DIRECTED DIFFUSION: DIRECT CONTROL OF OBJECT PLACEMENT THROUGH ATTENTION GUIDANCE

A PREPRINT

Wan-Duo Kurt Ma
Victoria University of Wellington
mawand@ecs.vuw.ac.nz

J.P. Lewis
Google Research*
noisebrain@gmail.com

Avisek Lahiri
Google Research
avisek@google.com

Thomas Leung
Google Research
leungt@google.com

W. Bastiaan Kleijn
Victoria University of Wellington and Google
bastiaan.kleijn@vuw.ac.nz

September 27, 2023

Keywords denoising diffusion · text-to-image generative models · artist guidance · storytelling

ABSTRACT

Text-guided diffusion models such as DALL·E 2, Imagen, eDiff-I, and Stable Diffusion are able to generate an effectively endless variety of images given only a short text prompt describing the desired image content. In many cases the images are of very high quality. However, these models often struggle to compose scenes containing several key objects such as characters in specified positional relationships. The missing capability to "direct" the placement of characters and objects both within and across images is crucial in storytelling, as recognized in the literature on film and animation theory. In this work, we take a particularly straightforward approach to providing the needed direction. Drawing on the observation that the cross-attention maps for prompt words reflect the spatial layout of objects denoted by those words, we introduce an optimization objective that produces "activation" at desired positions in these cross-attention maps. The resulting approach is a step toward generalizing the applicability of text-guided diffusion models beyond single images to collections of related images, as in storybooks. Directed Diffusion provides easy high-level positional control over multiple objects, while making use of an existing pre-trained model and maintaining a coherent blend between the positioned objects and the background. Moreover, it requires only a few lines to implement.[2]

1 Introduction

Text-to-image models such as DALL·E 2 Ramesh et al. [2022], Imagen Saharia et al. [2022], eDiff-I Balaji et al. [2022], and others have revolutionized image generation. Platforms such as Stable Diffusion Huggingface [2023] and similar systems have democratized this capability, as well as presenting new ethical challenges.

The forementioned systems generate arbitrary images simply by typing a "prompt" or description of the desired image. It is not always highlighted, however, that practical experience and experimentation are often needed if the user has a particular result in mind. Text-to-image diffusion methods often fail to produce desired results, requiring repeated trial-and-error experiments with "prompt engineering" including negative prompts, different random seeds,

*Current affiliation: NVIDIA Research

[2]Our project page: https://hohonu-vicml.github.io/DirectedDiffusion.Page

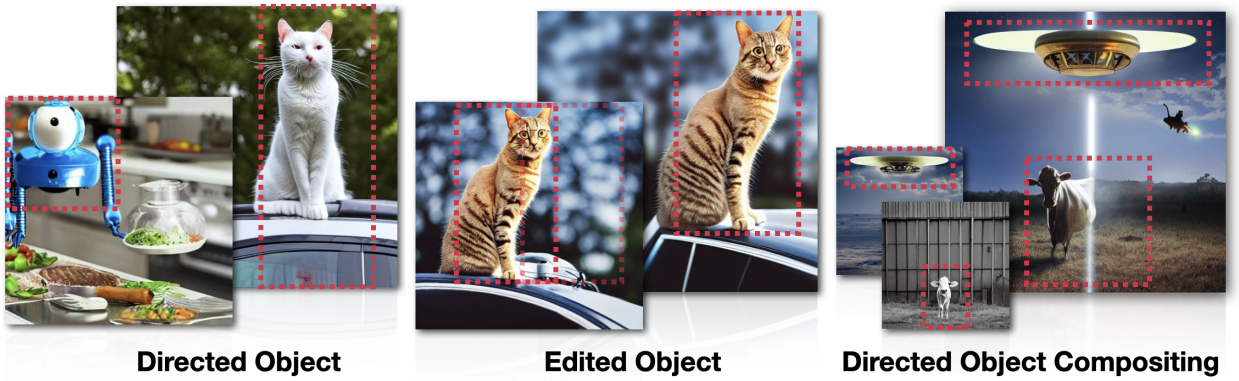


Figure 1: Directed Diffusion (DD) augments denoising diffusion text-to-image generation by allowing the position of specified objects to be controlled with user-specified bounding boxes (highlighted in red). (Left) DD generates specified objects (insect robot, cat) placed according to the given bounding boxes. We can move a synthesized object (middle), and place multiple objects in desired locations (right). All the directed objects show the appropriate "contextual" interaction (e.g. shadows) with the background.

and hyperparameters including the classifier-free guidance scale, number of denoising steps, and scheduler Smith [2022]. This is particularly true for complex prompts involving descriptions of several objects. For example, in Stable Diffusion a prompt such as *"a bird flying over a house"* fails to generate the house with some seeds, and in other cases renders both the bird and house but without the "on" relationship. The time required for experimentation is made worse if optimization or fine-tuning computation is required, especially on a per-image basis. These difficulties have led to the creation of "Prompt Marketplaces [2023]" where expert users share and sell successful settings.

The experimentation becomes prohibitive when the goal is to use the images for *storytelling*, since text-to-image methods offer no control over the required positioning of characters and objects. Prompts describing the content of an image do not indicate *where* objects should be placed, and indicating desired positions in the prompt usually fails (Figs. 1, 15). This is because the training data of text-image pairs is gathered from public sources, and people rarely annotate the location of objects in an image (*"A photo of grandmother and her cat. Grandmother is in the center of the picture, and the cat is to her left.")* because it is generally obvious to the viewer. As a consequence, extensive and tedious repeated trials are needed to obtain an image where the desired objects exist and their randomly generated positions are acceptable.

A story generally involves a character interacting with the environment or with other characters. Conveying a story with images requires creating not just images with the desired semantic content, but images with objects in suitable relative *positions* ("The thief looked back at the princess before hurrying toward the door"). The subject of "Film Grammar" Arijon [1976] seeks to codify established principles for positioning characters relative to each other and the camera. Film Language guides include principles such as *"To make things look natural, put lines, edges or faces about a third of the way across, up or down the picture 'frame'. To make them look formal, put them in the middle; and to make things seem uncomfortable, make the shot unbalanced or put it at on a slant"* https://www.learnaboutfilm.com/ [2023]. The staging principle in traditional animation Thomas and Johnston [1981] similarly recognizes the importance of controlling the placement of characters and objects.

Research is addressing some limitations of text-guided diffusion methods, including methods that define new text tokens to denote specific and consistent character or object identities, provide mask-guided inpainting of particular regions, manipulate text guidance representations, and mitigate the common failure of guidance using CLIP-like models Radford et al. [2021] to understand the target of attributes such as colors. Existing methods still generally struggle to synthesize *multiple* objects with desired positional relationships (see the Experiment section).

Our work is a further step toward guiding text-based diffusion, by introducing *coarse positional control for objects* as needed for storytelling. In this application only *coarse* positional control is needed – for example a director might instruct an actor to "start from over there and walk toward the door", rather than specifying the desired positions in floating point precision as is done in animation software packages. It is often necessary to directly control the position and interaction of *two* objects, as in the case where a character is interacting with another character or object in some specified environment, however control over exact placement of background environment objects is rarely necessary or desirable. We take inspiration from the observation that position is established early in the denoising process Liew et al. [2022] and from the fact that the cross-attention maps have a clear spatial interpretation (see Fig. 2). Our general approach is to edit the cross-attention maps for particular words during the early denoising steps, so as to concentrate activation at the desired location of the object. We introduce an optimization objective that achieves this without

disrupting the learned text-image association in the pretrained model and without requiring extensive code changes. Our method is implemented using the Python Diffusers implementation of stable diffusion (SD) and uses the available pre-trained model.

Our method makes the following contributions:

- **Storytelling.** Our method is a step towards storytelling by providing consistent control over the positioning of multiple objects.
- **Compositionality.** It provides a direct approach to "compositionality" by providing explicit positional control.
- **Consistency.** The positioned objects seamlessly and consistently fit in the environment, rather than appearing as a splice from another image with inconsistent interaction (shadows, lighting, etc.) This consistency is due to two factors. First, we use a simple bounding box to position and allow the denoising diffusion process to fill in the details, whereas specifying position with a pixel-space mask runs the risk that it may be inconsistent with the shape and position implicit in the early denoising results. Second, subsequent diffusion steps operate on the entire image and seamlessly produce consistent lighting, etc.
- **Simplicity.** Image editing methods for text-to-image models often require detailed masks, depth maps, or other precise guidance information. This information is not available when synthesizing images *ab initio*, and it would be laborious to create. Our method allows the user to control the desired locations of objects simply by specifying approximate bounding boxes.

  From a computational point of view, our method requires no training or fine tuning, and can be added to an existing text-driven diffusion model with cross-attention guidance with only a few lines of code. It requires a simple optimization of a small weight vector $\mathbf{a} \in \mathbb{R}^d, d < 77$, which does not significantly increase the overall synthesis time.

Storytelling also requires the ability to generate particular objects rather than generic instances (*this* cat rather than "any cat"), and our algorithm is complementary to methods Gal et al. [2022], Ruiz et al. [2022] that address this. For clarity, ***the examples in the paper do not make use of these other algorithms.***

2 Related Work

Denoising diffusion models Sohl-Dickstein et al. [2015], Song and Ermon [2019], Ho et al. [2020], Song et al. [2021] add Gaussian noise to the data in a number of steps and then train a model to incrementally remove this noise. Aspects of the mathematics are presented in most papers and good tutorials are available Weng [2021], so we will simply mention several high-level intuitions. The end-result of the forward process is an image effectively consisting of independent normal distributed pixels, which is easy to sample. The backward process denoises random samples from this distribution resulting in novel images (or other data). The mathematical derivation Sohl-Dickstein et al. [2015], Ho et al. [2020] somewhat resembles a hierarchical version of a VAE Kingma and Welling [2013], although with a fixed encoder and a "latent" space with the same dimensionality as the data. The fixed encoder provides an easy closed-form posterior for each step, allowing the overall loss to split into a sum over uncoupled terms for each denoising step, resulting in faster training. Song and Ermon [2019] introduced an alternate derivation building on score matching Hyvärinen [2005]. From this perspective adding noise is equivalent to convolving the probability density of the noise with the data, thus blurring the data distribution and providing gradients toward the data manifold from distant random starting points. The denoising process in Ho et al. [2020], Song and Ermon [2019] is stochastic, with an interpretation as Langevin sampling Song and Ermon [2019]. A deterministic variant Song et al. [2021] is widely used for image editing applications.

Text-to-image (T2I) models condition the image generation process on the text representation from joint text-image embedding models such as CLIP Radford et al. [2021], thereby providing the ability to synthesize images simply by typing a phrase that describes the desired image. While T2I models have employed GANs as well as autoregressive models and transformers Patashnik et al. [2021], Yu et al. [2022], Chang et al. [2023], a number of recent successful approaches use diffusion models Nichol et al. [2022], Ramesh et al. [2022], Saharia et al. [2022]. This choice reflects both the stable training of these models and their ability to learn diverse multi-subject datasets.

Our goal is to help democratize storytelling by enabling high-level open-set, zero-shot placement of several objects in T2I denoising-diffusion synthesis, *while exploiting a pre-trained model* to avoid requiring computational resources beyond those available to the typical user. To achieve this we note the following criteria:

- While a number of methods e.g. Avrahami et al. [2022a,b] use *shape masks* (silhouettes) to guide object placement, we intentionally avoid the use of detailed shape masks, since non-artist users have difficulty producing plausible outlines of objects in perspective as illustrated in **?** supplementary. In addition, it has been noted that some shape mask methods produce poor alignment of the synthesized object to the given shape Park et al. [2022] and may show poor interactions (e.g. shadows) between the object and background. Our approach makes use of a small optimization to guide the cross-attention maps to seamlessly place the directed object. We find that this gives better quality and is more reliable than approaches that directly edit the cross-attention maps see Fig. 9 in our supplementary.

- We desire the ability to control the placement of a "hero" character or object, as well as (possibly) the position of another object that the hero is interacting with, while including both physical interactions (e.g. shadows) and (to the extent possible) storytelling interactions ("the dog *chases* the ball"), along with optional description of the environment ("in the field"). Liu et al. [2022] provides control over multiple objects but requires training and does not give position control. Bar-Tal et al. [2023] demonstrates large scene composition with multiple objects but does not demonstrate specified interactions between placed objects.

- Bar-Tal et al. [2023] note the important distinction between methods that require costly training on curated datasets and those that control the generated content by manipulating the generation process of a pre-trained model. While training custom models or fine-tuning Avrahami et al. [2022b], Li et al. [2023] offers the most power and flexibility, we exclude these approaches since they often require computational resources outside the reach of typical users. For example Zhang and Agrawala [2023] reports requiring 100s of hours of GPU time for fine-tuning SD while Li et al. [2023] uses 16 V100 GPUs. We base our approach on SD since it is not proprietary.

Recent papers and preprints Li et al. [2023], Bar-Tal et al. [2023], Balaji et al. [2022], Xie et al. [2023] provide robust control over the placement of multiple objects. Balaji et al. [2022] is a powerful trained-from-scratch system that pioneered the idea of guiding object placement by per-word injection of cross-attention activation inside a coarse shape mask, controlled by a user-specified weight. Park et al. [2022] demonstrate high-quality editing of a single object driven by a combination of text and a detailed mask. They address the issue that the object shape arising from classifier-guided diffusion can conflict with the provided mask. Similar to our work, Li et al. [2023] argues that high-level box guidance has advantages over shape masks, however their approach requires fine-tuning. Xie et al. [2023] and Bar-Tal et al. [2023] are most similar to our aims. Like ours, these systems make use of a pretrained T2I model with no fine-tuning required, and both involve a relatively lightweight placement optimization during synthesis, although the particular approaches differ. While we only recently became aware of these concurrent developments, comparisons are provided in the experiments section and supplementary.

