SinDDM: A Single Image Denoising Diffusion Model Supplementary Material

Vladimir Kulikov [1] **Shahar Yadin** [*1] **Matan Kleiner** [*1] **Tomer Michaeli** [1]

Contents

A. Additional Examples

We provide additional examples for all the applications described in the main text. Figure S1 presents additional examples for unconditional sampling. In Figs. S2 and S3 we provide additional examples for image generation with text-guided contents, with and without ROI guidance, respectively. Figures S4, S5, S6, S7 and S8 provide additional examples of image generation with text-guided style, and in Figure S9 we provide examples for text-guided style transfer. Finally, Figure S10 illustrates generation guided by image contents within ROIs.

Figure S1. **Unconditional image generation.**

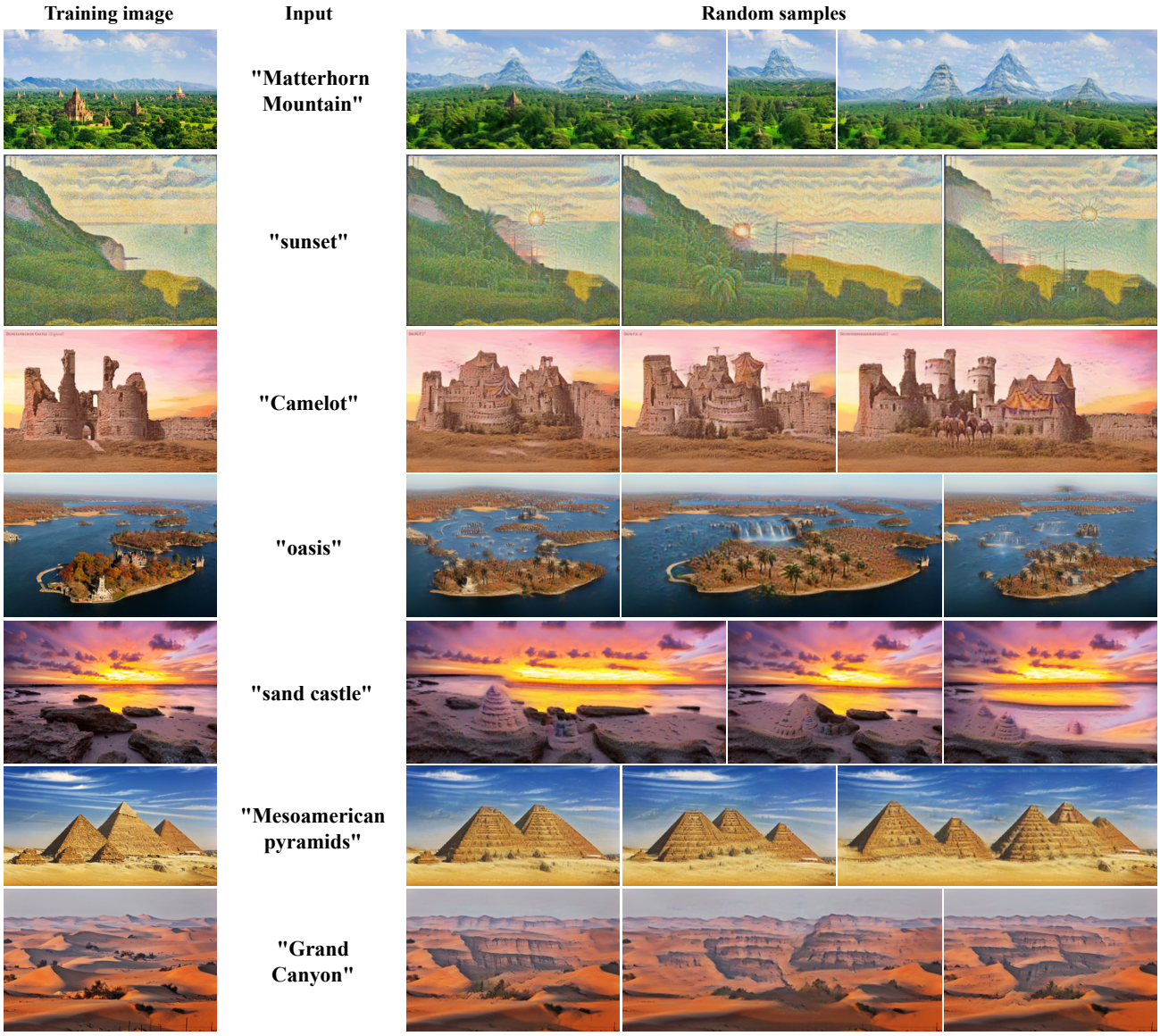

Figure S2. **Text-guided content generation.**

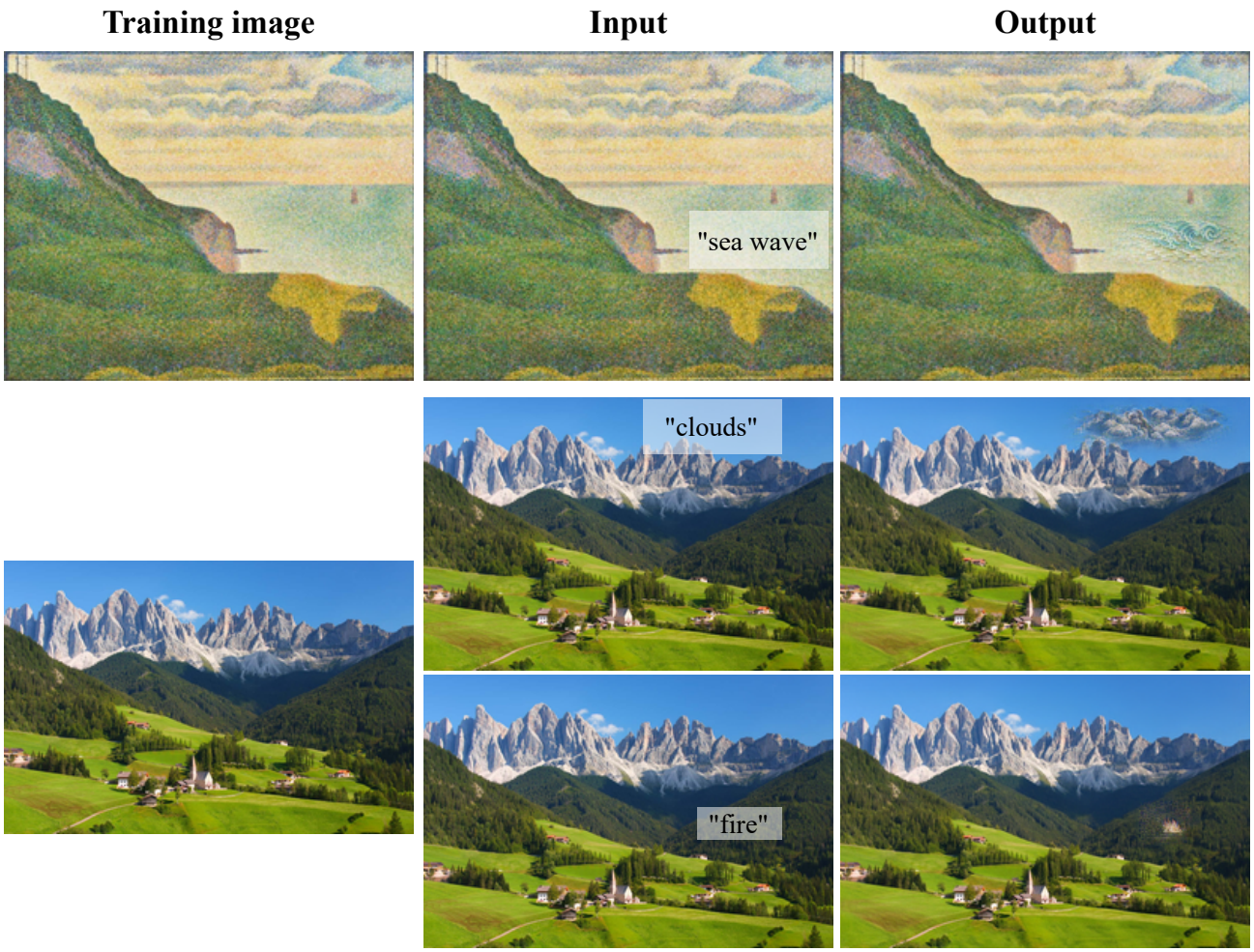

Figure S3. **Text-guided content generation within a ROI.**

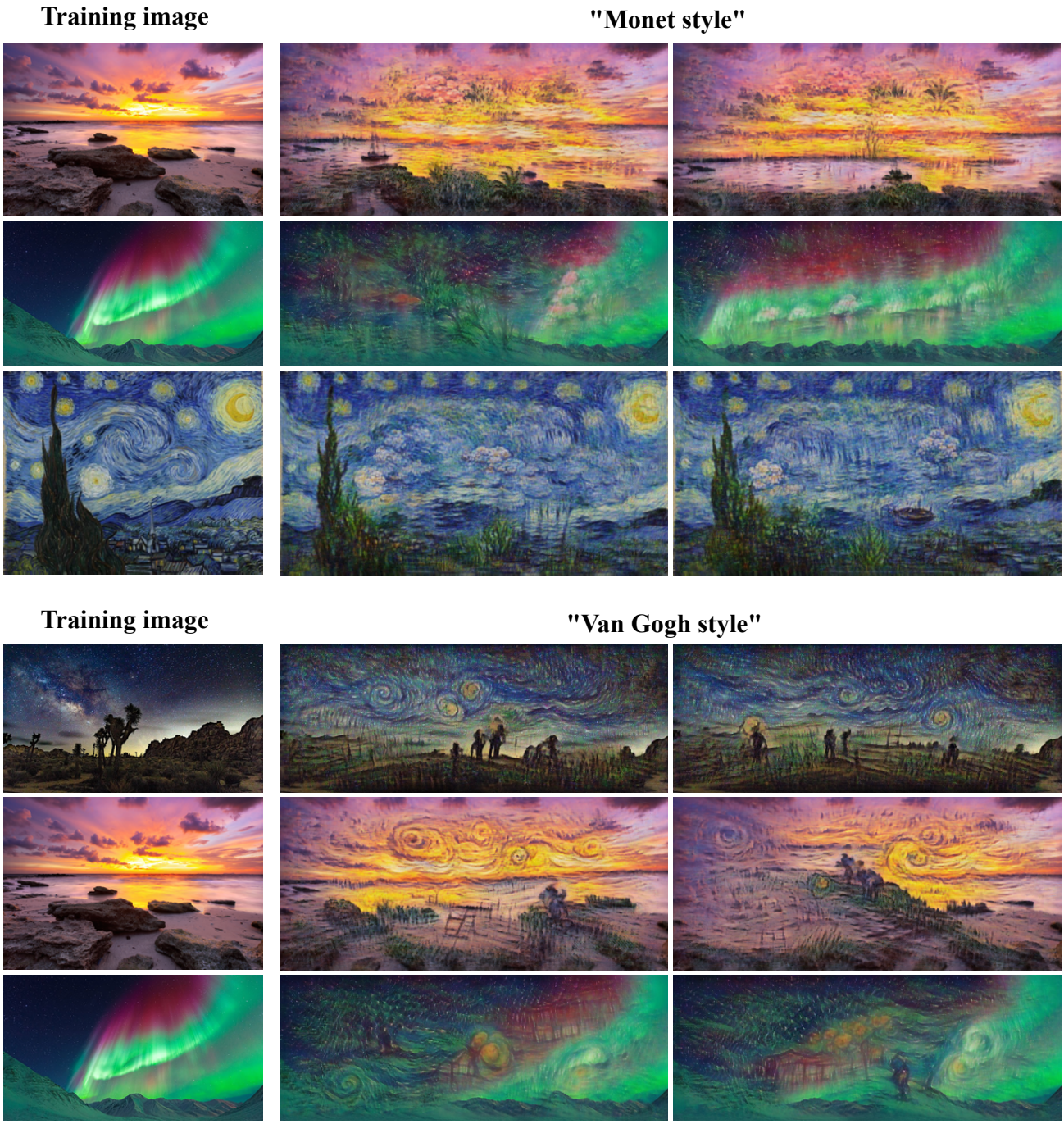

Figure S4. **Sampling with text-guided style at arbitrary aspect ratios.**

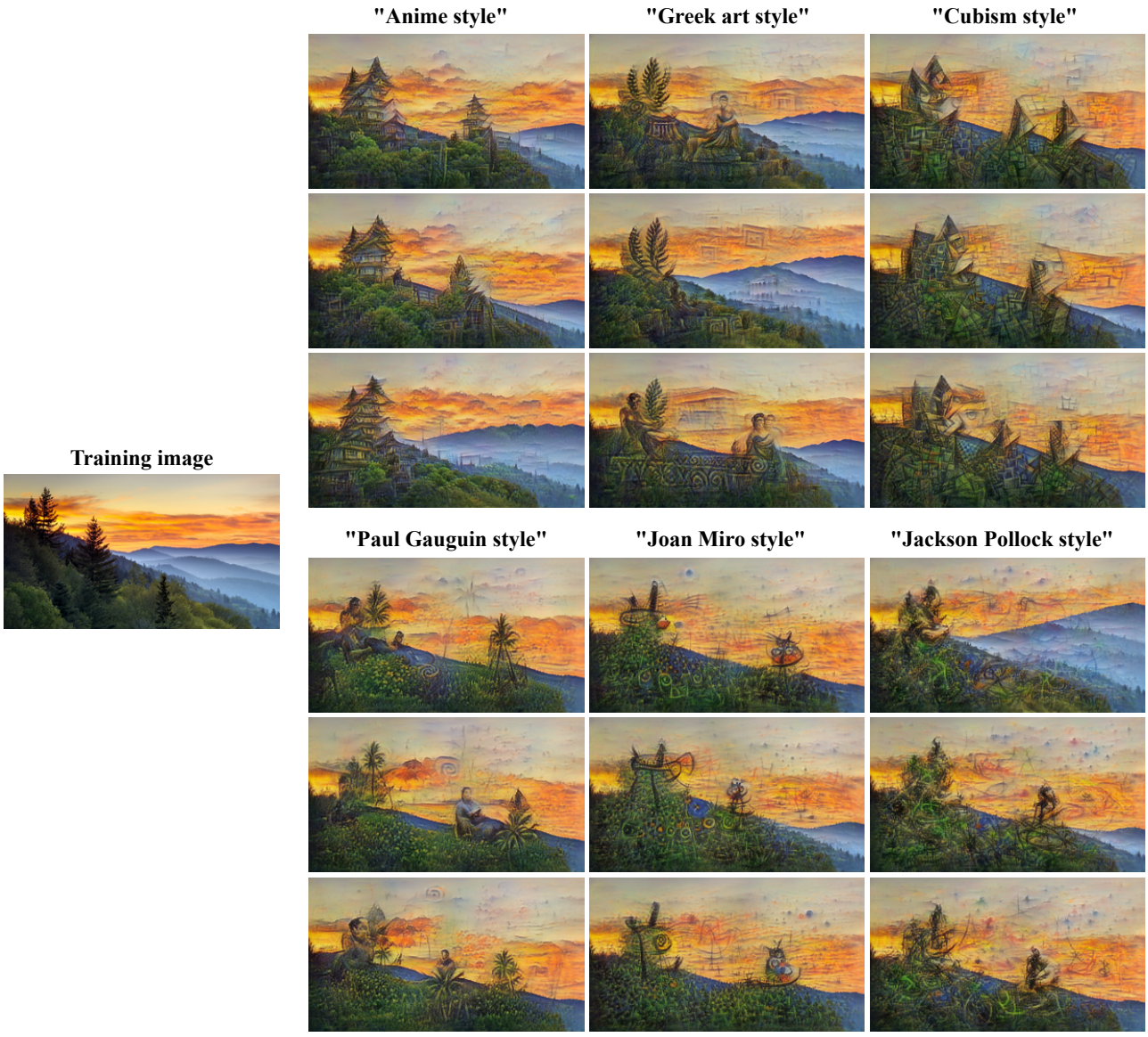

Figure S5. **Sampling with text guided style.**

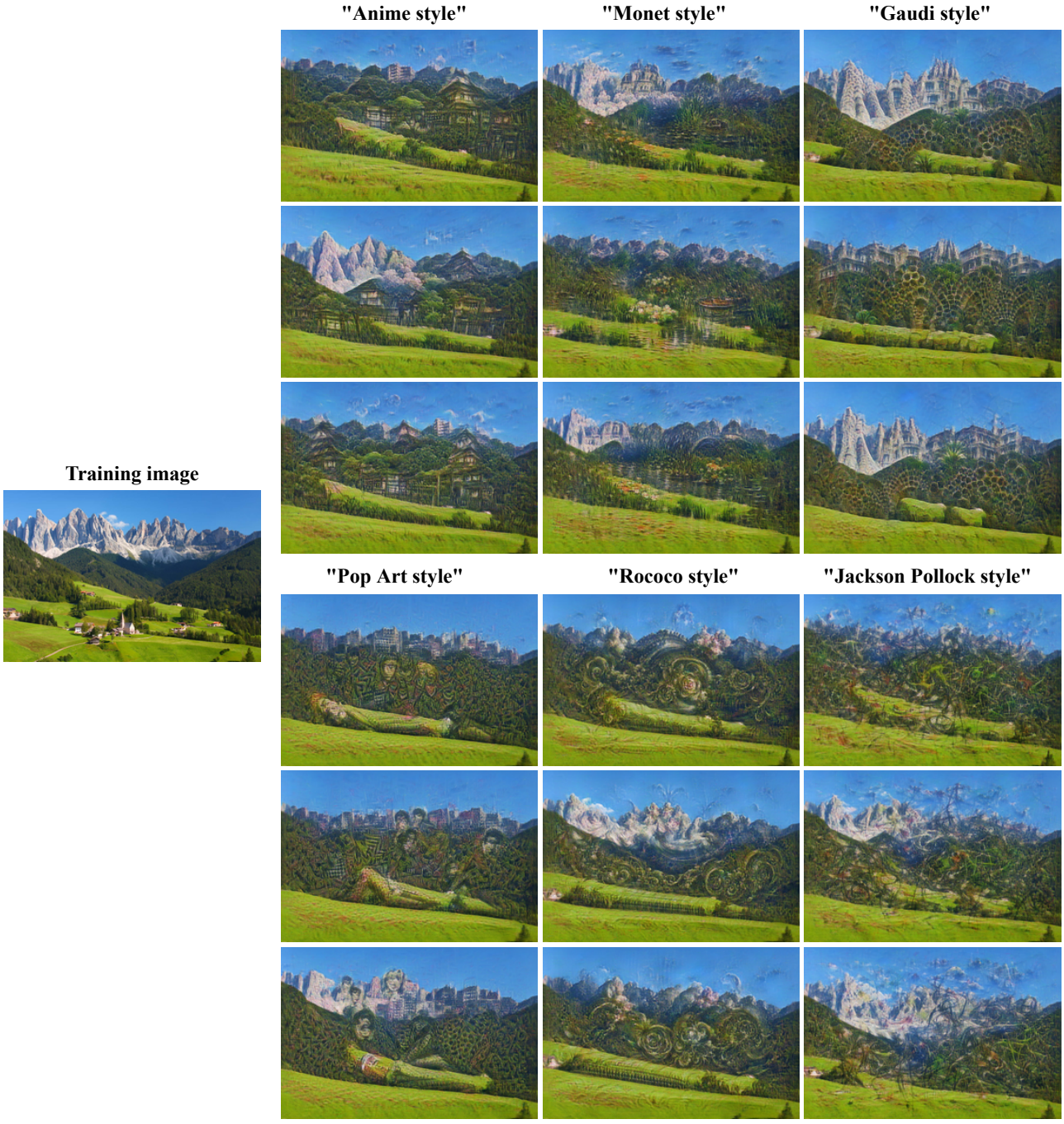

Figure S6. **Sampling with text guided style.**

