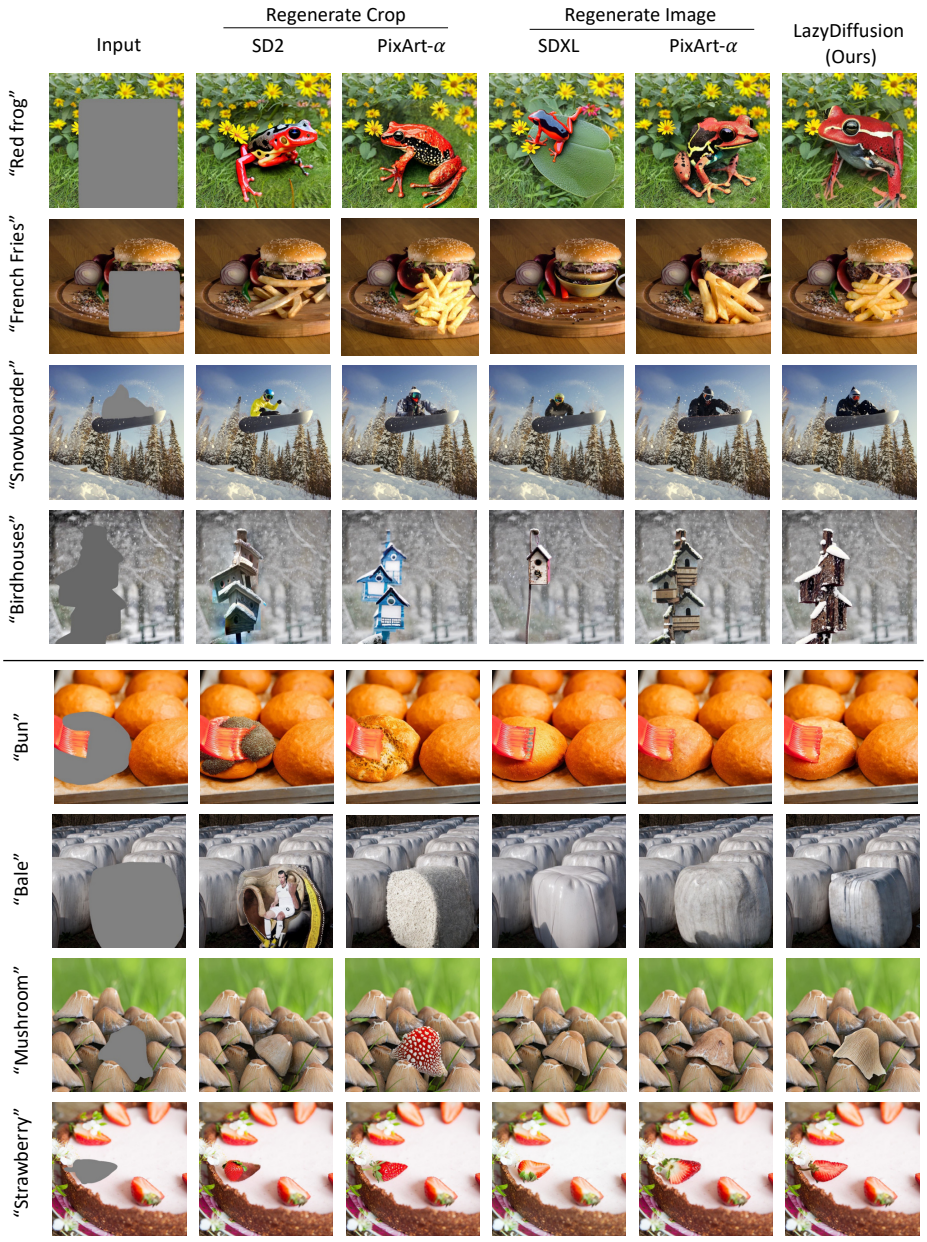

Fig. 7: Comparing Inpainting Results: (Top) Inpainting most objects requires relatively little semantic context. In such cases, all methods produce reasonably good results, even those processing only a tight crop. (Bottom) However, when inpainting an object closely related to others, such as one bun out of many, the inpainting model requires robust semantic understanding. Methods processing only a crop produce objects that may seem reasonable in isolation, but do not fit well within the greater context of the image. In contrast, *LazyDiffusion* adeptly leverages the compressed image context to generate high-fidelity results, comparable in quality to models regenerating the entire image and running up to ten times slower.