3 Method

Our objective is to create a controllable synthetic image from a text-guided diffusion model without any training by manipulating the attention from cross-attention layers, and the predictive latent noise. We use the following notation: Bold capital letters (e.g., $\mathbf{M}$) denote a matrix or a tensor, vectors are represented with bold lowercase letters (e.g., $\mathbf{m}$), and lowercase letters (e.g., m) denote scalars. Depending on the context, the superscript i on a three-dimensional tensor (e.g., $\mathbf{M}^{(i)}$) denotes a tensor slice (a matrix). In this section, this index specifies the slice of the cross-attention map associated with a particular token in the prompt. Similarly, $\mathbf{M}^{(i:j)}$ denotes the stacked slices i to j of the tensor.

The DD procedure controls the placement of objects corresponding to several groups of selected words in the prompt; we refer to these as *directed objects* and *directed prompt words* respectively. Our method is inspired by the intermediate result shown in Fig. 2 (Top). As shown in this figure, the overall position and shape of a synthesized object appears near the beginning of the denoising process, while the final denoising steps do not change this overall position but add details that make it identifiable as a particular object (cat). This observation has also been exploited in previous work such as Liew et al. [2022].

An additional phenomenon can be found in the cross-attention maps, as shown in Fig. 2 (bottom two rows). The cross-attention map has a spatial interpretation, which is the "correlation" between locations in the image and the meaning of a particular word in the prompt. For instance, we can see the cat shape in the cross-attention map associated with the word "cat". DD utilizes this key observation to spatially guide the diffusion denoising process to satisfy the user's requirement.

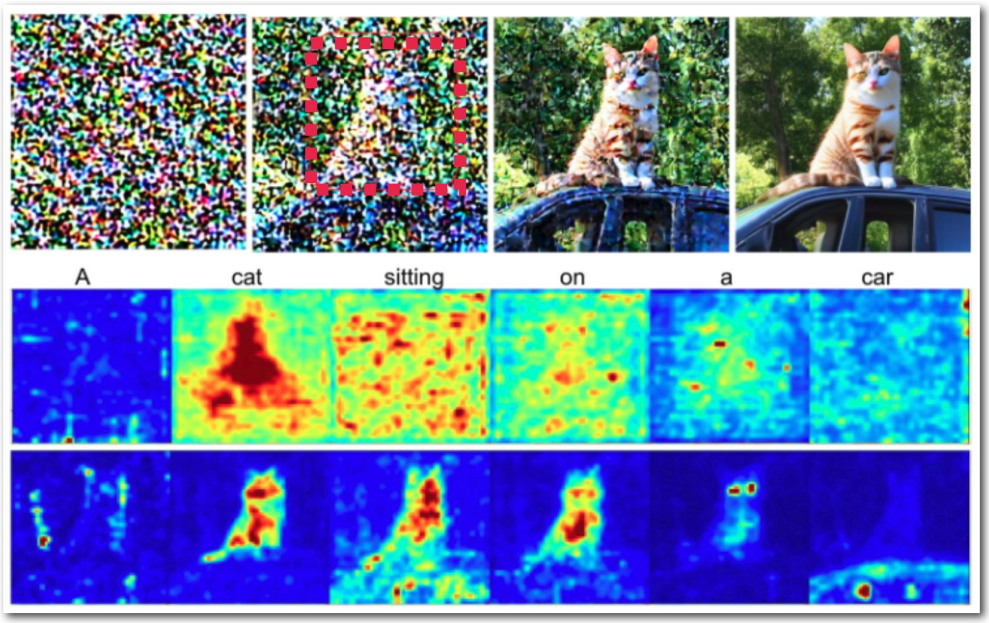


Figure 2: (Top, from left to right): The reverse SD denoising process from the initial stage to the end of process. Note that the position of the cat is evident early in the process (red box), however the details that define it as a cat are not yet clear. (Bottom): The cross-attention maps associated with each word in the prompt, where the first and second row are from the first denoising step, and the final step, respectively. The cat's shape is clearly visible in the cross-attention maps for the word "cat" (and in the maps for "sitting" and "on" in the final step).

3.1 Pipeline

Given the prompt $\mathcal{P}$ and the associated region information $\mathcal{R}$ indicating the directed objects, DD synthesizes an image $\mathbf{x}_0$ with appropriate "contextual interaction" between the objects and the remainder of the scene. It uses a pre-trained Latent Diffusion Model (LDM) Smith [2022] no further fine-tuning. The region information $\mathcal{R}$ comprises a set of parameters $\mathcal{R} = \{\mathcal{B}, \mathcal{I}\}$, denoting the bounding boxes to position the directed objects, and the directed prompt word indices in the prompt. We will describe these in Sec. 3.2.

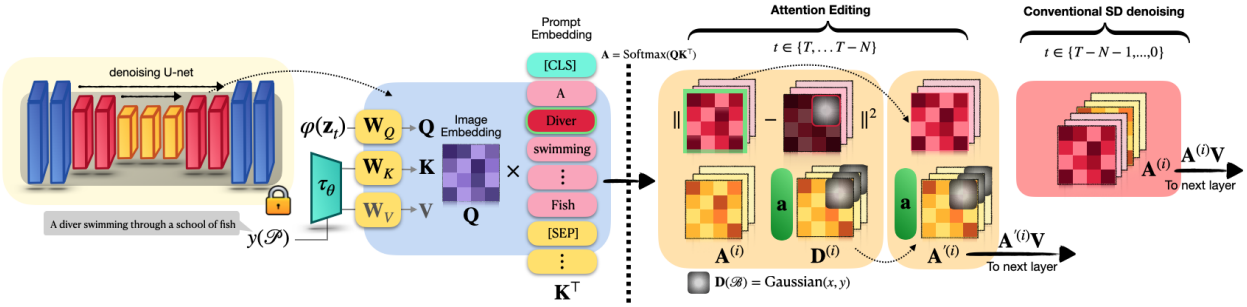


Figure 3: Directed Diffusion (DD) pipeline overview: DD divides the denoising process into initial steps where *Attention Editing* is performed, followed by refinement steps using *Conventional SD Denoising*. The goal of the *Attention Editing* stage is to optimize the cross attention map for a directed word (red, outlined in green) to approximately match a target $\mathbf{D}$ in which neural "activation" has been injected with a Gaussian fall-off inside a bounding box specified by the user. The optimization is performed by adjusting a vector ($\mathbf{a}$, green) that re-weights the trailing attention maps. Note that the optimization objective involves sulccessive two time steps. Please see the text for details.

Following conventional notation, we define $\mathbf{x}_t$ and $\mathbf{z}_t$ as the synthesized SD image and the predicted latent noise at time step t, respectively, where $t \in \{T, ..., 0\}$ in the denoising fashion. The images $\mathbf{x}_t$ are reconstructions obtained by feeding the latent $\mathbf{z}_t$ through the VAE decoder $\mathcal{D}(\cdot)$. The images $\mathbf{x}_T$ and $\mathbf{x}_0$ correspond to the Gaussian noise latent $\mathbf{z}_T$ and the final predicted latent $\mathbf{z}_0$, respectively.

The principle of this work is based on the concept *"first position the objects, then refine the results"*. This is reflected in the overall Directed Diffusion pipeline shown in Fig. 3 and also the applications detailed in Sec. 4.

Attention Editing. From a high-level perspective, this stage focuses on spatially editing the cross-attention map used for conditioning in stable diffusion. It operates during the diffusion steps $t \in [T, T-N)$ that establish the object location, where N is a hyperparameter determining the number of steps in this stage. We chose N=10 in most of our experiments. As described in Sec. 3.2, this stage modifies the cross-attention map during the first N denoising steps by amplifying the region inside $\mathcal{B}$ while down-weighting the surrounding areas through optimization (the path to the yellow regions in Fig. 3).

Conventional SD Denoising. Following the attention editing stage, this stage runs the standard SD process using classifier-free guidance Ho and Salimans [2021] over the remainder of the reverse diffusion denoising steps $t \in \{T-N, \cdots, 0\}$. Note that the only difference between the two stages is the cross-attention editing (the path to the red region in Fig. 3).

3.2 Cross-Attention Map Guidance

To achieve our goal, DD asks the user specific information about the "direction" of the object with $\mathcal{R} = \{\mathcal{B}, \mathcal{I}\}$ to guide the denoising SD process, where $\mathcal{B}$ and $\mathcal{I}$ denote the bounding box specification and the directed prompt words indices in $\mathcal{P}$, respectively. We will use the prompt *"A bear watching a flying bird"* as our specific example throughout this section and the next section.

Specifically, a bounding box $\mathcal{B} = \left\{(x, y) \mid b_{\text{left}} \times w \leq x \leq b_{\text{right}} \times w,\ b_{\text{top}} \times h \leq y \leq b_{\text{bottom}} \times h\right\}$ is the set of all pixel coordinates inside the bounding box of resolution $w \times h$. The bounding box with label i guides the location of the subject indicated by the ith prompt word. The exact location and shape of the subject results from the interaction of the Gaussian activation window inside this box with the activation $\mathbf{z}_i$ in the U-net, hence the object is not restricted in shape and may extend somewhat outside the box. In our implementation, $\mathcal{B}$ is generated based on a tuple of four scalars representing the boundary of the bounding box, denoted as $\mathbf{b} = (b_{\text{left}}, b_{\text{right}}, b_{\text{top}}, b_{\text{bottom}})$, where $b_{\text{left}}, b_{\text{right}}, b_{\text{top}}, b_{\text{bottom}} \in [0, 1]$ describe the bounding box of the directed object position expressed as fractions of the image size,

SD implements text-guided synthesis with classifier-free guidance using cross-attention between the projected embedding $\mathbf{Q}$ of the SD latent z_t and the projected embeddings $\mathbf{K}$ of the $|\mathcal{W}|$ potential words from the prompt. The product $\mathbf{QK}^T \in \mathbb{R}^{n^2 \times |\mathcal{W}|}$ (following a softmax and normalization) can be interpreted as $|\mathcal{W}|$ individual cross-attention maps $\mathbf{A}^{(i)}$ (see Fig. 2, bottom), each having a spatial interpretation of an $n \times n$ image. SD uses CLIP Radford et al. [2021] as its text encoder, which has has $|\mathcal{W}| = 77$, while $n = 64$, matching the latent spatial dimension in the current SD implementation. The cross-attention maps $\mathbf{A}^{(i)}$ are divided into a subset of *prompt maps* $\mathbf{A}^{(i)}$, $1 \leq i \leq |\mathcal{P}|$ corresponding to words in the prompt, and the remainder $i \in \mathcal{T}$ where $\mathcal{T} \coloneqq \{|\mathcal{P}|+1, \cdots, |\mathcal{W}|\}$ that do not correspond to any words but are computed to allow constant-size matrix computations. We term the latter as *trailing attention maps*. We denote the prompt words for an object to be directed with an indices set $\mathcal{I} \subset \{i | i \in \mathbb{N}, 1 \leq i \leq |\mathcal{P}|\}$. For the bear example, $\mathcal{I} = \{1, 2\}$ to denote "A" and "bear" in the prompt.

We wish to edit the cross attention maps corresponding to the "directed" words in the prompt so as to place the object at the bounding box. In one initial experiment, we used an optimization objective that encouraged the latent variables $\mathbf{z}_t$ to have high energy in the desired region during the attention editing steps. This was not stable, which might be explained by the fact that it is potentially pushing $\mathbf{z}_t$ outside the range of values encountered during training.

Thus, we wish to guide the directed cross-attention maps, but without disrupting the image-text relationship that is learned during training. One strategy is to use the network itself to project the solution near the "trained manifold", by expressing an objective at time t in terms of coarser guidance made at time $t+1$ that is refined during the $t+1 \rightarrow t$ denoising step. Another observation is that we can separate the desired coarse guidance from the directed cross attention maps by making use of the trailing maps to express the coarse guidance, since these are also included in the $\mathbf{QK}^T$ product (see supplementary for further discussion).

Given $\mathcal{B}$, we generate a modified cross-attention map using two functions:

$$\text{WeakenMask}(\mathcal{B}')_{xy} = \begin{cases} c, & (x, y) \in \mathcal{B}' \\ 1., & \text{otherwise,} \end{cases}$$

$$\text{StrengthenMask}(\mathcal{B})_{xy} = \begin{cases} f(x, y), & (x, y) \in \mathcal{B} \\ 0, & \text{otherwise,} \end{cases}$$

where $\mathcal{B}'$ is the complement of $\mathcal{B}$, and $f(\cdot)$ denotes the function that "injects attention" to amplify the region $\mathcal{B}$. In our implementation, we use a Gaussian window of size $\sigma_x = b_w/2, \sigma_y = b_h/2$ to generate the corresponding weight, where $b_w = \text{ceil}((b_{\text{right}} - b_{\text{left}}) \times w), b_h = \text{ceil}((b_{\text{top}} - b_{\text{bottom}}) \times h)$ are the width and the height of $\mathcal{B}$.

The WeakenMask$(\cdot)$ and StrengthenMask$(\cdot)$ functions "direct" selected subjects from the prompt toward specific locations of the image in the SD denoising process. StrengthenMask$(\cdot)$ inserts higher activation into the provided bounding box with a Gaussian window, while WeakenMask$(\cdot)$ attenuates the region outside $\mathcal{B}$ by multiplying by $c < 1$.

The resulting edited maps $\mathbf{D}^{(i)}$ are termed *target* maps. They are not used directly in the cross-attention denoising guidance. Instead, as described next, $\mathbf{D}^{(i)}\ \forall i \in \mathcal{I}$ provide a target for the loss in Eq. 1, while $\mathbf{D}^{(i)}\ \forall i \in \mathcal{T}$ are un-weighted sources for the weighted trailing maps created in Algo. 1 line 12.