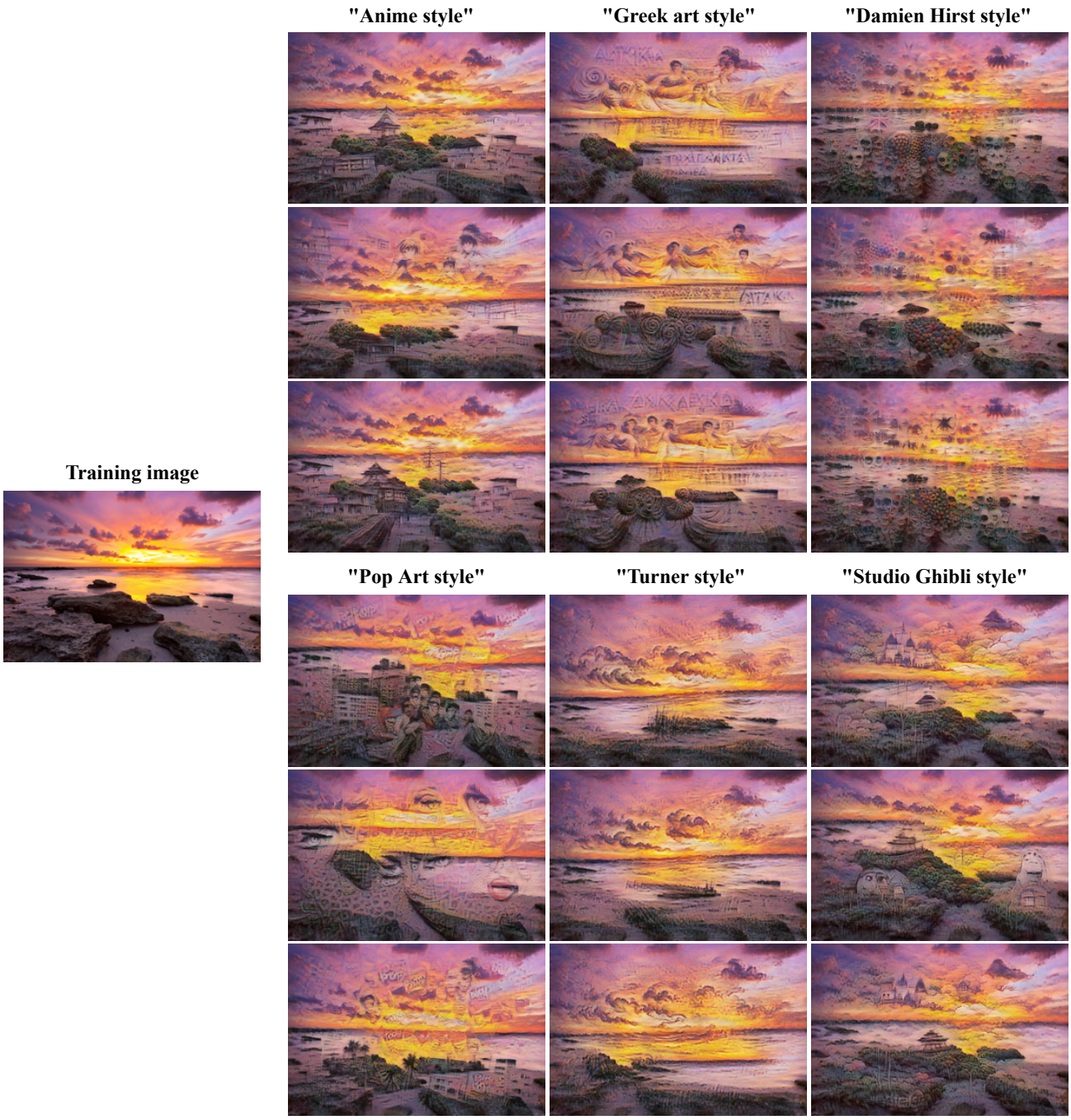

Figure S7. **Sampling with text guided style.**

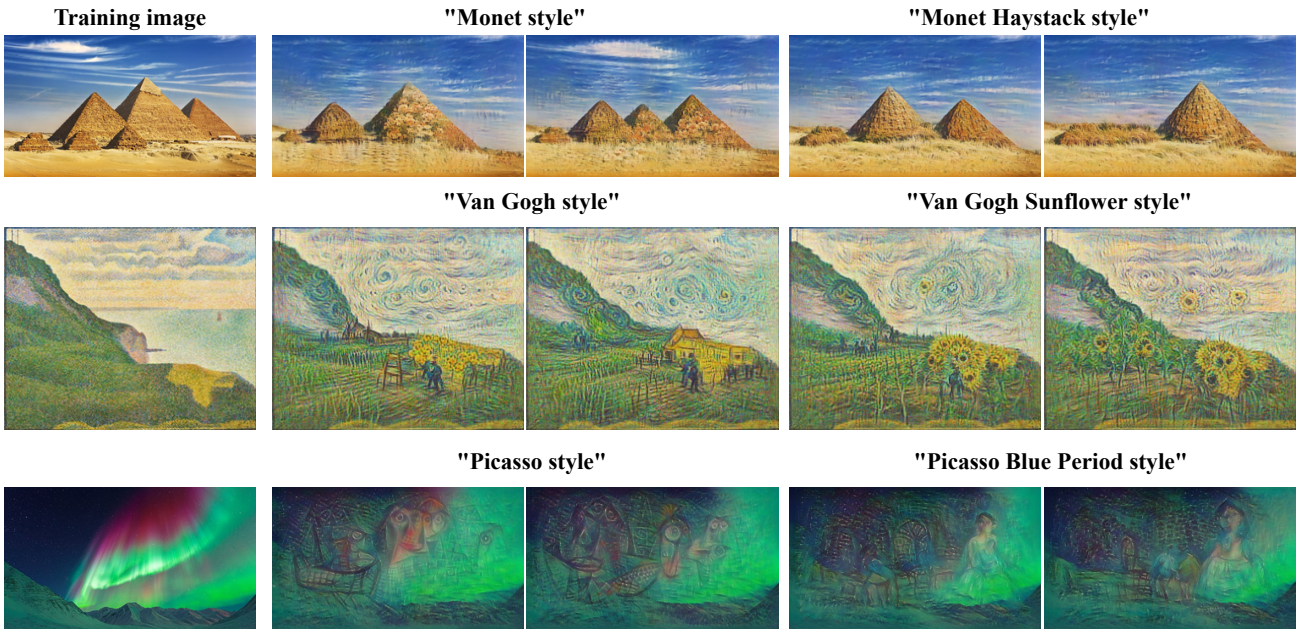

Figure S8. **Sampling with text guided style using a focused text prompt.** Oftentimes, when specifying a general style, CLIP guides the generation towards the most famous attributes associated with that style. For example, “Monet style” usually results in adding flowers. To obtain more focused results, it is possible to add the name of a painting or a distinct period associated with the desired artist. The resulting effect is exemplified on the right.

Figure S9. **Text-guided style transfer.** Rather than controlling the style of the random samples generated by our model, we can also use SinDDM to modify the style of the training image itself. We achieve this by injecting the training image directly to the finest scale so that the modifications imposed by our denoiser and by the CLIP guidance only affect the fine textures. This leads to a style-transfer effect, but where the style is dictated by a text prompt rather than by an example style image.

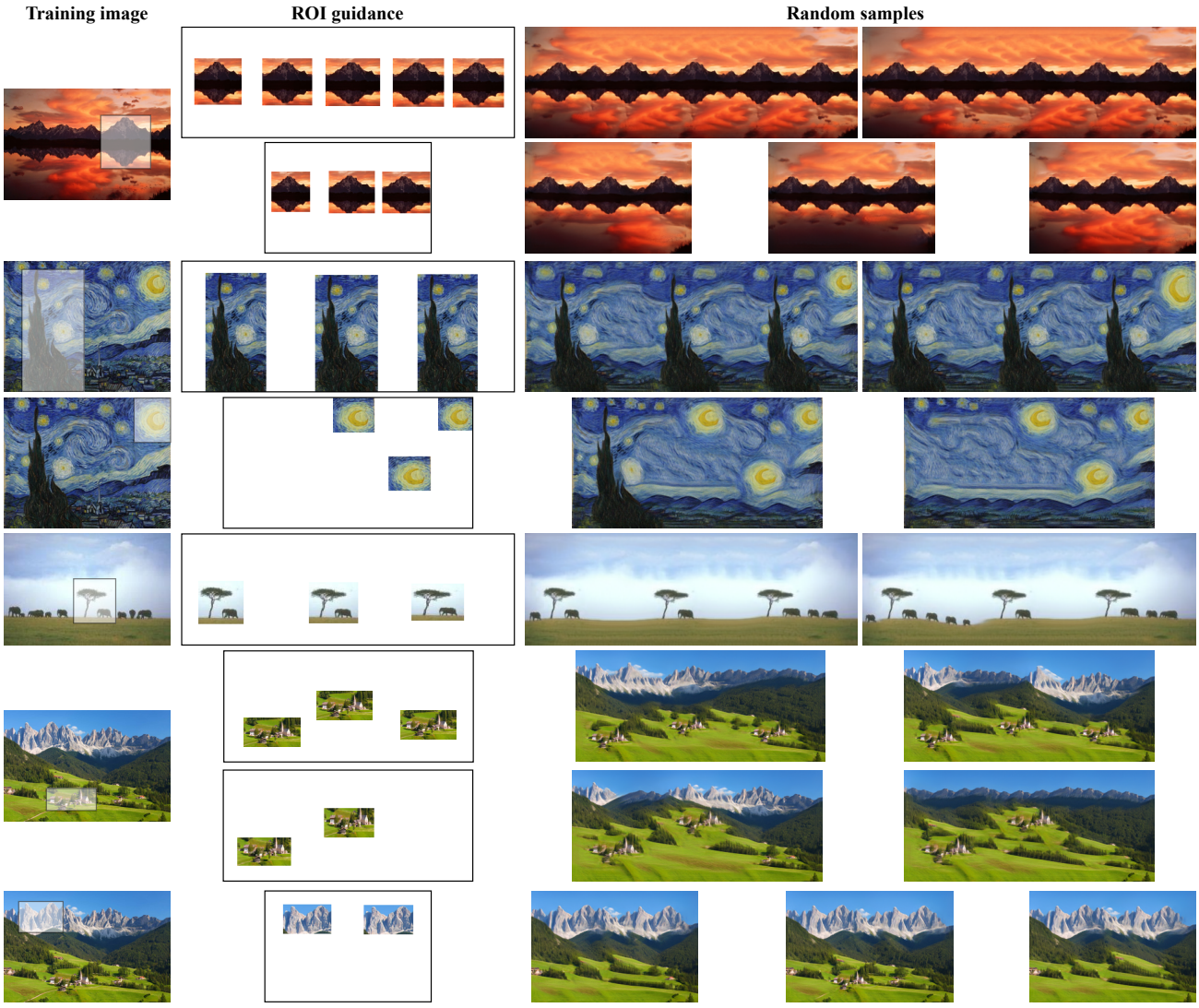

Figure S10. **Generation guided by image contents within ROIs.**

B. SinDDM Architecture

Similarly to externally-trained diffusion models, SinDDM receives a noisy image and a timestep t as inputs, and it predicts the noise. However, unlike external models, in our setting the noisy image is also a bit blurry, and our model also accepts the scale s as input. This is illustrated in Figure S11(a). Since we train on a single image, we must take measures to avoid memorization of the image. We do so by limiting the receptive field of the model. Specifically, we use a fully convolutional pipeline for the image input, which consists of 4 SinDDM conv blocks (Figure S11(c)). Each of these blocks has a receptive field of 9×9, yielding a total receptive field of 35×35. The timestep t and scale s first pass through an embedding block (Figure S11(b)), in which they go through Sinusoidal Positional Embedding (SPE) and then concatenated and passed through two fully-connected layers with GeLU activations to yield a time-scale embedding vector ts (Figure S11(b)). SinDDM's conv block's data input pipeline consists of 2D convolutions with a residual connection and GeLU activations, and the input time-scale embedding vector is passed through a GeLU activation and a fullly-connected layer. The time-scale embedding vector is passed to each SinDDM conv block as depicted in Figure S11(a).