pixels available in the crop (Fig. 7 (Top)). However, in numerous scenarios, the goal is to add an object that is strongly related to the existing context, such as adding another bun to a tray of buns. Models operating solely on a crop lack knowledge of the global image and consequently produce objects that may seem reasonable in isolation but do not fit well within the greater image context (Fig. 7 (Bottom)). In contrast, SDXL and *RegenerateImage* utilize direct and full access to all image pixels to consistently yield highly realistic results, where the generated region fits well with the existing content. Notably, we find that *LazyDiffusion* behaves similarly and produces comparable results even in these challenging edge cases. This suggests that the compressed image context is highly expressive and encodes meaningful semantic information.

User study. We measure the models' capability to produce highly-contextual inpainting through a user study. For this, we curate a specialized test set comprising scenarios that necessitate a high level of semantic image context for effective inpainting. Specifically, we select images featuring several closely related objects, such as a set of uniform buns on a tray. Subsequently, we evaluate all models based on their ability to regenerate one of these objects when masked. In this scenario, the models must rely on visible pixels to produce a high-fidelity result. Users are presented with the masked input image, a text prompt, and two results — ours and a baseline. They are then asked to "select the option in which the inpainted image, as a whole, looks best". We collect a total of 1778 responses from 48 unique users and find that our method is strongly preferred over methods operating solely on a crop and competitive with those regenerating the entire image. Specifically, *LazyDiffusion* is preferred over *RegenerateCrop* in 81% of cases, over SD2-crop in 82.5% of cases, over *RegenerateImage* in 46.1% of cases, and over SDXL in 48.5% of cases. These results indicate that the compressed encoder context retains the core semantic information required even for challenging use cases. In short, our model demonstrates competitive quality to our conceptual upper-bound *RegenerateImage*, but runs up to ten times faster.

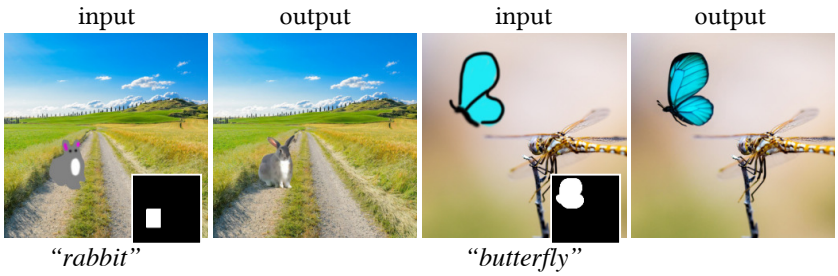

Fig. 8: Our model readily supports additional forms of local conditioning. For example, similar to SDEdit [31], a user can draw a simplistic colored sketch, providing the model shape and color information.

4.5 Sketch-guided inpainting

So far, our emphasis has been on generation guided solely by the mask and a text prompt. However, in principle, our method is applicable to any localized generation task and can accommodate other forms of conditioning, such as sketches and edge maps. In Fig. 8, we briefly showcase this versatility by guiding the generation with a coarse color sketch provided by the user. Following the SDEdit [31] approach, we initiate the generation process from the partially noised input image instead of Gaussian noise.

5 Conclusions, limitations and future work

We introduced a novel transformer-based encoder-decoder architecture for interactive image generation and editing using a diffusion model. Our approach reduces the diffusion runtime by only generating the patches corresponding to the small region to synthesize, rather than the entire image. This is achieved through a global context encoder that summarizes the entire image once, outside the diffusion sampling loop, ensuring globally-consistent outputs.

Our method maintains the generation quality of state-of-the-art models, and reduces runtime costs proportionally to the size of the region to generate. This reduction in latency, particularly for small masks, transforms image generation into an interactive process by spreading the generation cost across multiple user interactions.

Our architecture does have some weaknesses. Despite operating outside the diffusion loop, the context encoder processes the entire image, posing a potential bottleneck for very high-resolution images due to its quadratic scaling in input size. Addressing this limitation could enhance the scalability and applicability of our approach to larger and more intricate visual content. We observed that occasionally, generated results have a subtle color shift compared to the visible image regions, leading to visible patch boundaries. While the Poisson blending post-processing methods discussed in Section 3.3 effectively mitigates these issues, future research is needed to identify a more principled and systematic solution.

Societal impact. Generative models, including our work, can be used to produce misleading content that causes societal harm. Nevertheless, our work does not introduce unique concerns beyond this. We hope that increased public awareness and the development of more automatic tools for detecting generated media will mitigate these risks.