Algorithm 1 DD Cross-Attention Editing Algorithm

1: **Input:** diffusion model DM$(\cdot)$, prompt $\mathcal{P}$, directions $\mathcal{R} = \{\mathcal{B}, \mathcal{I}\}$, timestep t
2: **Output:** Predicted conditional noise $\mathbf{z}_{\text{cond}}$ at time t from DM$(\cdot)$
3: **Parameters:** Gaussian weighting scalar c_g
4: **procedure** DDCROSSATTNEDIT(DM$(\cdot)$, $\mathcal{P}$, $\mathcal{B}$)
5: **for** $l \in$ layer(DM$(\mathbf{z}_t, \mathcal{P})$) **do**
6: **if** type$(l) \in$ CrossAttn **then**
7: $\mathbf{A} = \text{Softmax}(\mathbf{Q}_l(\mathbf{z}_t) \cdot \mathbf{K}_l(\mathcal{P})^T)$
8: $\mathbf{W} \leftarrow$ WeakenMask$(\mathcal{B}')$
9: $\mathbf{S} \leftarrow$ StrengthenMask$(\mathcal{B})$
10: $\mathbf{D}^{(i)} \leftarrow \mathbf{A}^{(i)} \odot \mathbf{W} + \cdot\mathbf{S} \quad \forall i \in \mathcal{T} \cup \mathcal{I}$
11: $\mathbf{a}^* := \arg\min_{\mathbf{a}} \mathcal{L}_{\mathbf{a}}$
12: $\mathbf{A}'^{(|\mathcal{P}|+1:77)} := \text{Diag}(\mathbf{a}^*) \odot \mathbf{D}^{(|\mathcal{P}|+1:77)}$
13: $\mathbf{z}_t \leftarrow l(\mathbf{z}_t, \mathbf{A}' \cdot \mathbf{V}_l(\mathcal{P}))$
14: **else**
15: $\mathbf{z}_t \leftarrow l(\mathbf{z}_t)$
16: **end if**
17: **end for**
18: **return** $\mathbf{z}_t$
19: **end procedure**

Algo. 1 is the core of DD. SD implements denoising with a U-net architecture, where text guidance $\tau_\theta(y(\mathcal{P}))$ from the prompt is passed to layers of the U-net through cross-attention layers, with $y(\cdot)$ and $\tau_\theta(\cdot)$ denoting the tokenizer and text embedding, respectively. In each of these cross attention layers, DD clones and modifies selected cross-attention maps $\mathbf{A}^{(i)}, \forall i \in \mathcal{I} \cup \mathcal{T}$ to form the target maps $\mathbf{D}$ (line 10) for the optimization objective (line 11, Eq. 1).

Our optimization objective seeks to find the best weighed combination of the trailing maps at time t, by means of a learned weight vector $\mathbf{a} \in \mathbb{R}^{77-|\mathcal{P}|-1}$, such that the "directed" prompt maps $\mathbf{A}^{(i)}, i \in \mathcal{I}$ best match the corresponding target maps $\mathbf{D}^{(i)}$ (with some abuse of notation):

$$\mathcal{L}_{\mathbf{a}_t} = \sum_i \|\mathbf{A}_{t-1}^{(i)} \left(\mathbf{A}_t^{(|\mathcal{P}|+1:77)} \cdot \text{Diag}(\mathbf{a}_t)\right) - \mathbf{D}^{(i)}\|^2, \tag{1}$$

$\forall t \in \{T, ..., T-N\}$, where the notation $\mathbf{A}_{t-1}^{(i)}(\cdot)$ indicates an (indirect) *functional* dependence of the cross-attention map at time $t-1$ on the weighted sum of the trailing maps edited at time t. After the weights $\mathbf{a}^* = \arg\min_{\mathbf{a}} \mathcal{L}_{\mathbf{a}_t}$ are obtained (line 11), they are used to produce re-weighted trailing maps $\mathbf{A}'$ (line 12), which then influence the overall cross-attention conditioning at time t (line 13).

To summarize the dependency flow, the trailing maps at time t are edited, and are reweighted by the optimized $\mathbf{a}$. These then influence the denoising at time t, which in turn influences the cross attention map at time $t-1$. This dependency structure involving expressing the loss at time $t-1$ is necessary because at time t the loss does not involve the weights $\mathbf{a}$, resulting in a zero gradient. In addition we believe that influencing the desired cross-attention through an intermediate denoising step helps keep the solution closer to states seen in training and thus removes the instability we found when directly optimizing for $\mathbf{z}_t$. Please refer to the Appendix for further implementation details.

4 Applications

This section demonstrates two applications that can be utilized following the DD reconstruction process. Both methods re-use information recorded from the DD processes. In the descriptions, recall that in SD algorithms the latent $\mathbf{z}_t$ (64×64 in current implementations) has direct a spatial interpretation.

4.1 Scene compositing

Composing multiple objects is a challenging problem in image generation. Complex prompts involving a relation between two objects *("bear watching bird")* often fail in current T2I models. While our DD pipeline supports the direction of multiple objects by means of multiple bounding boxes $\mathcal{B}$, in practice, the results are unreliable when the number of bounding boxes is more than two. We resolve this problem by additional editing operations.

Specifically, we first use DD to individually generate multiple objects, and record the latent information $\mathbf{z}_t^{(r)}$ of all steps for the directed objects $r \in \{1, \cdots, R\}$, where R is the total number of directed objects. Then, inspired by Liu et al. [2022], we sum and average the linear interpolation of the $\mathbf{z}_t^{(r)}$ and the latent conditioned on the original prompt $\mathcal{P}$,

$$\mathbf{z}_t \coloneqq \frac{1}{R} \sum_r w_r \times \mathbf{z}_t + (1 - w_r) \times \mathbf{z}_t^{(r)}, \tag{2}$$

$\forall t \in \{T, ..., T-N\}$, where the $w_r \in \mathbb{R}$ is a given weight, generally set to 0.1 in our experiments. After N steps of applying Eq. 2 ($N \approx 10$, see supplementary), SD is used to refine the result and produce contextual interactions.

4.2 Placement Finetuning

In some cases the artist may wish to experiment with different object positions after obtaining a desirable image. However, when the object's bounding box is moved DD may generate a somewhat different object. Our placement finetuning (PF) method addresses this problem, allowing the artist to immediately experiment with different positions for an object while keeping its identity, and without requiring any model fine-tuning or other optimization Gal et al. [2022], Ruiz et al. [2022]. Note that the PF method is not intended to produce a sequence of images for a video, as that would generally require further control over 3D object pose, camera viewpoint, etc. Please refer to the supplementary for the diagram of PF.

The algorithm has three components. First, a mask $\mathbf{M}_o$ for the directed object is obtained by thresholding the final $t = 0$ cross-attention computed by DD and clipping by the bounding box $\mathcal{B}$. At a selected time step $T-N$ the transformation $X(\cdot)$ is applied to the recorded latent and to the mask $\mathbf{M}_o$, producing a translated mask $X(\mathbf{M}_o)$. A mask $\neg\mathbf{M}_o$ is computed as the complement of $\mathbf{M}_o$ and is used to extract the background region for an initial placement-finetuned latent $\mathbf{z}'_{T-N}$.

The second component (not shown in the figure) inpaints holes in the background caused by moving the forground object. These hole regions are initialized with the transformed latent $X(\mathbf{z}_T)$ masked by $\mathbf{M}_o$. In our experiments the transformation $X(\cdot)$ is translation implemented with `torch.roll`, which causes content that is translated beyond the tensor dimension to wrap around and re-appear on the opposite side. This has the effect of initializing the hole that would otherwise appear on the opposite side with somewhat reasonable values, however more sophisticated inpainting schemes are possible here. After these initializations we perform a single iteration of adding noise followed by denoising, in order to cause the inpainted areas to have distinct detail and blend with the background.

Lastly, there are $t \in \{T-N-1, \cdots, 0\}$ steps in which the transformed latent from the current step is composited with the background,

$$\mathbf{z}'_t \coloneqq \mathbf{z}'_t \odot \neg X(\mathbf{M}_o) + X(\mathbf{z}_t) \odot X(\mathbf{M}_o),$$

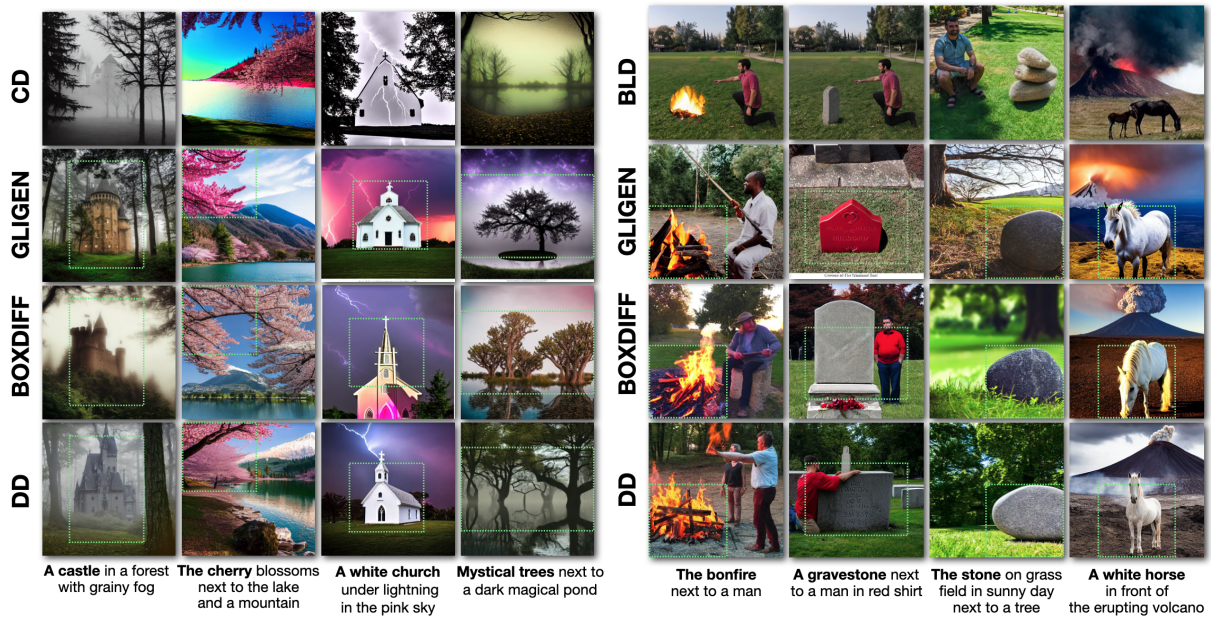
followed by denoising, resulting in a coherent refined image. N is generally set at 10 in our experiments (see supplementary). Large N better preserves the original foreground and background, while smaller N encourages more interaction between the foreground and background.

5 Experiments and Comparisons

We now present example results of Directed Diffusion. (Please see the supplementary for additional results and comparisons). Table 1 gives the quantitative result measured by the CLIP similarities between the embeddings of the prompts and the synthesized images. Our results are similar to or better than the compared methods.

Eval Type	SD	GLIGEN	CD	DD
SceneComp	0.821	0.802	0.791	**0.824**
OneMask	0.834	**0.810**	-	0.807
TwoMasks	0.842	0.802	-	**0.851**

Table 1: CLIP scores Hessel et al. [2021], Radford et al. [2021] of the three experimental categories in the Fig. 4a, Fig. 4, and Fig. 6. See the text for details.



(a) Composition comparison.

(b) Masking comparison.

Figure 4: SS@12 results from CD, GLIGEN, BOXDIFF, and our DD. The directed object is highlighted in bold in the prompt and its bounding box is shown in green (except for CD, which does not support position guidance). Please enlarge to see details including the bounding box. BLD images are reproduced from Avrahami et al. [2022a].

As described earlier, unmodified T2I methods such as stable diffusion are fallible and using such a system is *not* simply a matter of typing a text prompt. A prompt such as *"A dog chasing a ball"* may fail to generate the ball, generate a dog without legs, etc. This poses challenges for reporting qualitative evaluations: while there is not room to show a sufficient range of randomly chosen results, evaluating a *single* randomly-chosen example often results in a failure, and is not representative of how current generative AI tools are used. Users do not simply stop when they receive a bad result, but instead do trial-and-error exploration over a number of random seeds (as well as the prompt and hyperparameters Smith [2022]). To emulate this, we adopt a **SS@k** protocol, in which k images are generated with consecutive seeds following an initial randomly chosen seed, and the best-of-k image is subjectively selected.

5.1 Comparison: Scene compositing

As mentioned earlier, synthesizing and controlling several objects in an image is a challenge for many T2I methods. Common problems including missing objects, incorrectly colored objects, and incorrectly positioned objects.

In Fig. 4a, we showcase our results alongside the results obtained from the recommended configurations from the Composable Diffusion (CD) Colab demo[3] and the public implementations of BOXDIFF Xie et al. [2023] and GLIGEN Li et al. [2023]. The comparison clearly reveals that the directed objects, such as the castle, cherry blossoms, church, and trees, exhibit greater visibility in our approach compared to CD. Additionally, our results demonstrate better fidelity to the prompt (castle with fog, pond is dark) and less information blending between objects (e.g., church and mountain are not pink).

[3]https://colab.research.google.com/github/energy-based-model/Compositional-Visual-Generation-with-Composable-Diffusion-Models-PyTorch/blob/main/notebooks/demo.ipynb

5.2 Comparison: One Object

Methods for placing an object in another image can suffer from inconsistency between the object and the background environment. In Fig. 4b, we compare our DD result with Blended Latent Diffusion (BLD) Avrahami et al. [2022a], a text-driven editing method that guides object placement using using a mask. DD uses the bounding box $\mathcal{B}$ to direct the object, which can be seen as analogous to a simple form of mask.