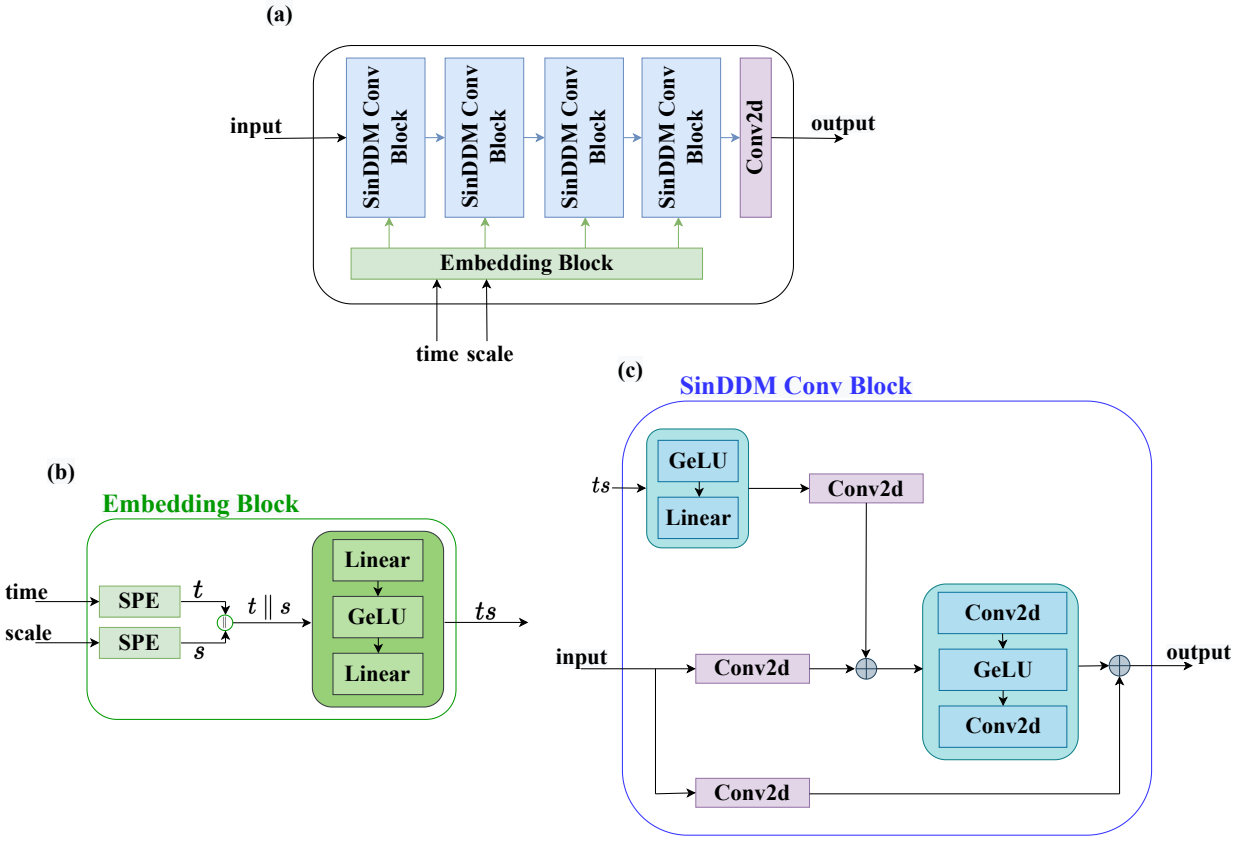

Figure S11. **SinDDM architecture.** Our model's architecture (a) comprises an embedding block (b) that gets as inputs the timestep t and scale s, and a fully convolutional image pipeline (c) that is conditioned on the embedding vector of t and s.

C. Training Details

We train our model using the Adam optimizer with its default Torch parameters. Similarly to the original DDPM training scheme, we apply an Exponential Moving Average (EMA) on the model's weights with a decay factor of 0.995. When using an A6000 GPU, we train our model for 120×10^3 steps with an initial learning rate of 0.001, which is reduced by half on $[20, 40, 70, 80, 90, 110] \times 10^3$ steps. We set the batch size to be 32. For a 200×250 image, training takes around 7 hours.

We also experimented with training on an RTX2080 GPU, which is slower and has less memory. In this case, we take the batch size to be 16, multiply the number of steps by 4, and modify the scheduler steps accordingly. In this setting, training on a 200×250 image takes around 20 hours.

D. Derivation of the Forward and Reverse Diffusion Processes

In the sampling process, at each scale except for the coarsest ($s = 0$), the reverse diffusion begins with a noisy version of the upsampled image generated at the previous scale. Since the upsampling introduces some blur, this implies that our model's task is to perform both deblurring and denoising. To account for this, our training algorithm for $s > 0$ is not identical to the standard one. Specifically, in our case we construct the image x_t at train time as a linear combination of not only the clean training image and Gaussian noise (as in DDPM (Ho et al., 2020)), but also of an upsampled version of the training image from the previous scale, which is a bit blurry.

Concretely, at train time we define

$$x_t^{s,\text{mix}} = \gamma_t^s \tilde{x}^s + (1 - \gamma_t^s) x^s, \tag{1}$$

where $\gamma_t^s \in [0, 1]$ is a non-increasing monotonic function of t (see below). Thus, $x_t^{s,\text{mix}}$ is a linear combination of x^s and its blurry version $\tilde{x}^s$. We then construct x_t^s as

$$x_t^s = \sqrt{\bar{\alpha}_t} x_t^{s,\text{mix}} + \sqrt{1 - \bar{\alpha}_t}\epsilon, \tag{2}$$

where $\epsilon \sim \mathcal{N}(0, I)$ and $\bar{\alpha}_t$ follows a cosine schedule as in Improved DDPM (Nichol & Dhariwal, 2021). Therefore, conditioned on x^s and $\tilde{x}^s$, we have that $x_t^s \sim \mathcal{N}(\sqrt{\bar{\alpha}_t} x_t^{s,\text{mix}}, \sqrt{1 - \bar{\alpha}_t} I)$. Now, for each timestep t, we train our model to predict the noise from x_t^s, as in DDPM (Ho et al., 2020). Substituting our noise estimate for ϵ in (2), we can isolate $x_t^{s,\text{mix}}$. This is line 6 of the sampling algorithm in the main text. And given $x_t^{s,\text{mix}}$, we can isolate x^s from (1). This leads to the estimate $\hat{x}_0^s$ appearing in line 7 of the sampling algorithm.

D.1. The Maximal Noise Level in Each Scale

We train each scale for $T = 100$ timesteps. However, when sampling from the model, we start the generation in every scale $s > 0$ from

$$T[s] = \min \left\{ t \;\; : \;\; \frac{\sqrt{1 - \bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}} > \|x^s - \tilde{x}^s\|_2 \right\}. \tag{3}$$

This ensures that the effective noise-to-signal ratio that is due to the noise, is roughly the same as the L^2 norm between the original image in scale s and its blurry version (the upsampled image from scale $s - 1$). In other words, this guarantees that we add just enough noise to generate the missing details in that scale, but not too much to mask the details already generated in the previous scale. It should be noted that with this strategy, we train on timesteps that we do not eventually use in sampling (those with $t > T[s]$). However, we found this to lead to better results than training only on the timesteps used in sampling.

D.2. The γ_t^s Schedule

For $s = 0$, we set $\gamma_t^s = 0$ for all t since we perform only denoising and not deblurring. For every $s > 0$ and $t \leq T[s]$ we choose γ_t^s such that the noise level is similar to the blur level. Thus, in the reverse diffusion, each step performs denoising just as it does deblurring. We do this by choosing γ_t^s such that

$$\left\| x^s - x_t^{s,\text{mix}} \right\|_2 = \frac{\sqrt{1 - \bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}}. \tag{4}$$

This equation possesses a closed form solution (obtained by substituting (1) in (4)), given by

$$\gamma_t^s = \frac{1}{\|x^s - \tilde{x}^s\|_2} \frac{\sqrt{1 - \bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}}. \tag{5}$$

Recall that during training we also work with $t > T[s]$, in which case the equation would give $\gamma_t^s > 1$. Therefore, at train time we use $\gamma_t^s = 1$ for $t > T[s]$. During sampling, we clamp γ_t^s to 0.55 for all timesteps and scales.

To illustrate the necessity of adding blur during training we trained models using the aforementioned γ_t^s schedule with $\gamma_t^s = 0, \forall s, t$, meaning without blur. Removing the blurring from the training results in smoother samples with lack of fine details. A qualitative comparison between models trained with and without blur is shown in Figure 5.

D.3. The Sampling Process

As shown in the DDIM formulation (Song et al., 2021), in a regular diffusion process, the distribution $q(x_{t-1}|x_t, x_0)$ can be chosen as

$$q_\sigma(x_{t-1}|x_t, x_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\cdot\frac{x_t-\sqrt{\bar{\alpha}_t}x_0}{\sqrt{1-\bar{\alpha}_t}}, \sigma_t^2\mathbf{I}\right), \tag{6}$$

where σ_t is a hyperparemter. In our case, this translates to

$$q_\sigma(x_{t-1}^s|x_t^s, x_0^s) = \mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}}x_{t-1}^{s,\text{mix}} + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\cdot\frac{x_t^s-\sqrt{\bar{\alpha}_t}x_t^{s,\text{mix}}}{\sqrt{1-\bar{\alpha}_t}}, \sigma_t^2\mathbf{I}\right). \tag{7}$$

We found empirically that the best results are achieved with $\sigma_t^s = \sqrt{\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}}\cdot\sqrt{1-\alpha_t}$ for $s=0$ and $\sigma_t^s = 0$ for $s>0$. As described in (Song et al., 2021), the first choice corresponds to the DDPM sampling process (Ho et al., 2020), while the second choice corresponds to a deterministic process.