Acknowledgement. We are grateful to Minguk Kang, Tianwei Yin and Wei-An Lin for technical suggestions, to Rotem Shalev-Arkushin for proofreading our draft and offering feedback, and to Yogev Nitzan for his help running the user study. This work was done while Yotam Nitzan was an intern at Adobe.

References

1. Avrahami, O., Lischinski, D., Fried, O.: Blended diffusion for text-driven editing of natural images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18208–18218 (2022) 2, 3, 5
2. Betker, J., Goh, G., Jing, L., Brooks, T., Wang, J., Li, L., Ouyang, L., Zhuang, J., Lee, J., Guo, Y., et al.: Improving image generation with better captions. Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf **2**, 3 (2023) 3
3. Brooks, T., Peebles, B., Holmes, C., DePue, W., Guo, Y., Jing, L., Schnurr, D., Taylor, J., Luhman, T., Luhman, E., Ng, C., Wang, R., Ramesh, A.: Video generation models as world simulators (2024), https://openai.com/research/video-generation-models-as-world-simulators 4, 10
4. Cao, M., Wang, X., Qi, Z., Shan, Y., Qie, X., Zheng, Y.: Masactrl: Tuning-free mutual self-attention control for consistent image synthesis and editing. arXiv preprint arXiv:2304.08465 (2023) 5
5. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 9650–9660 (2021) 7
6. Chang, H., Zhang, H., Jiang, L., Liu, C., Freeman, W.T.: Maskgit: Masked generative image transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11315–11325 (2022) 4
7. 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) 4, 6, 7, 8
8. Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Luan, D., Sutskever, I.: Generative pretraining from pixels. In: International conference on machine learning. pp. 1691–1703. PMLR (2020) 4
9. Couairon, G., Verbeek, J., Schwenk, H., Cord, M.: Diffedit: Diffusion-based semantic image editing with mask guidance. arXiv preprint arXiv:2210.11427 (2022) 5
10. 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) 8
11. Dhariwal, P., Nichol, A.: Diffusion models beat gans on image synthesis. Advances in neural information processing systems **34**, 8780–8794 (2021) 3
12. Ding, M., Zheng, W., Hong, W., Tang, J.: Cogview2: Faster and better text-to-image generation via hierarchical transformers. Advances in Neural Information Processing Systems **35**, 16890–16902 (2022) 4
13. 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) 2, 6
14. Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., Podell, D., Dockhorn, T., English, Z., Lacey, K., Goodwin, A., Marek, Y., Rombach, R.: Scaling rectified flow transformers for high-resolution image synthesis (2024) 4
15. Esser, P., Rombach, R., Ommer, B.: Taming transformers for high-resolution image synthesis. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12873–12883 (2021) 4
16. Gao, S., Zhou, P., Cheng, M.M., Yan, S.: Masked diffusion transformer is a strong image synthesizer. arXiv preprint arXiv:2303.14389 (2023) 4

17. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16000–16009 (2022) 4
18. Hertz, A., Mokady, R., Tenenbaum, J., Aberman, K., Pritch, Y., Cohen-Or, D.: Prompt-to-prompt image editing with cross attention control. arXiv preprint arXiv:2208.01626 (2022) 5
19. 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) 11
20. 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) 11
21. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in neural information processing systems **33**, 6840–6851 (2020) 3
22. 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) 3
23. Li, J., Li, D., Savarese, S., Hoi, S.: Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. arXiv preprint arXiv:2301.12597 (2023) 8
24. Li, M., Lin, J., Meng, C., Ermon, S., Han, S., Zhu, J.Y.: Efficient spatially sparse inference for conditional gans and diffusion models. Advances in neural information processing systems **35**, 28858–28873 (2022) 4
25. Liu, W., Cun, X., Pun, C.M., Xia, M., Zhang, Y., Wang, J.: Coordfill: Efficient high-resolution image inpainting via parameterized coordinate querying. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 1746–1754 (2023) 4
26. Liu, X., Zhang, X., Ma, J., Peng, J., Liu, Q.: Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. arXiv preprint arXiv:2309.06380 (2023) 3
27. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017) 8
28. Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., Zhu, J.: Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. Advances in Neural Information Processing Systems **35**, 5775–5787 (2022) 3, 10
29. Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., Zhu, J.: Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. arXiv preprint arXiv:2211.01095 (2022) 3
30. Luo, S., Tan, Y., Huang, L., Li, J., Zhao, H.: Latent consistency models: Synthesizing high-resolution images with few-step inference. arXiv preprint arXiv:2310.04378 (2023) 3
31. Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J.Y., Ermon, S.: Sdedit: Guided image synthesis and editing with stochastic differential equations. arXiv preprint arXiv:2108.01073 (2021) 3, 13, 14
32. Nguyen, T.H., Tran, A.: Swiftbrush: One-step text-to-image diffusion model with variational score distillation. arXiv preprint arXiv:2312.05239 (2023) 3
33. 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, 5, 9
34. Nichol, A.Q., Dhariwal, P.: Improved denoising diffusion probabilistic models. In: International Conference on Machine Learning. pp. 8162–8171. PMLR (2021) 8