In this comparison, note that the results shown for BLD are reproduced from the original paper, whereas we guessed a prompt to generate roughly similar images. In our results it is evident that DD generates strong and realistic interactions between the directed object and the background, including the man touching the gravestone, the shadows cast on the grass, the occlusion of the volcano by the horse, and the man's hand that catches on fire :) We also emphasize that BLD addresses a different (and difficult) purpose of editing real images. The comparison here is intended simply to highlight the realistic interactions between the directed object and the background that are obtained using our method.

5.3 Placement finetuning

Fig. 5 shows results of the placement finetuning method. Fig. 5 (top) uses the sliding window associated with the word "castle". Using PF (middle), the castle identity is preserved, while the surrounding environment such as the trees and lake reflection are reasonably synthesized. The bottom row shows the reconstruction of the latent image before and after transformation. The bird is successfully re-positioned while maintaining coherent interaction with the background (E.g., reflection, waves).

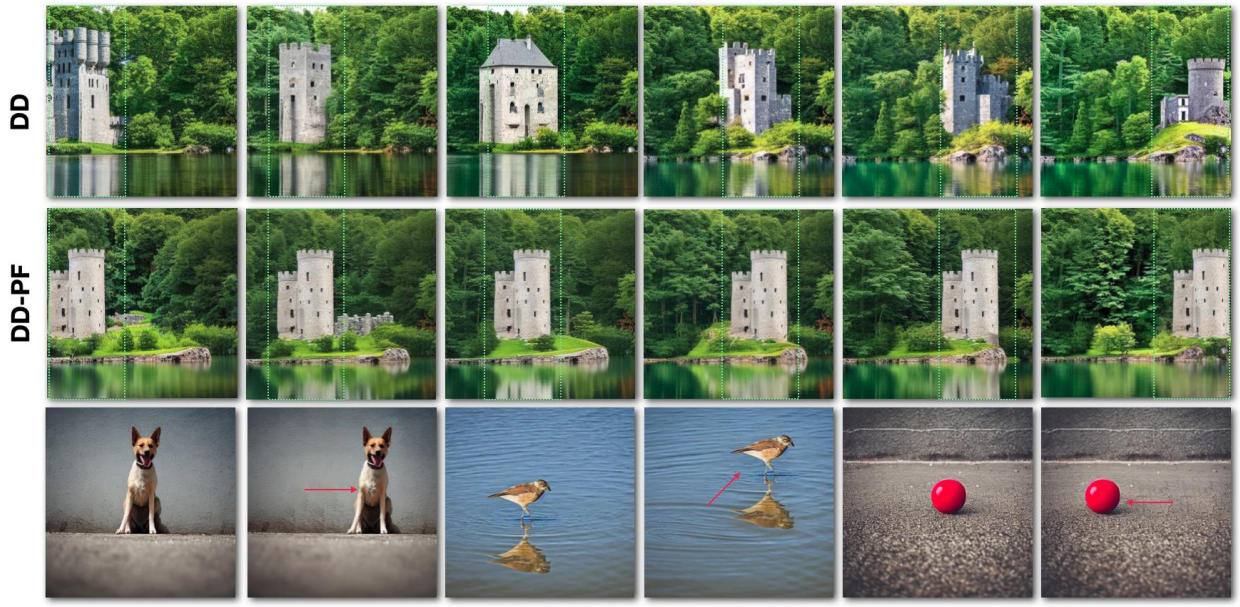


Figure 5: Comparison between DD-PF (Placement finetuning) and DD. The first row shows the DD result based on the sliding bounding box associated with castle from left to right. The second row is the PF alternative. The third row shows using PF to move the dog, bird, and sphere, respectively.

5.4 Comparison: Two Objects

Fig. 6 compares the results of several methods in placing two objects. In this task, a core consideration is scene consistency and the "correlation" between the directed objects. Our results show natural interactions such as the running dog touching the red ball, and the red cube and blue sphere supported naturally in a shopping basket.

5.5 Comparison: object interactions from prompt verbs

Fig. 7 showcases the ability of DD to (sometimes) convey interaction between objects. The figure compares two prompts: *"A white dog **and** a red ball"* and *"A white dog **chasing** a red ball."* While both results show the white dog

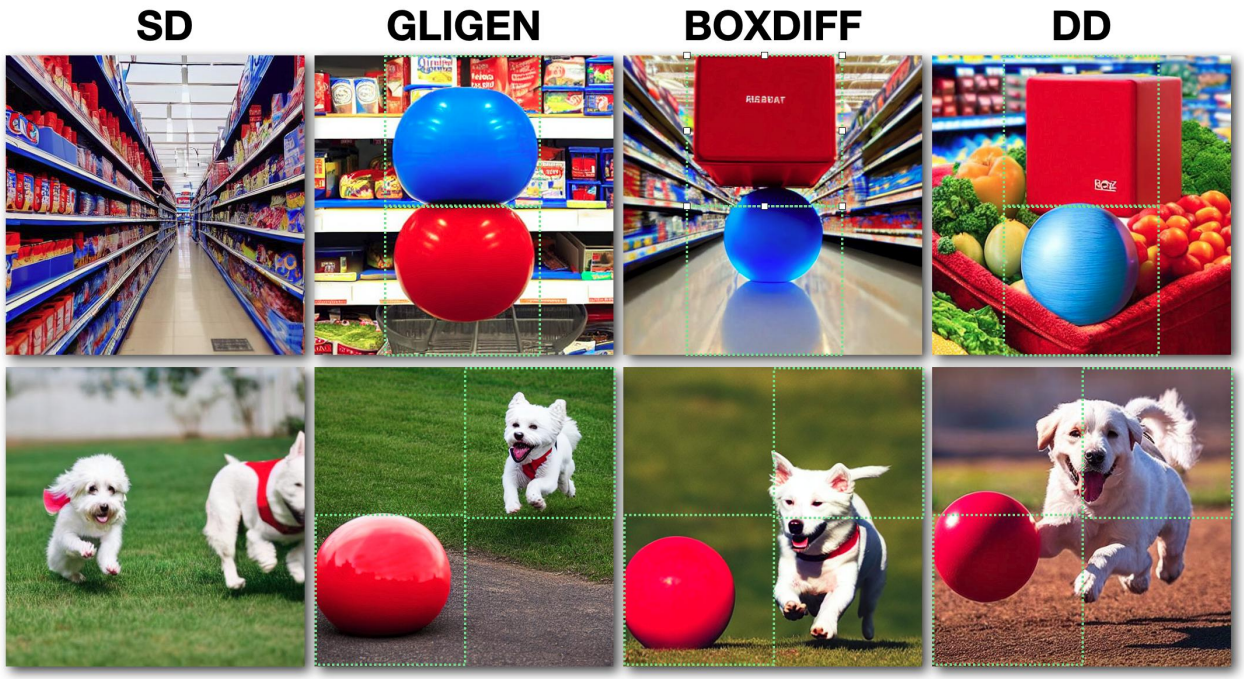


Figure 6: SS@12 results from SD, GLIGEN, BOXDIFF, and DD. The prompts are (first row) "A **red cube** above **blue sphere** in the supermarket", (second row) "A **white running dog** chasing after a **red ball**." The directed object is denoted in bold and its bounding box is shown in green. Please enlarge to see details including the bounding box.



Figure 7: SS@12 results of the prompt *"a white dog S* a red ball"*, where the token S* is replaced by *"and"* or *"chasing"* in the first row and second row, respectively. In the second row the dog more frequently appears to be running rather than sitting, consistent with the verb "chasing". Note the results of each column share the same random seed.

and red ball consistently, it is evident that the "chasing" prompt better conveys the action of this verb, e.g. the dog is running rather than sitting. Note however that this capability is bounded by CLIP's limited understanding of language structure Feng et al. [2022] and will certainly fail with more complex sentences.

6 Limitations and Conclusion

Storytelling with images requires directing the *placement* of important objects in each image. Our algorithm, directed diffusion, is a step toward this goal. The algorithm is simple to implement, requiring only a few lines of modification

of a widely used library Diffusers. Directed Diffusion inherits imitations of stable diffusion, including the need for trial-and-error exploration mentioned earlier. Some random seeds fail to produce the desired subject or produce distorted objects such as animals with missing limbs. In common with methods such as Meng et al. [2022], Liew et al. [2022], it is necessary to specify the number of steps over which editing is active. Objects do not strictly respect the provide bounding box and may extend outside it. On the other hand, this is an advantage as well as a disadvantage, since forcing the object to closely follow the box can result in an unnatural appearance (Fig. 9 in supplementary). Significant additional advances will be needed before video storytelling is possible. However, our method may be sufficient for the creation of storybooks, comic books, etc. when used in conjunction with other existing tools.

7 Acknowledgements

We thank Jason Baldridge for helpful feedback.

References

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *CoRR*, abs/2204.06125, 2022.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. *CoRR*, abs/2205.11487, 2022.

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, Tero Karras, and Ming-Yu Liu. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *CoRR*, abs/2211.01324, 2022. doi:10.48550/arXiv.2211.01324.

Huggingface. Stable diffusion 1 demo, 2023. URL `https://huggingface.co/spaces/stabilityai/stable-diffusion-1`. Accessed: 2023-01-01.

E. Smith. A traveler's guide to the latent space, 2022. URL `https://www.notion.so/A-Traveler-s-Guide-to-the-Latent-Space-85efba7e5e6a40e5bd3cae980f30235f`.

Marketplaces. Best stable diffusion prompts for ai art; top 5 list marketplaces, 2023. URL `https://csform.com/best-prompt-marketplaces-for-ai-art`.

Daniel Arijon. *Grammar of the Film Language*. Focal Press, 1976.

https://www.learnaboutfilm.com/. Telling your story: Film language for beginner filmmakers, 2023. URL `https://www.learnaboutfilm.com/film-language`.

Frank Thomas and Ollie Johnston. *Disney animation : the illusion of life*. Abbeville Press New York, 1st ed. edition, 1981. ISBN 0896592332 0896592324.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proc. ICML*, 2021.

Jun Hao Liew, Hanshu Yan, Daquan Zhou, and Jiashi Feng. Magicmix: Semantic mixing with diffusion models. *CoRR*, abs/2210.16056, 2022.

Diffusers. Diffusers library, 2023. URL `https://github.com/huggingface/diffusers`.

Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H. Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. *CoRR*, abs/2208.01618, 2022.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. *CoRR*, abs/2208.12242, 2022.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *NeurIPS*, volume 32, 2019.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33, 2020.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Lilian Weng. What are diffusion models?, 2021. URL `https://lilianweng.github.io/posts/2021-07-11-diffusion-models/`.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *J. Mach. Learning Research*, 6: 695–709, 2005.

Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In *ICCV*, 2021.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Parti: Pathways autoregressive text-to-image model, 2022. URL `https://github.com/google-research/parti`.

Huiwen Chang, Han Zhang, Jarred Barber, Aaron Maschinot, José Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T. Freeman, Michael Rubinstein, Yuanzhen Li, and Dilip Krishnan. Muse: Text-to-image generation via masked generative transformers. *CoRR*, abs/2301.00704, 2023.

Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. GLIDE: towards photorealistic image generation and editing with text-guided diffusion models. In *ICML*, 2022.

Omri Avrahami, Ohad Fried, and Dani Lischinski. Blended latent diffusion. *CoRR*, abs/2206.02779, 2022a.

Omri Avrahami, Thomas Hayes, Oran Gafni, Sonal Gupta, Yaniv Taigman, Devi Parikh, Dani Lischinski, Ohad Fried, and Xi Yin. Spatext: Spatio-textual representation for controllable image generation. *CoRR*, abs/2211.14305, 2022b. doi:10.48550/arXiv.2211.14305.

Dong Huk Park, Grace Luo, Clayton Toste, Samaneh Azadi, Xihui Liu, Maka Karalashvili, Anna Rohrbach, and Trevor Darrell. Shape-guided diffusion with inside-outside attention, 2022. URL `https://arxiv.org/abs/2212.00210`.

Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B. Tenenbaum. Compositional visual generation with composable diffusion models. In *ECCV*, 2022.

Omer Bar-Tal, Lior Yariv, Yaron Lipman, and Tali Dekel. Multidiffusion: Fusing diffusion paths for controlled image generation. *CoRR*, abs/2302.08113, 2023.

Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. GLIGEN: open-set grounded text-to-image generation. *CoRR*, abs/2301.07093, 2023.

Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023.

Jinheng Xie, Yuexiang Li, Yawen Huang, Haozhe Liu, Wentian Zhang, Yefeng Zheng, and Mike Zheng Shou. Boxdiff: Text-to-image synthesis with training-free box-constrained diffusion. *CoRR*, abs/2307.10816, 2023.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPScore: a reference-free evaluation metric for image captioning. In *EMNLP*, 2021.

Weixi Feng, Xuehai He, Tsu-Jui Fu, Varun Jampani, Arjun R. Akula, Pradyumna Narayana, Sugato Basu, Xin Eric Wang, and William Yang Wang. Training-free structured diffusion guidance for compositional text-to-image synthesis. *CoRR*, abs/2212.05032, 2022.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *Int. Conf. on Learning Representations (ICLR)*, 2022.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Katherine Crowson. k-diffusion. `https://github.com/crowsonkb/k-diffusion`, 2022.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Proc. NeurIPS*, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

Raphael Tang, Linqing Liu, Akshat Pandey, Zhiying Jiang, Gefei Yang, Karun Kumar, Pontus Stenetorp, Jimmy Lin, and Ferhan Ture. What the daam: Interpreting stable diffusion using cross attention, 2022.