E. Text Guided Generation

E.1. Algorithm

As described in the main text, text guidance is achieved by adding CLIP's gradients to $\hat{x}_0^s$ in each step, using an adaptive step size. Specifically, we aim at updating $\hat{x}_0^s$ as

$$\hat{x}_0^s \leftarrow \eta\,\delta\,m^s\odot\nabla_{\hat{x}_0^s}\mathcal{L}_{\text{CLIP}} + (1-m^s)\odot\hat{x}_0^s, \tag{8}$$

where $\delta = \|\hat{x}_0^s\odot m\|/\|\nabla_{\hat{x}_0^s}\mathcal{L}_{\text{CLIP}}\odot m\|$ and $\eta\in[0,1]$ is a *strength* parameter that controls the intensity of the CLIP guidance. However, due to the strong prior of our denoiser (which is overfit to the statistics of the training image) such update steps tend to be ineffective. This is because each denoiser step undoes the preceding CLIP step. To overcome this effect, we use a momentum over $\hat{x}_0^s$. Specifically, our update rule for $\hat{x}_0^s$ is

$$\hat{x}_0^s \leftarrow \eta\,\delta\,m^s\odot\nabla_{\hat{x}_0^s}\mathcal{L}_{\text{CLIP}} + (1-m^s)\odot\left(\lambda\hat{x}_0^s + (1-\lambda)\hat{x}_{0,\text{prev}}^s\right), \tag{9}$$

where $\hat{x}_{0,\text{prev}}^s$ is $\hat{x}_0^s$ from the previous timestep and $\lambda\in[0,1]$ is a momentum parameter which we set to 0.05 in all our experiments. Additionally, in contrast to regular sampling, here we use $\gamma_t^s = 0$ for all t and s and $\sigma_t = \sqrt{\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}}\cdot\sqrt{1-\alpha_t}$ for every s. In other words, we use the DDPM sampling algorithm (Ho et al., 2020) for all scales, which only denoises the image and does not explicitly attempt to remove blur.

E.2. Data Augmentation for Text Guided Generation

Following (Bar-Tal et al., 2022), we augment our images and text, feeding all augmented inputs into CLIP's image encoder and text encoder. For regular text-guidance, we use the image and text augmentations from (Bar-Tal et al., 2022). For text-guidance within a ROI, we feed only the ROI (and its augmentations) into CLIP's image encoder. This requires upsampling the ROI to the size of 224×224, which typically results in a very blurry image. Therefore, in this case, we additionally augment the given text prompt using the following text templates:

- "photo of {}."
- "low quality photo of {}."
- "low resolution photo of {}."
- "low-res photo of {}."
- "blurry photo of {}."
- "pixelated photo of {}."
- "a photo of {}."

- "the photo of {}."
- "image of {}."
- "an image of {}."
- "low quality image of {}."
- "a low quality image of {}."
- "low resolution image of {}."
- "a low resolution image of {}."
- "low-res image of {}."
- "a low-res image of {}."
- "blurry image of {}."
- "a blurry image of {}."
- "pixelated image of {}."
- "a pixelated image of {}."
- "the {}."
- "a {}."
- "{}."
- "{}"
- "{}!"
- "{}..."

E.3. Effect of Initial Scale in Generation with Text Guided Contents

In text-guided image generation, we can choose the scale from which to begin the guidance. All the results in the main text are with CLIP guidance from the second-coarsest scale, $s = 1$. The effect of the initial scale is illustrated in Figure S12. When using CLIP guidance from $s = 0$, we take the first $T/2$ steps to be purely generative, without any guidance. This allows the model to create diverse global structures before incorporating the CLIP guidance.

Figure S12. **Effect of initial scale on generation with text guided content.** Here, the training image is Van Gogh's "Starry Night", the text prompt is "medieval castle", the fill-factor is $f = 0.3$ and the strength is $\eta = 0.3$. When applying CLIP guidance only in finer scales, the global structure of the image is already set (by the previous scales) and the guidance only changes fine details and textures. When starting from coarse scales, CLIP's guidance modifies also the global structure.

E.4. Effect of the Fill-Factor and Strength Parameters

We let the user choose the values of the fill factor f and the strength parameter η, in order to achieve the desired result. The effects of these parameters are illustrated in Figure S13.

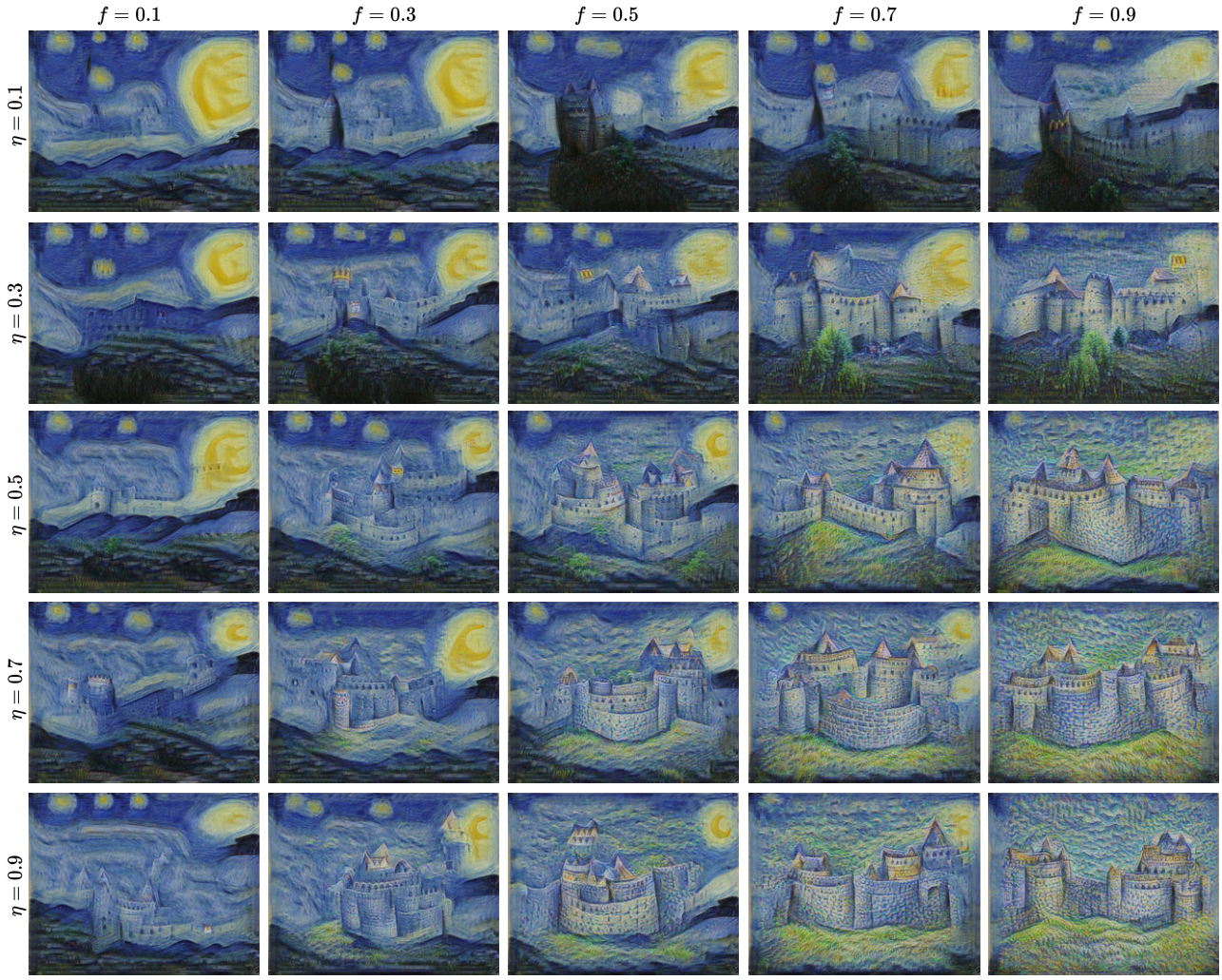

Figure S13. **Effect of user prescribed parameters on generation with text guided content.** Here, the training image is Van Gogh's "Starry Night", the text prompt is "medieval castle", and the starting scale is $s = 1$. As the fill factor f increases, a larger region of the image is affected by the text. As the strength η increases, the modification within the affected regions is more prominent.

F. Controlling Object Sizes

Manipulation of image content can be achieved by choosing the dimensions of the initial noise map at the coarsest scale, $s = 0$. For example, we can start from a noise map that has the same dimensions as the image at $s = 1$. After completing the reverse diffusion process for $s = 0$, we pass the output to $s = 1$ *without upsampling*, as the image is already at the correct size for that scale. This causes the objects in the image to appear smaller. This is because when injecting to our model the conditioning $s = 0$, it generates objects from the smallest training image. Thus, the larger the starting noise map, the smaller the objects appear relative to the image size. This is illustrated in Figure S14.

Figure S14. **Controlling object sizes.** We show the effect of the image dimensions we use relative to the conditioning s with which we supply our model. From left to right, we start from the size of scale 4,3,2,1,0.

G. Comparisons

We next provide comparisons to several competing methods on the tasks of unconditional sampling learned from a single image (Figure S15), image generation with text-guided contents (Figs. S16, S17 and S18), image generation with text-guided contents within ROIs (Figure S20), image generation with text-guided style (Figs. S21, S22 and S23), style transfer (Figure S24) and harmonization (Figure S25).

G.1. Unconditional Sampling

Figure S15. **Comparison between single-image generative models on the task of unconditional sampling.** We show random samples generated by SinGAN, ConSinGAN, GPNN and SinDDM (ours). Our results are at least on par with existing models in terms of visual quality and generalization beyond the details and object sizes appearing in the training image.

G.2. Generation with Text-Guided Content

We next compare our method to Text2Live (Bar-Tal et al., 2022) and Stable Diffusion (Rombach et al., 2022) on the task of text-guided image generation. The comparisons are shown in Figs. S16, S17 and S18. As opposed to existing methods, SinDDM is not constrained to the aspect ratio or precise object configurations of the input image. Therefore, for our method we show several samples at different aspect ratios.

Text2Live For Text2Live, we used 1000 bootstrap epochs. Furthermore, this method requires four text prompts (three prompts in addition to the one describing the final desired result, as in our method). In each example, we list all four prompts.

Stable Diffusion For Stable Diffusion, we used the same text we provided to our method. We ran the image-to-image option with different strength values. In each example we show strengths from 0.2 (left) to 1 (right) in jumps of 0.2. When the strength parameter is small, the edited image is loyal to the original image but the effect of the text is barely noticeable. When the strength parameter is large, the effect of the text is substantial, but the result is no longer similar to the original image. It should be noted that the official code of Stable Diffusion[1] yields unnatural results for the images we used for comparions (Fig. S19). We identified that this happens because of the logic which the code uses to scale the image prior to injecting it to the encoder. To overcome this issue, we instead resized the images to 512×512 before injecting them to the Stable Diffusion pipeline, and at the output of the pipeline, we scaled the results back to the original dimensions. This leads to natural looking results.

Figure S16. **Comparisons for image generation and editing guided by text.** Here we used text prompt "stars constellations in the night sky". For Text2Live we provided "stars constellations in the night sky" as the text describing the edit layer, "a desert in night with stars constellations in the night sky" as the text describing the full edited image, "a desert in night" as the text describing the input image, and "night sky" as the text describing the region of interest in the input image. For Stable Diffusion we show strengths from 0.2 (left) to 1 (right) in jumps of 0.2.