35. Patashnik, O., Garibi, D., Azuri, I., Averbuch-Elor, H., Cohen-Or, D.: Localizing object-level shape variations with text-to-image diffusion models. arXiv preprint arXiv:2303.11306 (2023) 5
36. Peebles, W., Xie, S.: Scalable diffusion models with transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4195–4205 (2023) 2, 4, 6, 7
37. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. In: ACM SIGGRAPH 2003 Papers, pp. 313–318 (2003) 8
38. von Platen, P., Patil, S., Lozhkov, A., Cuenca, P., Lambert, N., Rasul, K., Davaadorj, M., Wolf, T.: Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers (2022) 2, 3, 8
39. 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) 3, 8, 9, 11
40. 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) 8
41. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research **21**(1), 5485–5551 (2020) 7
42. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with clip latents. arXiv preprint arXiv:2204.06125 **1**(2), 3 (2022) 3
43. Razavi, A., Van den Oord, A., Vinyals, O.: Generating diverse high-fidelity images with vq-vae-2. Advances in neural information processing systems **32** (2019) 4
44. 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) 2, 3, 4, 5, 6, 8, 9, 11
45. Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E.L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al.: Photorealistic text-to-image diffusion models with deep language understanding. Advances in Neural Information Processing Systems **35**, 36479–36494 (2022) 3
46. Salimans, T., Ho, J.: Progressive distillation for fast sampling of diffusion models. arXiv preprint arXiv:2202.00512 (2022) 3
47. Schuhmann, C., Vencu, R., Beaumont, R., Kaczmarczyk, R., Mullis, C., Katta, A., Coombes, T., Jitsev, J., Komatsuzaki, A.: Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. arXiv preprint arXiv:2111.02114 (2021) 11
48. Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics. In: International conference on machine learning. pp. 2256–2265. PMLR (2015) 3
49. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. arXiv preprint arXiv:2010.02502 (2020) 3
50. Song, Y., Dhariwal, P., Chen, M., Sutskever, I.: Consistency models (2023) 3, 10
51. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. arXiv preprint arXiv:2011.13456 (2020) 3
52. Tevet, G., Raab, S., Gordon, B., Shafir, Y., Cohen-Or, D., Bermano, A.H.: Human motion diffusion model. arXiv preprint arXiv:2209.14916 (2022) 4
53. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017) 2, 6

54. 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) 2, 5, 6, 8
55. stable-diffusion webui: stable-diffusion-webui. https://github.com/AUTOMATIC1111/stable-diffusion-webui (2024), accessed: Jan 2024 2, 3, 8
56. Wei, C., Mangalam, K., Huang, P.Y., Li, Y., Fan, H., Xu, H., Wang, H., Xie, C., Yuille, A., Feichtenhofer, C.: Diffusion models as masked autoencoders. arXiv preprint arXiv:2304.03283 (2023) 4
57. 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) 5, 8
58. Yin, T., Gharbi, M., Zhang, R., Shechtman, E., Durand, F., Freeman, W.T., Park, T.: One-step diffusion with distribution matching distillation. arXiv preprint arXiv:2311.18828 (2023) 3, 10
59. 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) 2
60. Zhao, S., Cui, J., Sheng, Y., Dong, Y., Liang, X., Chang, E.I., Xu, Y.: Large scale image completion via co-modulated generative adversarial networks. In: International Conference on Learning Representations (ICLR) (2021) 2
61. Zheng, H., Nie, W., Vahdat, A., Anandkumar, A.: Fast training of diffusion models with masked transformers. arXiv preprint arXiv:2306.09305 (2023) 4
62. 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) 8