This supplementary presents general **implementation details**, further details on the implementation of placement finetuning and scene compositing, an **ablation** on the core optimization method, more extensive **qualitative comparisons** against several alternative methods, several further **quantitative evaluations** along with a discussion of their **limitations**. Following the evaluations we give several comments on how the reproducibility guidelines are addressed in the paper.

8 Implementation

Here we discuss our implementation in more detail. DD is implemented with PyTorch 2.01 Paszke et al. [2017] and the DIFFUSERS library version 0.14 from Huggingface Diffusers. We use the pretrained stable diffusion model CompViz/stable-diffusion-v1-4 Rombach et al. [2022] for all our experiments. The classification free guidance scale is 7.5. The Linear Multistep Scheduler implemented in diffusers are used for discrete beta schedules based on k-diffusion Crowson [2022], Karras et al. [2022]. The examples used 50 denoising steps. The default PyTorch random generator was used. The seed was initialized at 0 (1 in the case of BoxDiff and incremented and incremented through each experiment. The experiments were performed on a single Linux OS machine containing a consumer NVIDIA 3090 GPU (24GB memory). Generating a 512x512 image takes roughly 10 seconds on this machine.

The attention editing stage of the DD pipeline (Fig. 3 and Eq. 1 in the main text), and the scene compositing (Scene compositing section and Eq. 2 in the main text), both involve a choice of the number of editing steps N. This choice is informed by the rate of change of the predicted noise between the steps. Fig. 8 illustrates the norm $\|\nabla_t \mathbf{z}_t\|$ of the gradient of the latent with respect to the time steps.

As can be seen, there is an initial period of $\approx$ 10 time steps labeled as 40 in x-axis in which the norm is rapidly decreasing, indicated by a yellow arrow in the figure. During this period the initial shape of objects are established. This is seen in Fig. 2 in the main text, which visualizes reconstructions from the VAE decoder $\mathcal{D}(\cdot)$ and the cross-attention maps $\mathbf{A}^{(i)}, \forall i \in \{1, ..., |P|\}$ at several time steps during the denoising.

Following this initial period, there is a subsequent stage in which the latents change more slowly as fine details are filled in (visualized with the green arrow). We run the editing operations during the initial "yellow" phase, while conventional SD refinement is used in the "green" phase. For nearly all experiments we empirically selected N=10 (indicated with the star in the figure) as the time step to switch from editing to refinement.

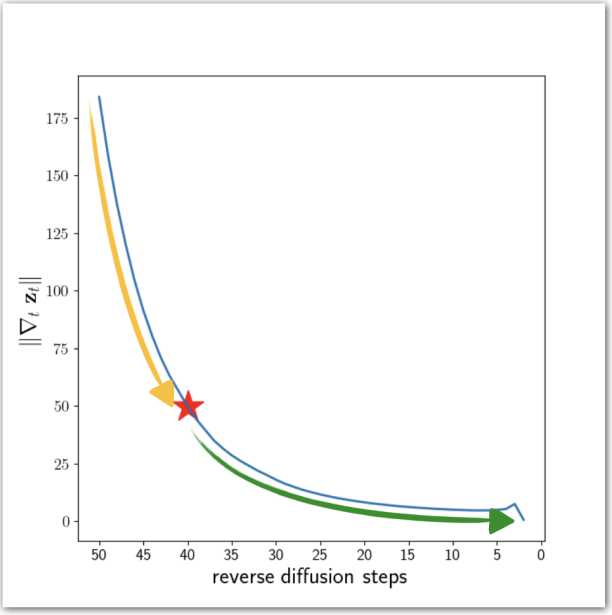


Figure 8: The norm of the gradient of latents with respect to time, plotted across time steps. The yellow and the green arrows indicate two different phases that correspond to our editing and refinement stages.

The optimizer Adam is used with the learning rate 0.0005 to optimize the Eq.(1) in the first stage. The vector $\mathbf{a}$ is initialized with a uniform distribution in the interval $[0, c]$, where generally c=0.15 in our reported experiments. A higher constant tends to lead to an empty background, while very low values fail to guide the directed object.

9 Analysis and Ablation

Several concurrent papers direct object placement by manipulating the cross attention from tokens corresponding to the directed objects. DD adopts an unusual approach of directing objects by doing a small optimization to reweight "trailing attention maps". Here we give further intuition on this choice and present evidence in the form of an ablation where the optimization is removed.

A number of text-to-image models use CLIP Radford et al. [2021] embeddings for the text guidance. CLIP accepts an input text representation with up to 77 tokens, including the initial "[CLS]" token representing the sentence level classification. The number of tokens $|\mathcal{P}|$ in the prompt $\mathcal{P}$ is generally less than this number. We call the remaining $\mathcal{T} \in \{k||\mathcal{P}| < k \leq 77-|\mathcal{P}|-1\}$ attention maps corresponding to the non-prompt tokens the *trailing attention maps* $\mathbf{A}^{(i)}, \forall i \in \mathcal{T}$. Note that the output embedding for [CLS] is not used in our approach.



Figure 9: The number of trailing attention maps (5, 10, 15, 20, maps on the vertical axis) versus the number of attention map editing steps (1, 3, 5, 10, and 15 steps on the horizontal axis). The prompt is *"A photo of a squirrel eating a burger"* with the directed object "burger" positioned at the bottom left. The best results are obtained with an intermediate number of editing steps and edited trailing attention maps, as in the case of the image with the red border.

We edit the trailing maps by injecting Gaussian-falloff activations inside the user-specified bounding box, followed by a small optimization (Eq. 1 in the main text) to find best weights for the trailing maps, resulting in the targets $\mathbf{D}^{(i)}$ in section 3.2 and Algo. 1 in the main text.

It may not be initially obvious why the trailing attention maps should affect object placement, nor why we would want to use them. While the DIFFUSERS library implements multi-head attention and a variety of conditioning mechanisms, the effect of the trailing maps can be seen from considering a generic cross attention scheme. The cross attention

maps $\mathbf{A}^{(i)} = \text{Softmax}(\mathbf{Q}\mathbf{K}^{(i)^T}), \forall i \in \{1, \cdots, |\mathcal{W}|\}$ where $\mathbf{Q} \in \mathbb{R}^{n^2 \times L}$ and $\mathbf{K} \in \mathbb{R}^{|\mathcal{W}| \times L}$, are essentially the cross-covariance of each "pixel" of the projected embedding of the SD latent z_t with projected embeddings of the $|\mathcal{W}| = 77$ potential words from the prompt. Consider the standard cross-attention computation $\mathbf{C} = \text{Softmax}(\mathbf{Q}\mathbf{K}^T) \cdot \mathbf{V}$, where $\mathbf{V} \in \mathbb{R}^{|\mathcal{W}| \times L}$ is a second projected text embedding of dimension L for each token. The resulting conditioning matrix $\mathbf{C}$ has dimension $\mathbb{R}^{n^2 \times L}$, while a single "pixel" of this matrix is a convex sum (from the softmax) of all the embeddings in $\mathbf{V}$, *including those in the trailing words.*

Manipulating the trailing maps thus influences the conditioning used in the (classifier-free) text guidance. The optimization scheme in Eq. 1 in the main text uses trailing maps to provide a clean separation between the desired object location (specified via the trailing maps at time step t) and the computed attention map at time $t-1$. As stated in the main text, we believe that this intermediate denoising step serves the purpose of moving our position guidance near the range of values expected by the pretrained SD model. The optimization exploits the fact that the maps do not have identical effects (ultimately due to the random initialization of the model), to find a weighted combination that successfully produces the desired object direction.

Fig. 9 shows an ablation in which the optimization in Eq. 1 is removed and the Gaussian-falloff activations are instead directly injected in the cross attention maps for the directed prompt tokens as well as a specified number of trailing maps. The figure shows the number of trailing maps plotted versus the number of attention editing time steps. When the number of edited trailing attention maps is small (e.g., first row, $|\mathcal{T}| = 5$), the image $\mathbf{x}_0$ is barely modified. In contrast, when the number of edited trailing attention maps is too large (last row), $|\mathcal{T}| = 20$), $\mathbf{x}_0$ is missing the background and satisfies only the bounding box information. Using an intermediate number of edited trailing attention maps results in the image simultaneously portraying the desired semantic content at the desired location(s).

We initially experimented with implementing DD by directly injecting the Gaussian-falloff activations in the cross-attention maps for the directed prompt words and a specified number of trailing attention maps, however (as can be seen with this ablation) this was somewhat unstable and required a grid search to obtain good results. To avoid this, we instead formulate Eq. 1 to automatically obtain the best combination of the trailing attention maps to control the directed object.

Here we show more experiments with one directed object with a guided bounding box positioned in the four quadrant and the slide window style.

Fig. 10 shows examples of directing objects to lie in each of four image quadrants as well as at the image center, with the directed objects "a large cabin", "an insect robot", and "the sun". Notice for example that the "sun" examples show the effect of the sun in the correct quadrant but in different contexts. The 3rd quadrant and the center show the real sun at the edge of the house roof, while the other images show the reflection of the sun such as from a window (e.g., 1st, 4th quadrant), or the wall (e.g., 2nd quadrant).

Fig. 11 shows the result of moving an object from left to right. All the results are generated with a bounding box with the full image height and 40% of the image width. The directed objects are "a stone castle", "a white cat", and "a white dog", respectively.

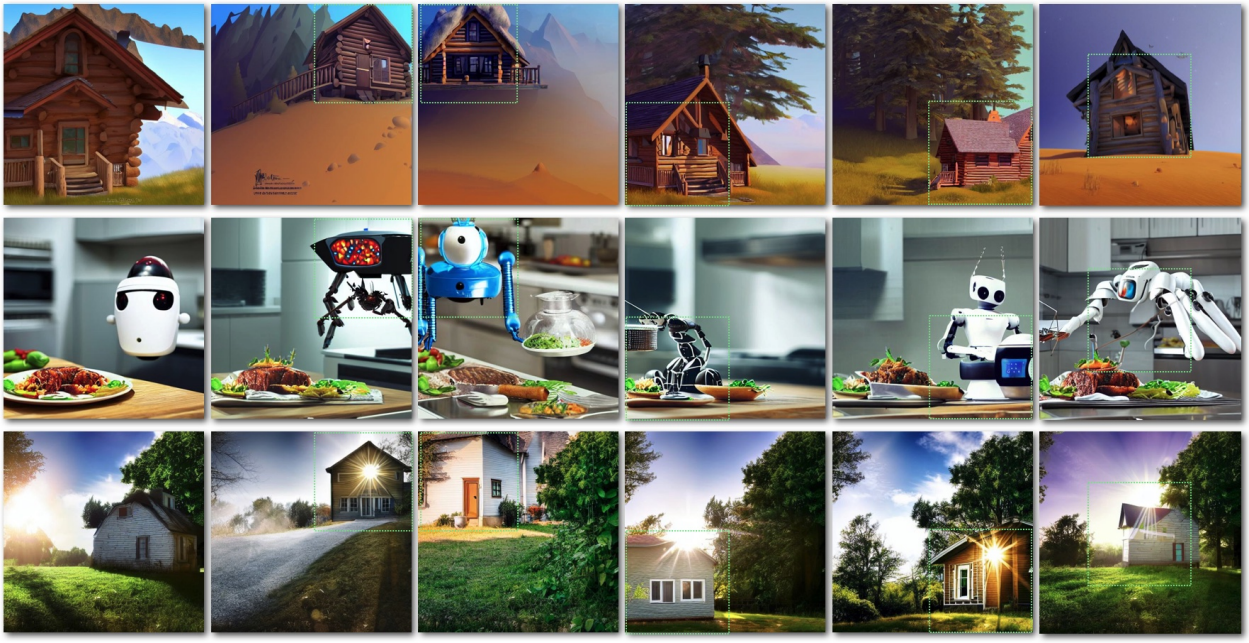


Figure 10: Four quadrants: (Top) Prompt: *"A large cabin on top of a sunny mountain in the style of Dreamworks artstation"*, with the directed object "a large cabin". (Middle) Prompt: *"An insect robot preparing a delicious meal and food in the kitchen"*, with the directed object "an insect robot". (Bottom) Prompt: *"The sun shines on a house"*, with the directed object "the sun". Images from the left are SD, DD(1st Q), DD(2nd Q), DD(3rd Q), and DD(4th Q), respectively, where Q denotes the directed quadrant.

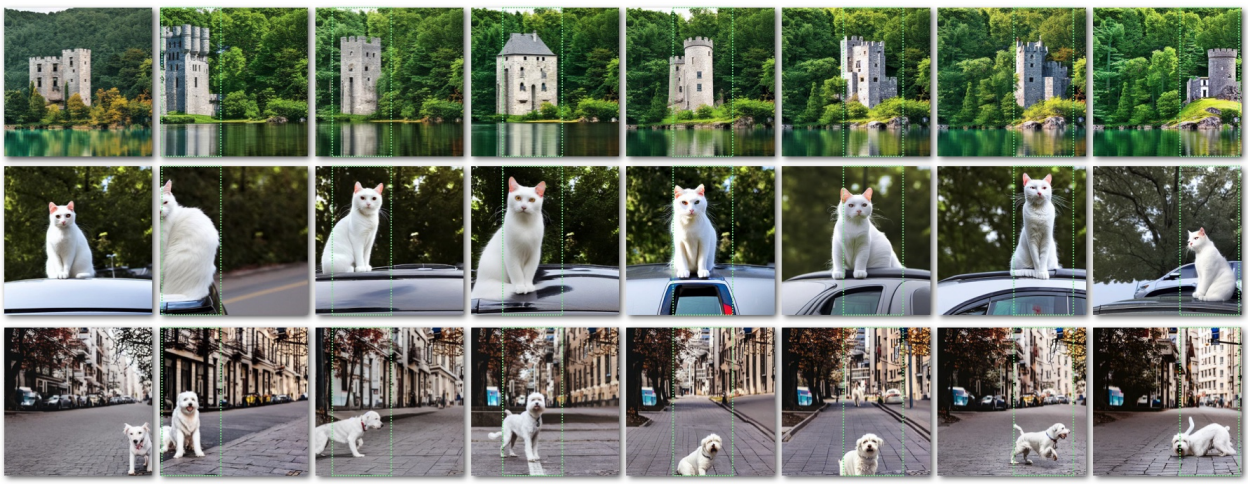


Figure 11: Sliding window: (Top) Prompt: *"A stone castle surrounded by lakes and trees"*, with the directed object "A stone castle". (Middle) Prompt: *"A white cat sitting on a car"*, with the directed object "A white cat". (Bottom) Prompt: *"A white dog waking on the street in the city"*, with the directed object "A white dog". From the left: SD, $\text{DD}_h(0\%, 40\%)$, $\text{DD}_h(10\%, 50\%)$, $\text{DD}_h(20\%, 60\%)$, $\text{DD}_h(30\%, 70\%)$, $\text{DD}_h(40\%, 80\%)$, $\text{DD}_h(50\%, 90\%)$, $\text{DD}_h(60\%, 100\%)$, where $\text{DD}_h(\cdot, \cdot)$ denotes the denotes the left and right edge of the directed region, expressed as percentages of the image resolution.