[1] https://github.com/CompVis/stable-diffusion

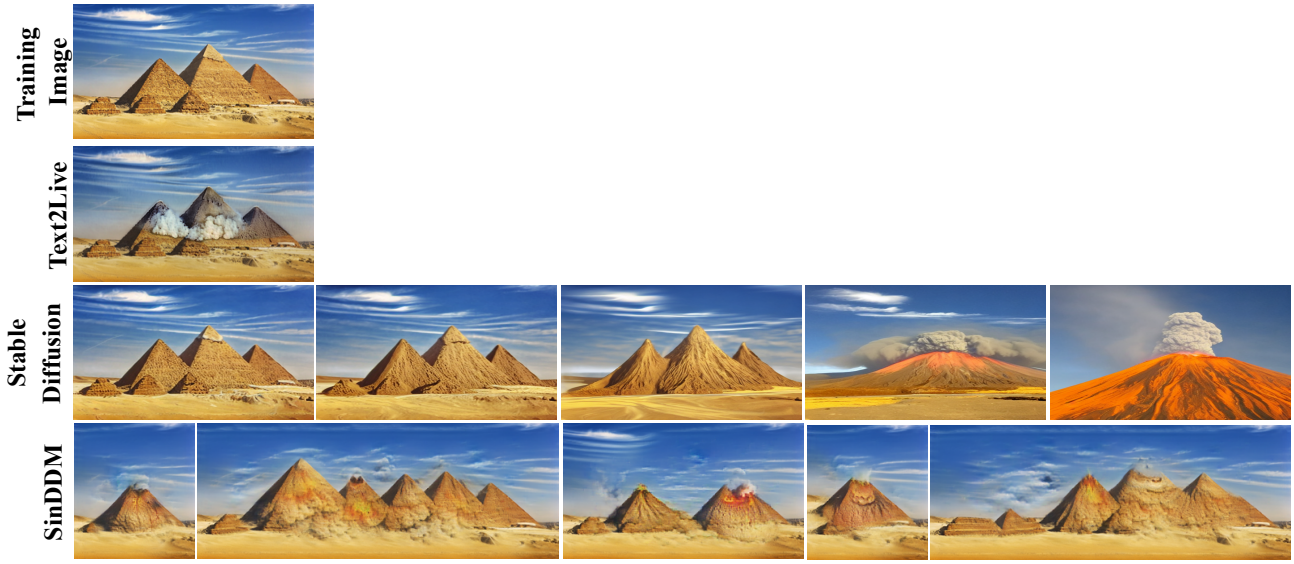

Figure S17. **Comparisons for image generation and editing guided by text.** Here we used the text prompt "volcano eruption". For Text2Live we provided "volcano eruption" as the text describing the edit layer, "volcano erupt from the pyramids in the desert" as the text describing the full edited image, "pyramids in the desert" as the text describing the input image, and "the pyramids" as the text describing the region of interest in the input image. For Stable Diffusion we show strengths from 0.2 (left) to 1 (right) in jumps of 0.2.

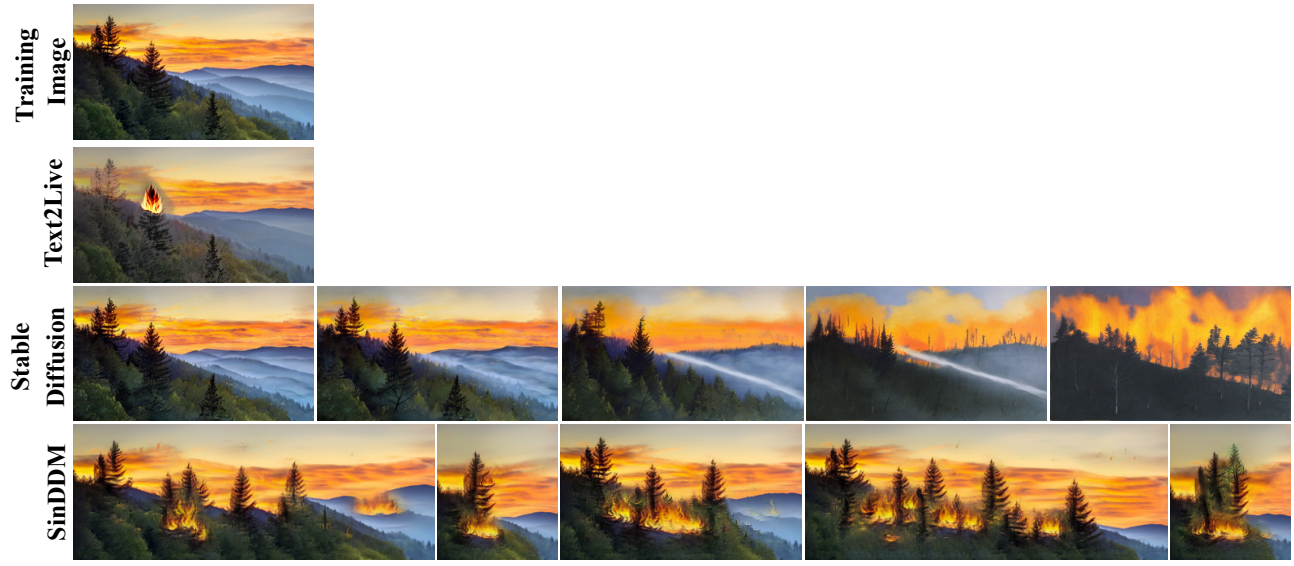

Figure S18. **Comparisons for image generation and editing guided by text.** Here we used the text prompt "a fire in the forest". For Text2Live we provided "a fire in the forest" as the text describing the edit layer, "a fire in the forest on a mountain in sunset" as the text describing the full edited image, "a forest on a mountain in sunset" as the text describing the input image, and "the forest" as the text describing the region of interest in the input image. For Stable Diffusion we show strengths from 0.2 (left) to 1 (right) in jumps of 0.2.

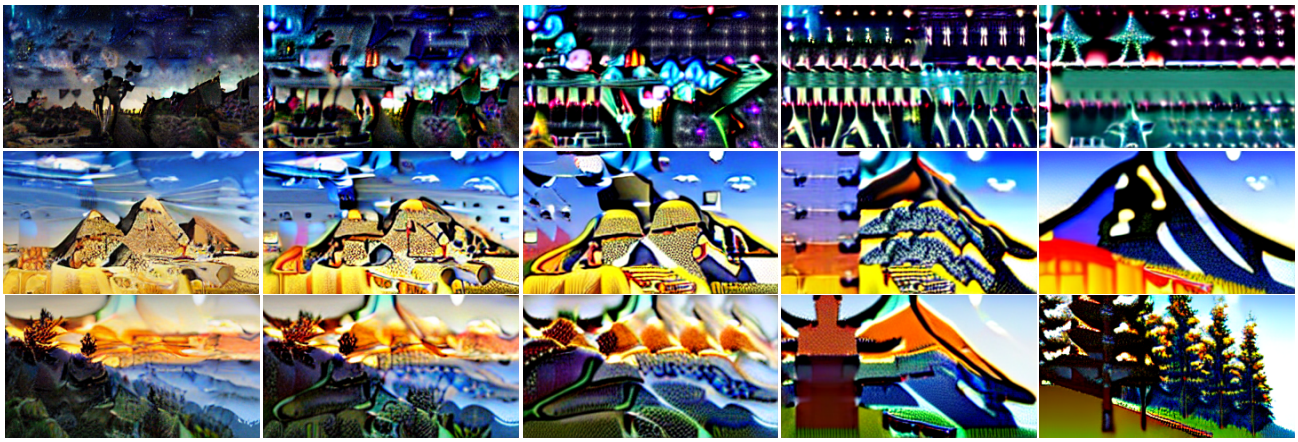

Figure S19. **Stable diffusion official implementation results.** When using our images in the image-to-image option of the official code of Stable Diffusion, we get poor non-photo-realistic results. To obtain the good results we present in Figs. S16, S17 and S18, we had to use a different resizing strategy before injecting the images into the Stable Diffusion pipeline (see text for details). Here we show the results with official code (without our modification) with strengths from 0.2 (left) to 1 (right) in jumps of 0.2.

G.3. Generation with Text-Guided Content in ROI

In Figure S20 we compare our text-guided content generation in ROI to the Stable Diffusion inpainting method (as implemented in HuggingFace diffusers[2]) and to the DALL-E 2 editing option that is accessible via their API[3]. The Stable Diffusion inpainting method creates images that are not loyal to the text input and and do not blend well with the original image. Using the DALL-E 2 API, the image needed to be cropped and reshaped to 1024×1024 prior to editing. As can be seen, the DALL-E 2 results are unrealistic and in some cases the results are not related to the input text (like in the middle example, where the DALL-E 2 results do not depict cracks).