10 Placement finetuning

Placement finetuning offers a way to refine the DD (or, conventional stable diffusion, or any diffusion-based method) result without changing the subject content while keeping the environment context consistency. Fig. 12 below shows a diagram of the whole process to supplement our main paper.

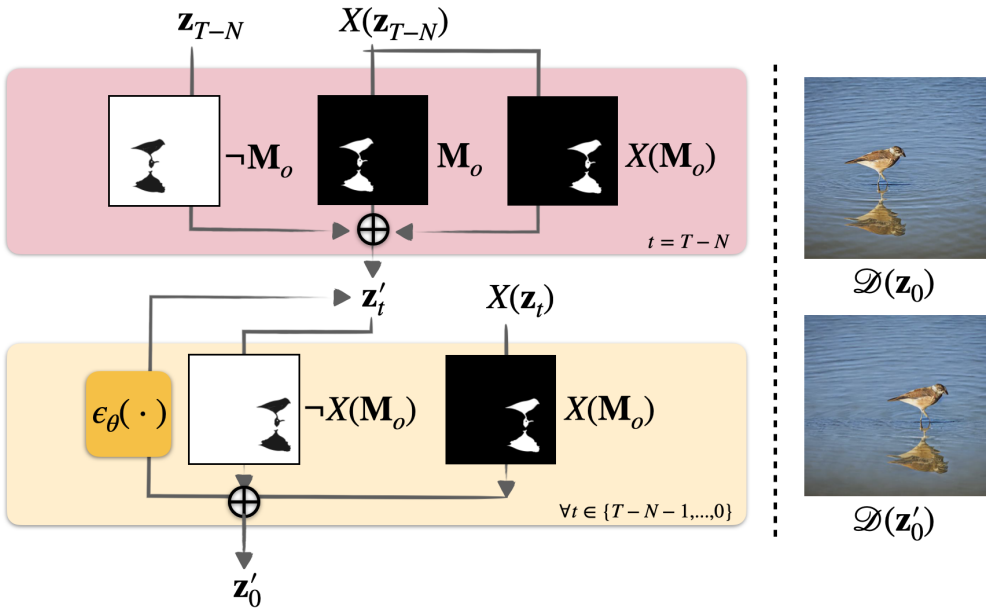


Figure 12: Placement finetuning. The final (t=0) cross-attention map recorded from a previous DD synthesis is segmented into foreground and background. At a chosen editing time step $T - N$ the foreground is moved and the background is inpainted. This edited latent is then further denoised to produce a coherent image. See the text for details.

We first define three masks by using the cross attention map of the directed object in the prompt (e.g., bird), namely, the background mask $\neg\mathbf{M}_o$, the foreground mask before moving the object $\mathbf{M}_o$, and the foreground mask after moving $X(\mathbf{M}_o)$, where $X(\cdot)$ is the spatial transformation function (restricted to translation in our current implementation). $\neg\mathbf{M}_o$ is used to inpaint the area after the object is moved, and $\mathbf{M}_o$, $X(\mathbf{M}_o)$ extract the shifted latent $X(\mathbf{z}_{T-N})$ to get the latent of the transformed object. This process is only executed once at denoising time step $T-N$. Finally, it modifies the latent to keep the repositioned directed object as constant as possible while producing a seamless blend with the background by denoising the composited result into $\mathbf{z}'_t$.

11 Scene compositing

Fig. 13 provides an additional diagram of scene compositing to supplement the description in the main paper.

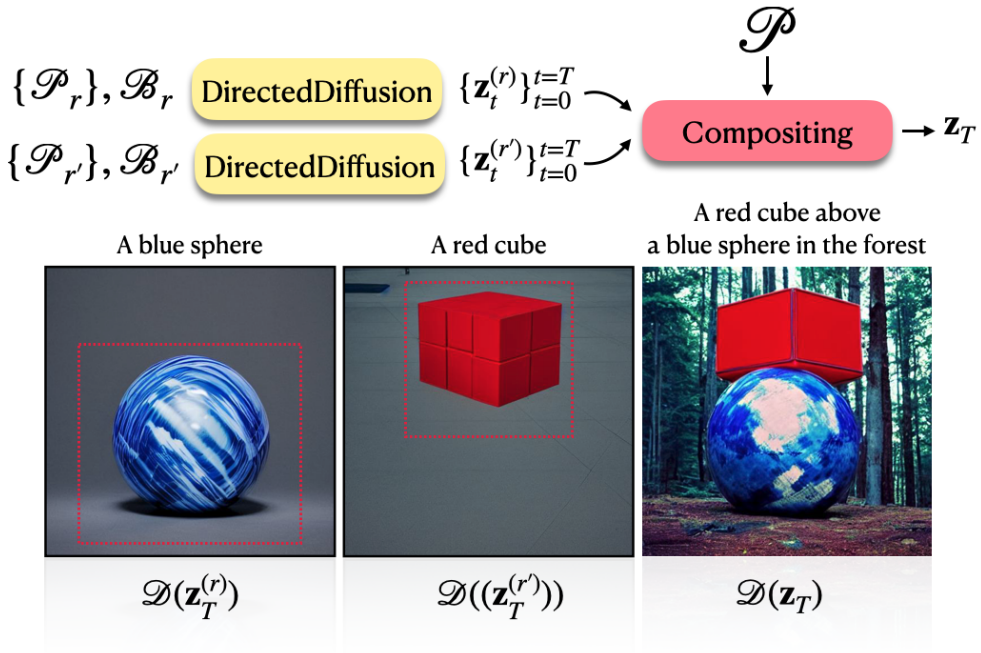


Figure 13: Scene compositing. The synthesized reconstruction $\mathcal{D}(\mathbf{z}_T)$ takes multiple latent results from DD (e.g., $\mathbf{z}_T^{(r)}$ and $\mathbf{z}_T^{(r')}$) and the original prompt $\mathcal{P}$, whose tokens are a superset of those of the individual sub-prompts ($\mathcal{P}_r$, $\mathcal{P}_{r'}$ in this example).

To sum up, DD first generates directed objects separately (e.g., a blue sphere, a red cube). For illustration purposes, the first two figures on the left depict the reconstructed sphere and cube. Subsequently, we engage in a compositing step to combine the latents of each directed object $\mathbf{z}^{(r)}$, taking their respective bounding boxes into account. After completing $N{=}10$ compositing steps (see main paper) the resulting $\mathbf{z}_t$ is then denoised with conventional SD.

Fig. 14 shows two experiments each with two directed objects: Fig. 14 (top) directs the objects *"red cube", "blue sphere"*, while Fig. 14 (bottom) shows *"bear", "bird"*, respectively. This is the well known "compositionality" failure case for SD,[4], however, DD is able to handle this case. Additionally, unlike Liu et al. [2022], which addresses this problem using a specifically trained model in their polygons example, we are able to use the generic pre-trained SD model without further modification. This results in our backgrounds having a variety of natural textures correctly aligned with the prompt contextual information (for example, the shadow of the blue sphere). In addition, the correlation of the directed objects is also retained, such as the *"watching"* prompt word associated with the bear and the bird.

[4]This failure case is highlighted on the huggingface Huggingface [2023] stable diffusion website `https://huggingface.co/CompVis/stable-diffusion-v-1-1-original`

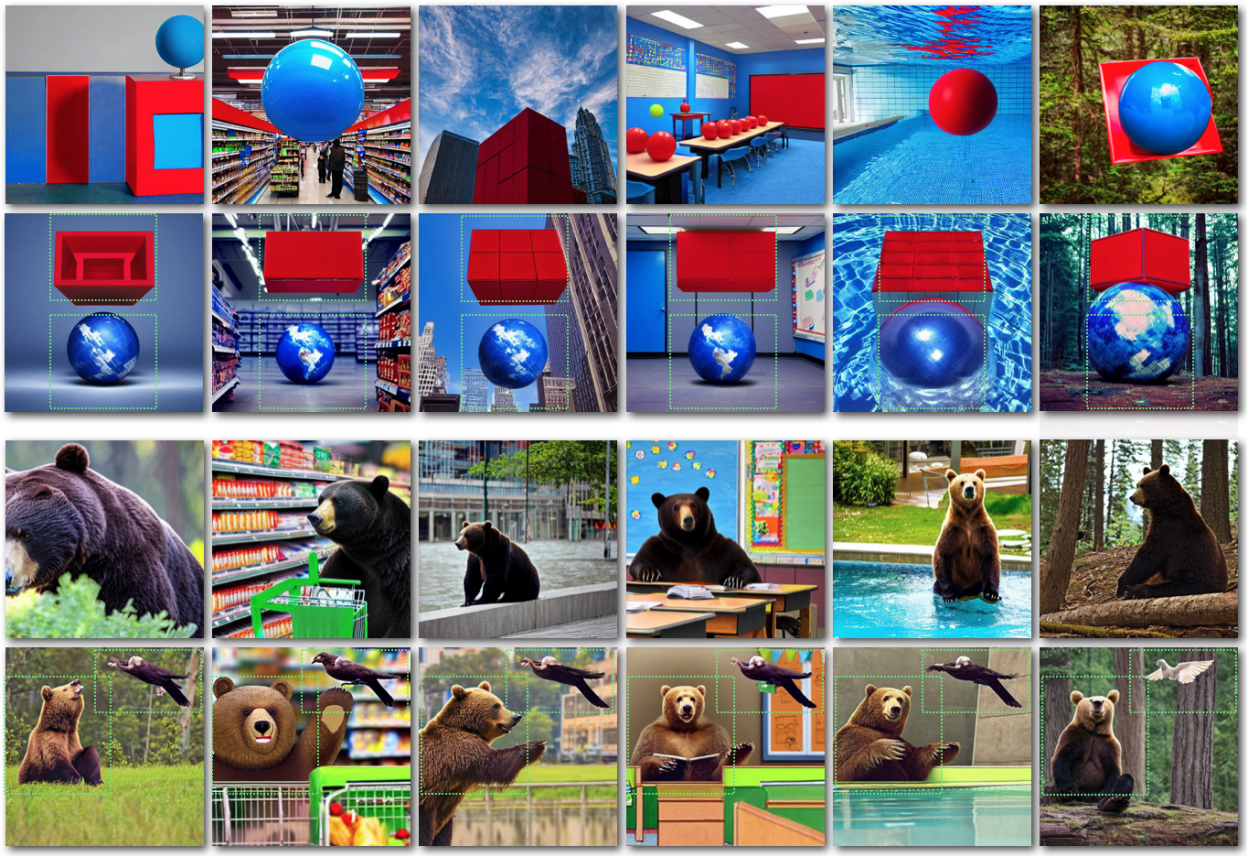


Figure 14: Two directed objects. The second and fourth rows show DD directing two objects with the green bounding boxes. For comparison, the first and third rows show the SD results on the same prompt. The first column is the original prompt *"A red cube above the blue sphere"* and *"A bear watching a bird"*, respectively, followed by columns appending various contexts at the end of the prompt: *"in the supermarket"*, *"in the city"*, *"in the classroom"*, *"in the swimming pool"*, and *"in the forest"*.

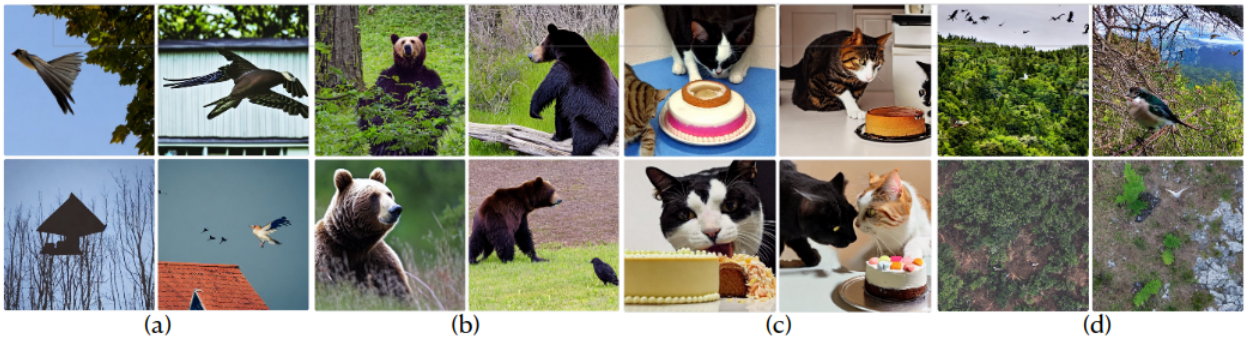


Figure 15: Stable Diffusion Huggingface [2023] failure cases. (a) *A bird flying over a house*: no house, flying house. (b) *A bear watching a bird*: no bird, bear is not watching bird. (c) *A cat watching a dog eat cake*: cat is eating, no dog. (d) *A nature photo with a bird in the upper left*: "upper left" is ignored.

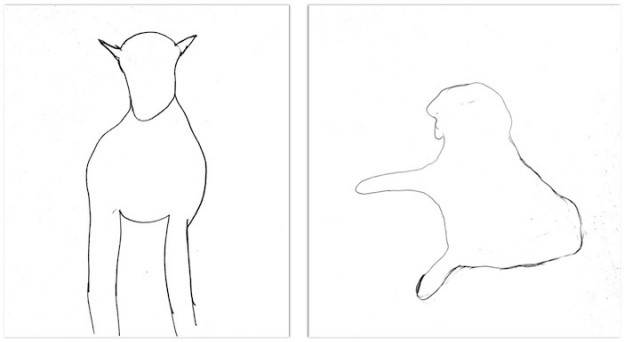


Figure 16: Users who are not artists have difficulty producing realistic detailed masks for placement guidance. Left: attempted mask for *"A camel walking toward you"*. Right: *"A dog jumping with a ball"*.

12 Qualitative Evaluations

This section provides a more extensive comparison of our method DD with the counterparts BoxDiff Xie et al. [2023] and GLIGEN Li et al. [2023], together with a brief comparison to Multidiffusion Bar-Tal et al. [2023]. [5] Although these methods were concurrently developed and we only recently became aware of them, to our knowledge they are the methods that satisfy the criteria laid out in the related work section, i.e., excluding methods that require training for layout guidance (e.g. Rombach et al. [2022]), fine tuning, or the use of detailed masks (Fig. 16).

An issue in showing qualitative comparisons is the fact that T2I methods often simply fail to synthesize the objects requested by the prompt, with successful results only being obtained after experimentation. Fig. 15 provides some examples of failures in the case of prompts that involve multiple objects. Even with single objects, however, SD often fails, for example, objects may have incorrect colors Feng et al. [2022] or animals may have the wrong number of limbs, or appear as a head with no body (or vice-versa). Because of these frequent failures, simply showing a randomly selected result has the risk of showing only a failure case, whereas cherry picking a successful result is biased. To address this we adopted the SS@k protocol in which k=12 images are generated following from an initial random seed, and the best-of-k image is subjectively selected (cherry picked) for each method. In this supplement however we show comparisons with **all 12** images and no subjective selection.

Please browse Fig. 17–Fig. 27 to examine our results relative to three concurrently developed competing methods. The results of all four methods are broadly comparable. However our method sometimes performs better on specific issues such as color bleeding due to attribute binding failure (Fig. 17), missing objects (Fig. 23), incorrect objects (Fig. 25, Fig. 27), (im)plausible scene composition and layout (Fig. 28), object color (Fig. 24), and object-object interaction (Fig. 24).

Note that one of the most important factors in our paper is the interaction among the directed objects, other subjects that are not part of the algorithm, and the environment. Following the experiment section in the paper, we divide the experiments into *Scene Compositing, One Object,* and *Two Objects* tasks.

In the **Scene Compositing** task (Figs. 17–20), the prompt comprises an array of descriptive terms aimed at describing a holistic scene. For instance, consider the prompt *"A white church under lightning in a pink sky."* In this case, the attributes "white" and "pink" should be distinctly associated with the "church" and the "sky", respectively, to the greatest extent possible.

The **One Object** task (Figs. 21–23) explores how interactions manifest between a single directed object and other subjects not directed by the algorithm. For instance, when the directed object is "bonfire" and the prompt is "A bonfire next to a man", the synthesized image should not only depict the bonfire located at the specified region. but it should also portray the man positioned next to the bonfire, and with realistic interaction and lighting between the bonfire and the man. The One Object task is extended to our final category, **Two Objects** (Figs. 24–27), which showcases control of *two* directed objects as well as an optionally specified background.

In each experiment starting with Fig. 17, we show the results of multiple random seeds for a single prompt, allowing easy visual comparison of the quality and failures of each method. Bold text in the prompt indicates the directed object(s). Please enlarge the .pdf view to see details. The associated parameters and the commands in GLIGEN and BoxDiff are displayed below each experiment.

[5] We reproduce the result with the authors' public repositories GLIGEN:`https://github.com/gligen/GLIGEN` @commit:f0ede1e; BoxDiff: `https://github.com/showlab/BoxDiff` @commit:3dbedc5; Multidiffusion: `https://huggingface.co/spaces/weizmannscience/MultiDiffusion`

13 Quantitative Evaluation and Limitations

In this section we discuss common quantitative metrics for generative models, and point out several reasons why these metrics are not as appropriate for our task as might be initially assumed.

CLIP score. This metric evaluates whether the generated image successfully reflects the content of the prompt that was used. We report a quantitative evaluation using CLIP score Radford et al. [2021] in the Experiments section of the main text. Although our results on this metric are favorable, we believe it is not the best measure for our task. Stable diffusion has considerable difficulty synthesizing scenes guided by a prompt containing several objects. This includes *attribute binding* failures, in which a color or attribute that should be applied to one object is applied to others. One hypothesis is that this difficulty with SD is due to CLIP itself. Indeed, it appears that CLIP does not have a true grammatical understanding of the prompt Feng et al. [2022], Tang et al. [2022] but rather behaves more as unordered sets of objects and attributes. In this respect, while our method escapes these limitations of CLIP and allows more reliable synthesis of multiple objects, the CLIP score may be blind to the improvement that has been achieved.

IOU. IOU is very relevant for our position-directed generation task. In our results it can be seen that while the prominent portion of the synthesized object generally lies within the desired bounding box, some portions of it frequently extend outside the provided bounding box. However, one subtlety in this task is that while position directed generative algorithms such as ours synthesize an object guided by a position indication such as a bounding box, the position of the synthesized object is not explicitly given by the network itself. One way of dealing with this is use a separate object segmentation method to attempt to locate the positioned object in the generated image. An issue here is that there is no recognized baseline for this task and concurrent papers have selected different algorithms for this purpose. The subjective choice of particular segmentation networks will lead to different IOU scores.

A larger issue is that achieving good IOU is somewhat in opposition to one of our goals (and a core strength of our method), specifically that of achieving strong *interactions between objects*. For example, in the case of a prompt *"a dog chasing a ball"*, a plausible interaction is that the dog is touching and partially obscured by the ball (see Fig. 24). This hinders the IOU since the estimated segmentation of the generated dog may not include the obscured region – but it produces more natural interaction. To quantitatively illustrate this issue, in Fig. 28 we show the IOUs of our synthesized objects together with those of two competing methods. While our IOU results are comparable to one of the methods, and to some other results presented in the literature, they are clearly inferior to the other competing methods However the visual results of both our method and the competing method with the lower IOU are *better* than the method with the higher IOU. In other words, *enforcing a high IOU appears to interfere with visual quality on our task.*

FID. The lack of relevance of FID for our task is direct: FID is intended to show improvements in generative models, however our innovation is not in the generative model but merely in how the objects are placed. Indeed, our method uses a pre-trained model with *no modifications to the weights*. Further, FID evaluates a distance between the distribution of representations of the new and baseline data in the representation layer of a classifier. Classifiers are designed to be *position invariant*, e.g. as implemented through pooling layers. In other words, FID is designed to be invariant to our task of changing the position of objects.

14 Societal Impact

The aim of this project is to extend the capability of text-to-image models to visual storytelling. In common with most other technologies, there is potential for misuse. In particular, generative models reflect the biases of their training data, and there is a risk that malicious parties can use text-to-image methods to generate misleading images. The authors believe that it is important that these risks be addressed. This might be done by penalizing malicious behavior through the introduction of appropriate laws, or by limiting the capabilities of generative models to serve this behavior.

14.1 Scene compositing: A white church under lightning in the pink sky



Figure 17: Prompt: A **white church** under lightning in the pink sky

Some GLIGEN and BoxDiff results fall short in accurately disambiguating the "pink" attribute associated with the sky from the church. Additionally, certain outcomes from GLIGEN lack natural realism, featuring "picture-in-picture" scenarios (sometimes with captions) in which the church subject is unnaturally constrained to the position box. The DD results show a more natural variety of church orientations while still being placed in roughly the desired position. The picture-in-picture behavior resembles our finding in Fig. 9 (bottom row), in which excessively strong direction results in a nearly square burger in the desired bounding box with the squirrel and surrounding context missing.

The command and the code snippet for BoxDiff and GLIGEN:

```
# BoxDiff
python run_sd_boxdiff.py --prompt "a white church under lightning in the pink sky" \
    --P 0.2 --L 1  --token_indices [3] --bbox [[82,131,357,383]]
# GLIGEN
dict(
    prompt = "a white church under lightning in the pink sky",
    phrases = ['a white church'],
    locations = [[0.25, 0.25, 0.75, 0.75]],
    alpha_type = [0.3, 0.0, 0.7],
)
```

14.2 Scene compositing: Mystical trees next to a dark magical pond

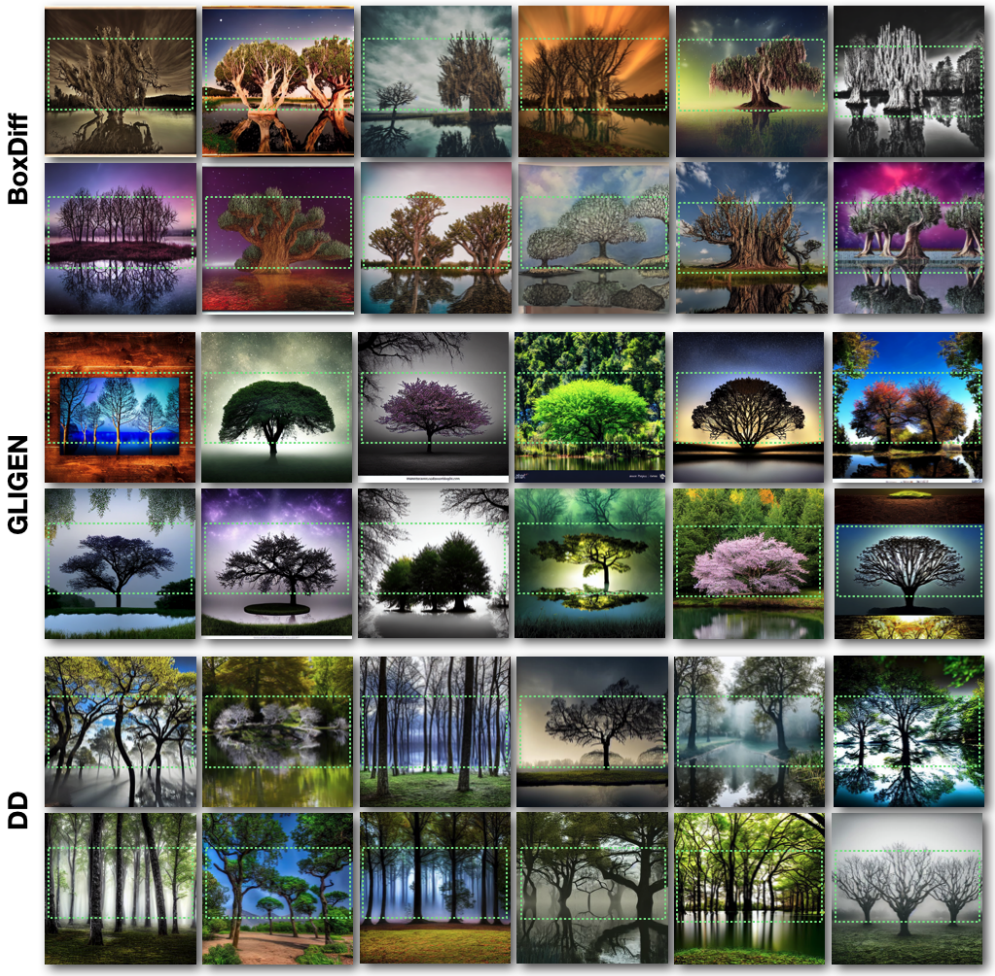


Figure 18: Prompt: **Mystical trees** next to a dark magical pond

The images produced by GLIGEN tend to position a single prominent tree within the bounding box, which contradicts the inclusion of multiple “‘trees” as specified in the prompt.

The command and the code snippet for BoxDiff and GLIGEN:

```
# BoxDiff
python run_sd_boxdiff.py --prompt "mystical trees next to a dark magical pond" \
    --P 0.2 --L 1  --token_indices [2] --bbox [[46,86,504,385]]
# GLIGEN
dict(
    prompt = "mystical trees next to a dark magical pond",
    phrases = ['mystical trees'],
    locations = [[0.1, 0.3, 0.9, 0.8]],
    alpha_type = [0.3, 0.0, 0.7],
)
```

14.3 Scene compositing: A castle in a forest with grainy fog



Figure 19: Prompt: A **castle** in a forest with grainy fog.