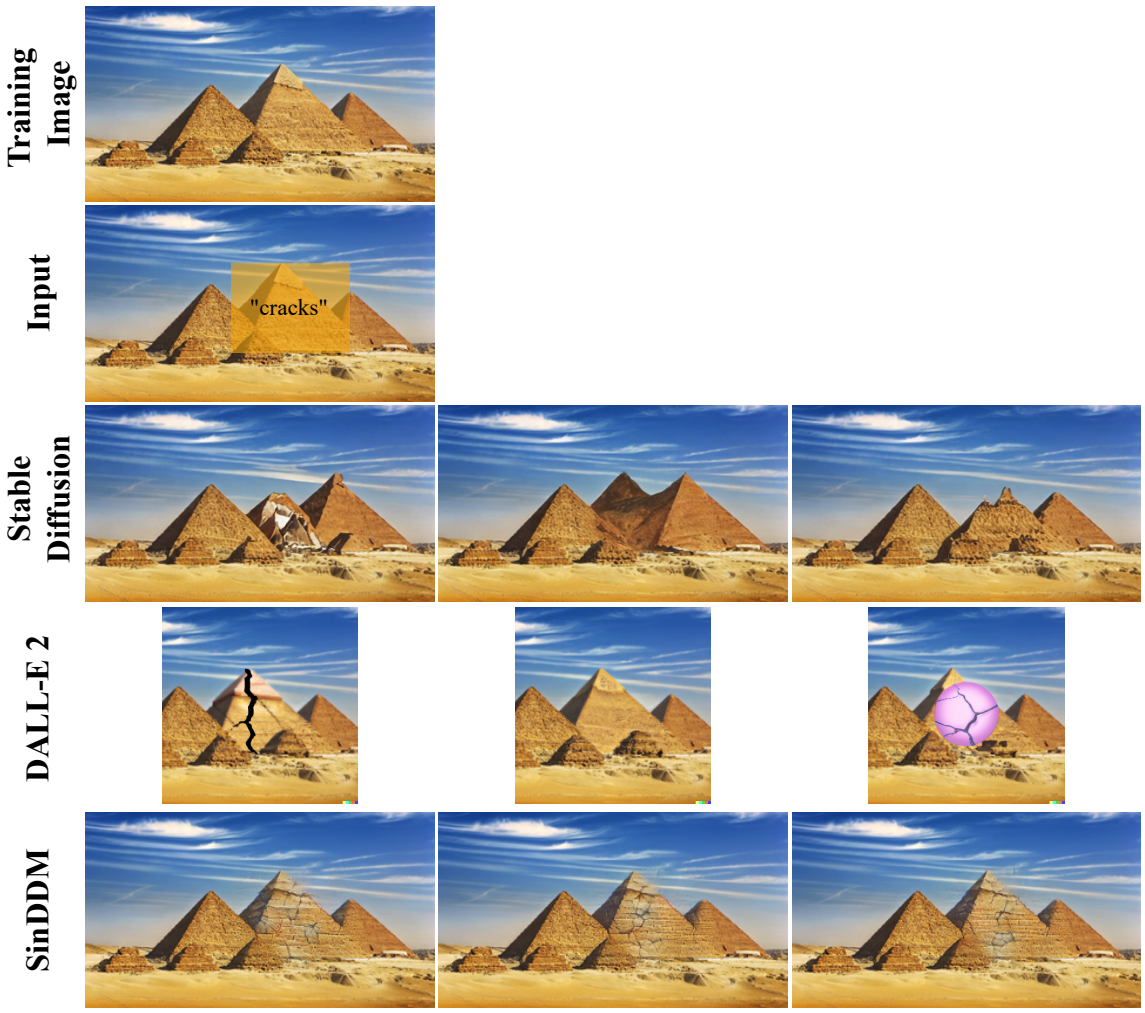

Figure S20. **Comparison of text-guided content generation in ROI.** We show comparisons to the Stable Diffusion inpainting and DALL-E 2 editing methods on the task of text-guided generation in a ROI.

[2] https://huggingface.co/docs/diffusers/using-diffusers/inpaint
[3] https://openai.com/api/

G.4. Generation with Text-Guided Style

Here we compare our method of generation with text-guided style to Text2Live and Stable Diffusion. The comparisons are shown in Figures S21, S22 and S23.

Text2Live As in App. G.2, for Text2Live, we used 1000 bootstrap epochs. Furthermore, this method requires four text prompts (three prompts on top of the one for describing the final desired result, as in our method). In each example, we list all four prompts.

Stable Diffusion For Stable Diffusion, we used our modified version of the image-to-image option that reshapes the image to 512×512 before injecting it to the Stable Diffusion pipeline. In each example we show strengths from 0.2 (left) to 1 (right) in jumps of 0.2. As before, when the strength parameter is small, the edited image is loyal to the original image but the effect of the text is barely noticeable. When the strength parameter is large, the effect of the text is substantial, but the result is no longer similar to the original image.

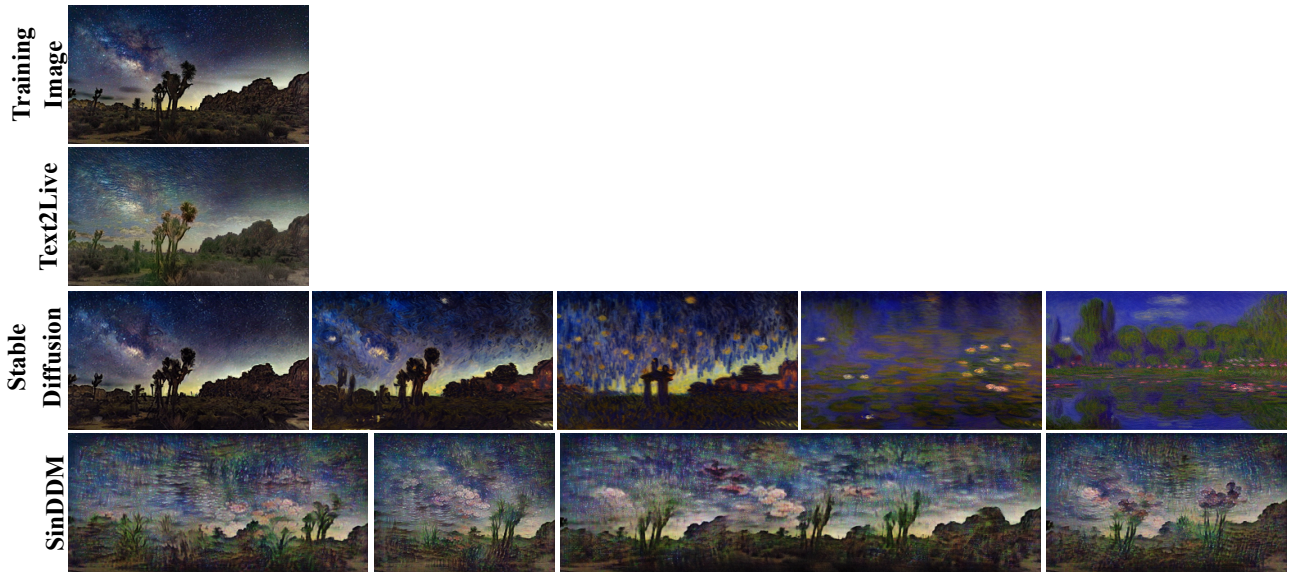

Figure S21. **Comparisons for image generation with text-guided style.** Here we used the text prompt "Monet style". For Text2Live we provided "Monet style" as the text describing the edit layer, "a desert with vegetation in night under starry sky in the style of Monet" as the text describing the full edited image and "a desert with vegetation in night under starry sky" as the text describing the input image and as the text describing the region of interest in the input image. For Stable Diffusion we show strengths from 0.2 (left) to 1 (right) in jumps of 0.2.

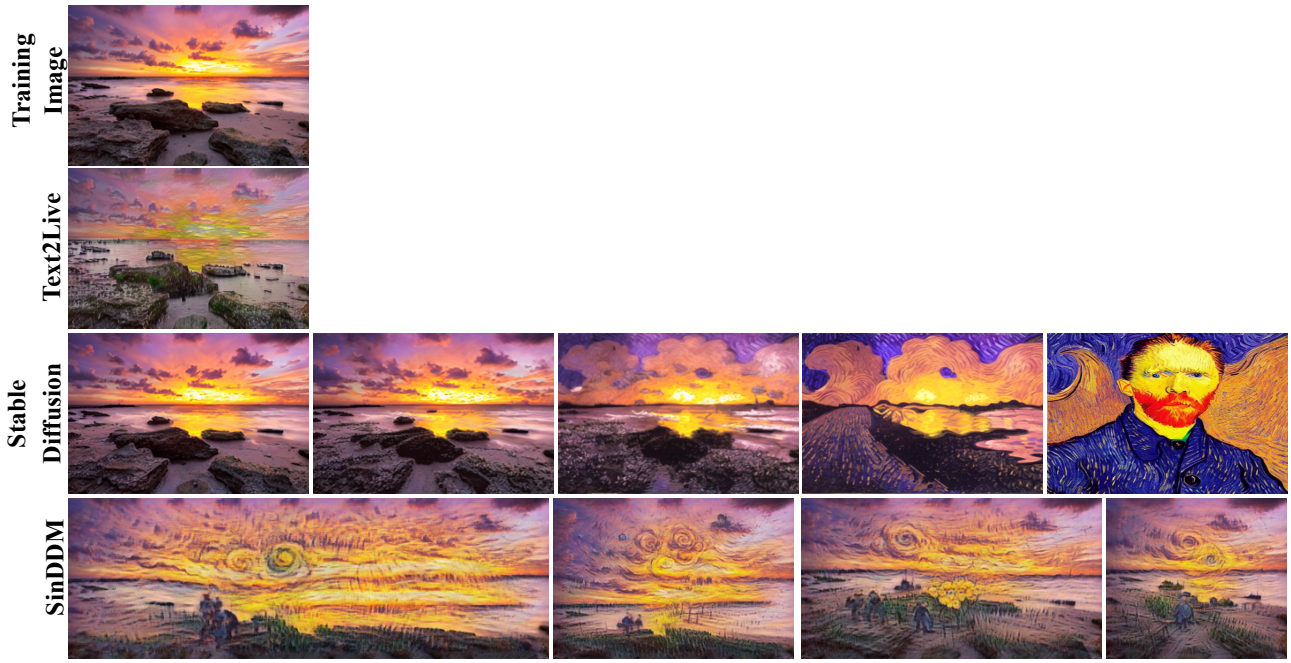

Figure S22. **Comparisons for image generation with text-guided style.** Here we used the text prompt "Van Gogh style". For Text2Live we provided "Van Gogh style" as the text describing the edit layer, "an isolated sea shore with rocks and white sand, during pink and orange sunset in the style of Van Gogh" as the text describing the full edited image and "an isolated sea shore with rocks and white sand, during pink and orange sunset" as the text describing the input image and as the text describing the region of interest in the input image. For Stable Diffusion we show strengths from 0.2 (left) to 1 (right) in jumps of 0.2.

Figure S23. **Comparison of image generation guided by style.** Here we used the text prompt "Monet style". For Text2Live we provided the same prompts as described in Figure S22 but used "Monet" instead of "Van Gogh". For Stable Diffusion we show strengths from 0.2 (left) to 1 (right) in jumps of 0.2.

G.5. Style Transfer

In Figure S24, we compare SinDDM to SinIR on the task of style transfer. As can be seen, SinDDM can generate results that are simultaneously more loyal to the content image and to the style image.

Figure S24. **Style Transfer.** Comparison between SinIR and SinDDM (our).

G.6. Harmonization

In Figure S25, we compare SinDDM to SinIR and ConSinGAN on the task of harmonization. Here SinDDM leads to stronger blending effects, but at the cost of some blur in the pasted object.

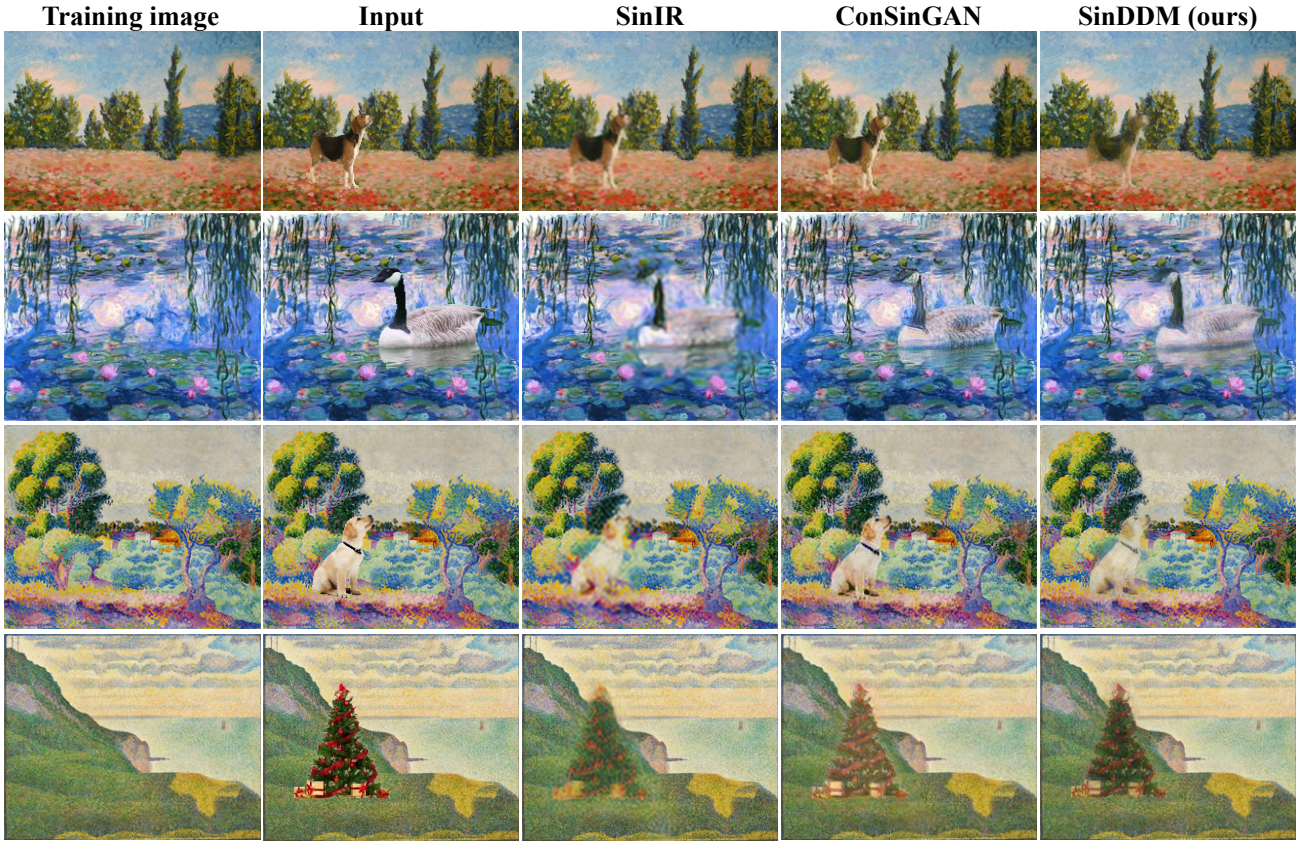

Figure S25. **Harmonization.** Comparison between SinIR, ConSinGAN and SinDDM (our). For ConSinGAN, we used the fine-tuning option provided in the code, which fine-tunes the trained model on the naively pasted image.

H. Limitations and Directions for Future Work

H.1. Misrepresentation of the Distribution of Certain Objects

Our sampling algorithm often misrepresents the true distribution of the objects in the original image. For example, in the Elephants image in Figure S26, our samples usually contain more elephants than in the original image, and less trees (more often than not, the samples do not contain a single tree). In certain images with many small objects, our sampling algorithm can fail to represent them faithfully. An extreme example is the Birds image in Figure S26, where the birds are severely under-represented in our samples. This can be solved by (1) manually tuning the initial T for each scale, and (2) modifying the sampling scheme such that it does not involve the blending with the image generated at the previous scale. Such a fix is illustrated at the bottom of Figure S26. However, for (1) we currently do not have an automatic algorithm which works well across all images, and using (2) introduces undesired blur to the samples. We therefore leave these directions for future research.

H.2. Inner Distribution Preservation Under Text Guided Content Generation

An additional limitation of our method relates to text guidance. Specifically, when using text guidance to generate an image with new content, we are limited to images that adhere to the internal distribution of the training image. This limits the new content to contain only textures which exist in the training image. For example, "fire in the forest" seen in Figure S18 looks realistic because the orange sky texture was used to create the fire.

H.3. Boundary conditions and the effect of padding

Similarly to SinGAN (see Supplementary Sec. 2 in (Shaham et al., 2019)), the type of padding used in the model influences the diversity among generated samples at the corners. Figure S27 illustrates this effect by depicting the standard deviation among generated samples for each location in the image. As can be seen, zero padding in each convolutional layer (layer padding) leads to small variability at the borders but large variability away from the borders. Padding only the input to the network (initial padding), leads to increased variability at the borders but smaller variability at the interior part of the image

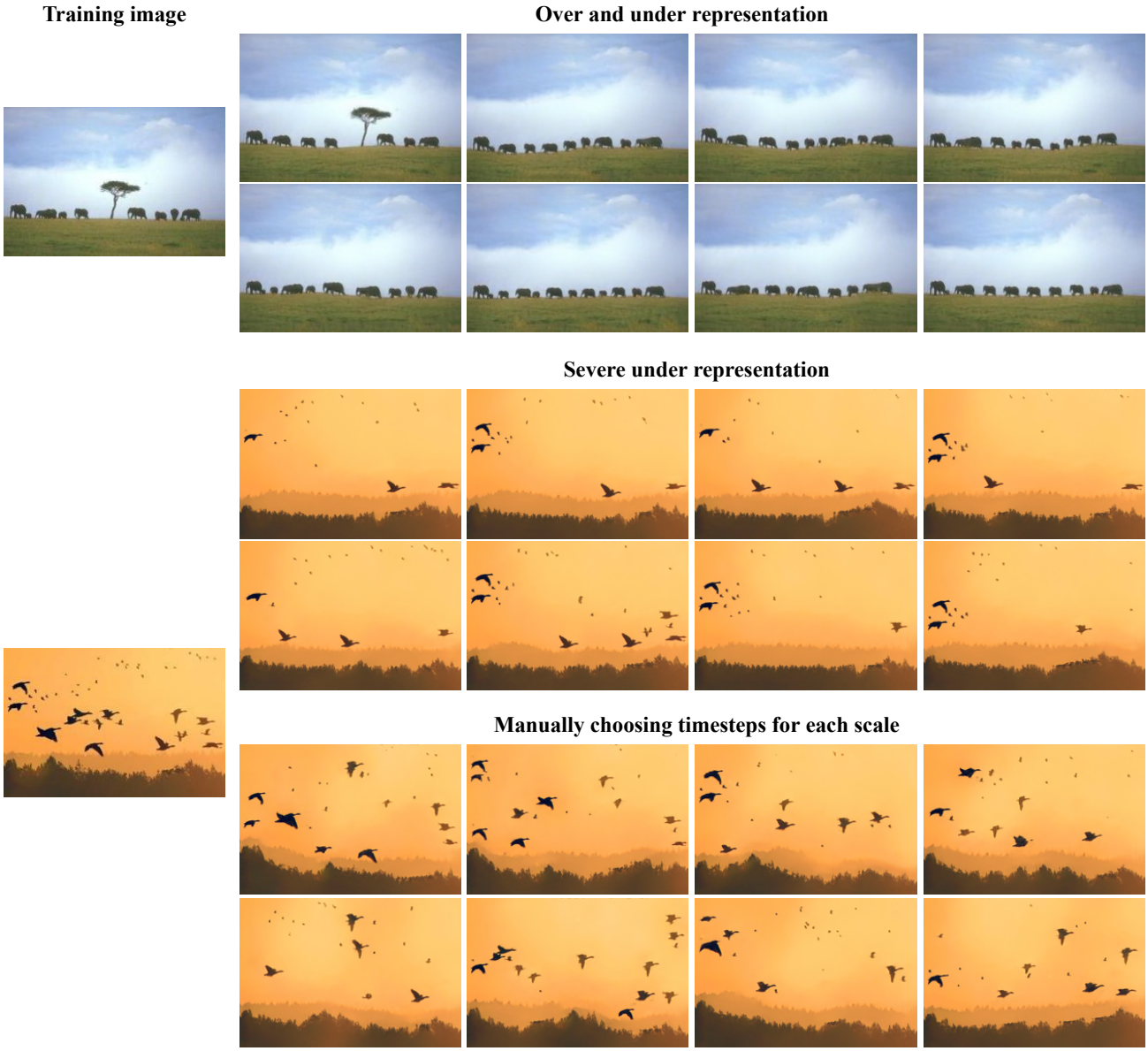

Figure S26. **Limitations.** Our model can sometimes under- or over-represent certain objects in the training image, as seen in the Elephants and Birds images. These problems can be fixed by manually choosing the initial timestep for each scale. However we do not currently have an automatic way to choose these values, which avoids these problems for all training images.

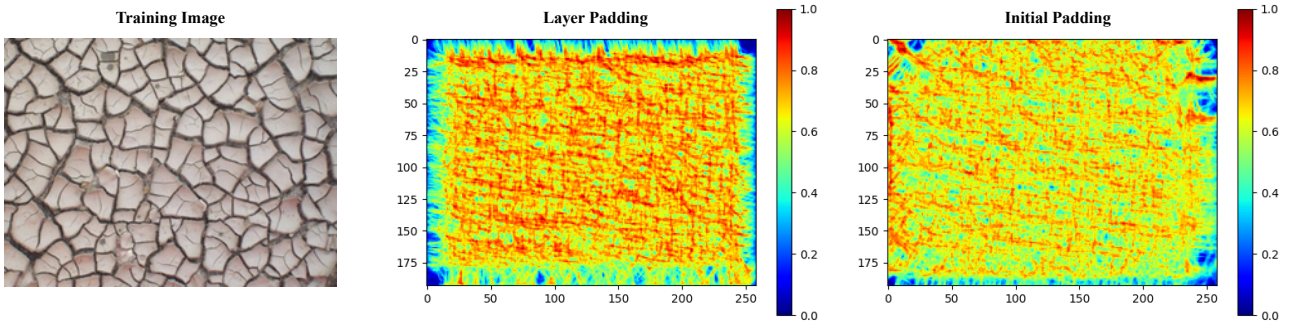

Figure S27. **Effect of padding.** The padding configuration affects the diversity between samples at the corners. Zero padding in each convolutional layer (layer padding) leads to small variability at the borders but large variability away from the borders. Padding only the input to the net (initial padding), leads to increased variability at the borders but smaller variability at the interior part of the image.

References

Bar-Tal, O., Ofri-Amar, D., Fridman, R., Kasten, Y., and Dekel, T. Text2LIVE: Text-driven layered image and video editing. In *European conference on computer vision*, 2022.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Nichol, A. Q. and Dhariwal, P. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pp. 8162–8171. PMLR, 2021.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and 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.

Shaham, T. R., Dekel, T., and Michaeli, T. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4570–4580, 2019.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.