The command and the code snippet for BoxDiff and GLIGEN:

```
# BoxDiff
python run_sd_boxdiff.py --prompt "a castle in a forest with grainy fog" \
    --P 0.2 --L 1  --token_indices [2] --bbox [[84,109,446,436]]
# GLIGEN
dict(
    prompt = "a castle in a forest with grainy fog",
    phrases = ['a castle'],
    locations = [[0.2, 0.1, 0.7, 0.8]],
    alpha_type = [0.3, 0.0, 0.7],
)
```

14.4 Scene compositing: cherry blossoms next to the lake and a mountain

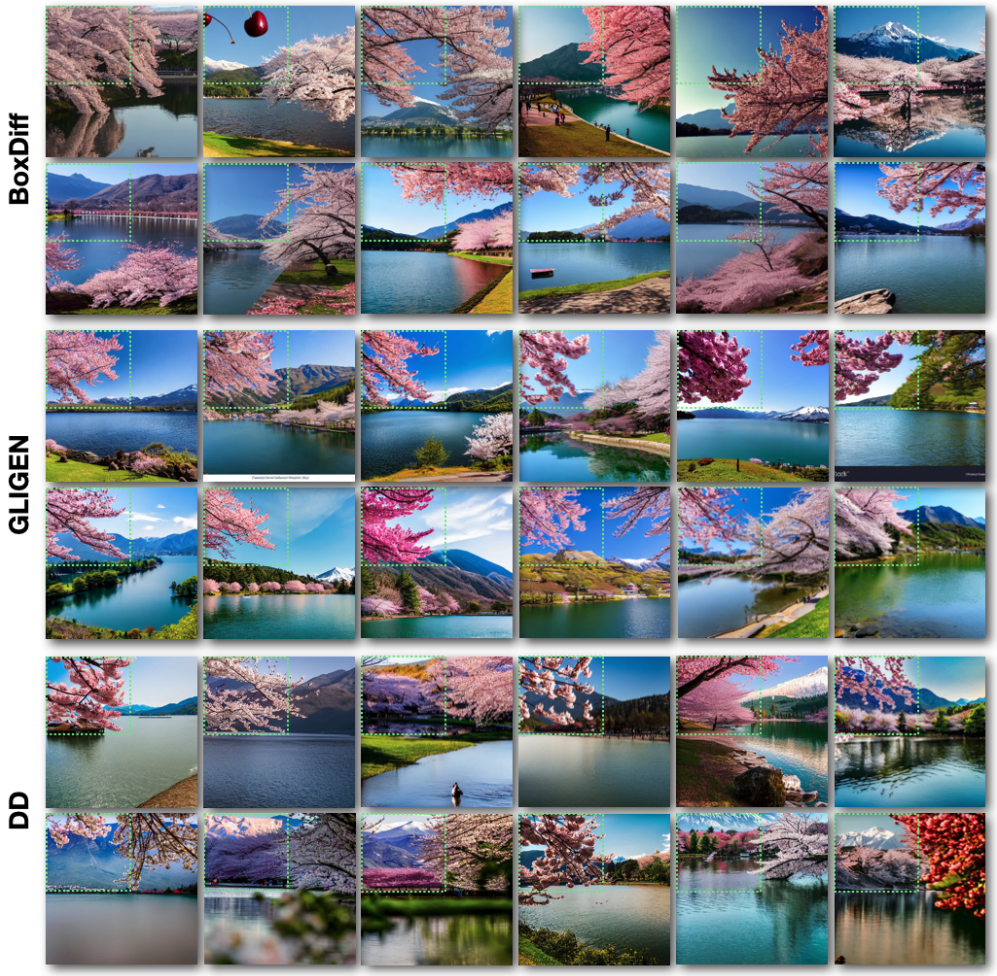


Figure 20: Prompt: The **cherry blossoms** next to the lake and a mountain

The command and the code snippet for BoxDiff and GLIGEN:

```
# BoxDiff
python run_sd_boxdiff.py --prompt "the cherry blossoms next to the lake and a mountain" \
    --P 0.2 --L 1  --token_indices [2] --bbox [[0,0,256,256]]
# GLIGEN
dict(
    prompt = "the cherry blossoms next to the lake and a mountain",
    phrases = ['the cherry blossoms'],
    locations = [[0.0, 0.0, 0.5, 0.5]],
    alpha_type = [0.3, 0.0, 0.7],
)
```

14.5 One Object: bonfire next to a man



Figure 21: Prompt: The **bonfire** next to a man

The command and the code snippet for BoxDiff and GLIGEN:

```
# BoxDiff
python run_sd_boxdiff.py --prompt "the bonfire next to a man" \
    --P 0.2 --L 1  --token_indices [2] --bbox [[0,256,256,512]]
# GLIGEN
dict(
    prompt = "the bonfire next to a man",
    phrases = ['the bonfire'],
    locations = [[0.0, 0.5, 0.5, 1.0]],
    alpha_type = [0.3, 0.0, 0.7],
)
```

14.6 One Object: A white horse in front of the erupting volcano



Figure 22: Prompt: A **white horse** in front of the erupting volcano

The command and the code snippet for BoxDiff and GLIGEN:

```
# BoxDiff
python run_sd_boxdiff.py --prompt "a white horse in front of the erupting volcano" \
    --P 0.2 --L 1  --token_indices [3] --bbox [[150,235,423,494]]
# GLIGEN
dict(
    prompt = "a white horse in front of the erupting vocano",
    phrases = ['white horse'],
    locations = [[0.25, 0.4, 0.75, 1.0]],
    alpha_type = [0.3, 0.0, 0.7],
)
```

14.7 One Object: A gravestone next to a man in red shirt



Figure 23: Prompt: A **gravestone** next to a man in red shirt

The outcomes from GLIGEN do not align with the prompt (e.g., a man in a red shirt), whereas the directed object "gravestone" consistently fits within the bounding box across all instances. Our method sometimes fails to generate the man or generates multiple men, although several seeds are successful.

The command and the code snippet for BoxDiff and GLIGEN:

```
# BoxDiff
python run_sd_boxdiff.py --prompt "a gravestone next to a man in red shirt" \
    --P 0.2 --L 1  --token_indices [2] --bbox [[48,165,461,438]]
# GLIGEN
dict(
    prompt = "a gravestone next to a man in red shirt",
    phrases = ['gravestone'],
    locations = [[0.2, 0.4, 0.7, 0.8]],
    alpha_type = [0.3, 0.0, 0.7],
)
```

14.8 Two Objects: A white dog chasing a red sphere

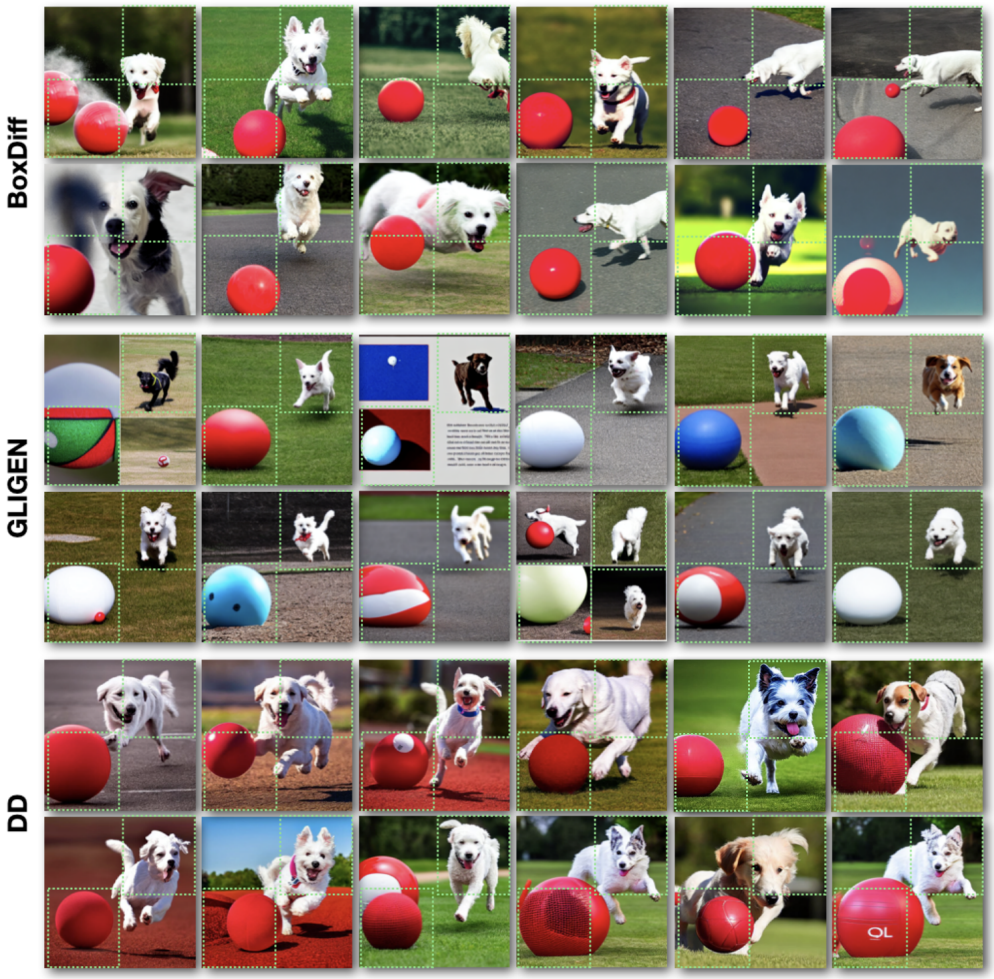


Figure 24: Prompt: A **white dog** chasing a red sphere.

A majority of the outcomes produced by GLIGEN are flawed (e.g., generating four subfigures inside the overall image or depicting the dog and sphere on a page of a book) or exhibit incorrect coloring for the ball and dog. The BoxDiff results are more successful with only a couple failures (missing background, dog running away from the sphere). The DD results are the most natural, although there is leakage of the red color to the background in a couple cases.

The command and the code snippet for BoxDiff and GLIGEN:

```
# BoxDiff
python run_sd_boxdiff.py --prompt "a white dog chasing a red ball" \
    --P 0.2 --L 1 --token_indices [3,7] --bbox [[243,6,506,263],[5,265,244,501]]
# GLIGEN
dict(
    prompt = "A white running dog chasing a red ball",
    phrases = ['white running dog', 'red ball'],
    locations = [(0.6, 0.1, 0.9, 0.5), (0.0, 0.5, 0.5, 0.9)],
    alpha_type = [0.3, 0.0, 0.7],
)
```

14.9 Two Objects: A red cube above the blue sphere in the supermarket

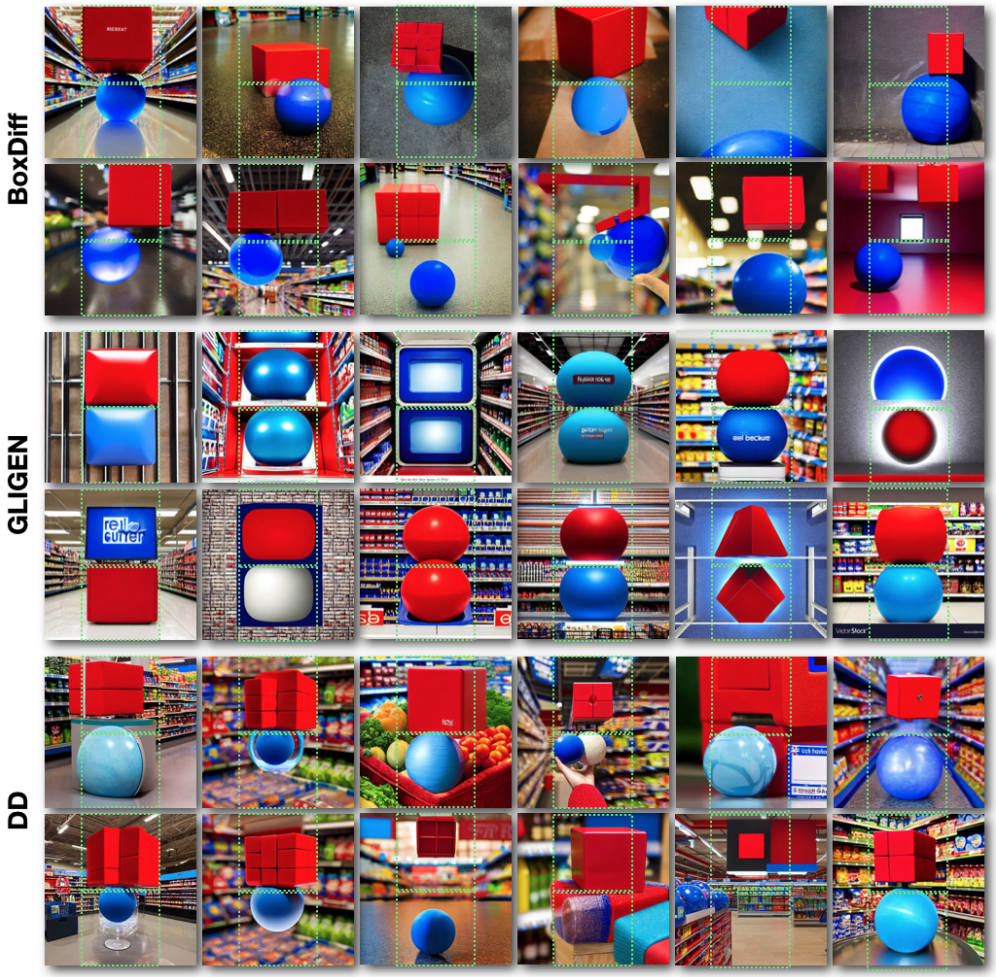


Figure 25: Prompt: A **red cube** above the **blue sphere** in the supermarket.

The outcomes obtained from GLIGEN generally do not faithfully replicate the (cube, sphere) objects requested in the prompt. The BoxDiff results correctly show a red cube above a blue sphere, but fail to show a supermarket background in some cases. Our results have the best alignment to the prompt, including a nice example where the red cube and blue sphere are placed in a basket brimming with vegetables and fruits.

The command and the code snippet for BoxDiff and GLIGEN:

```
# BoxDiff
python run_sd_boxdiff.py --prompt "a red cube above the blue sphere in the supermarket" \
    --P 0.2 --L 1 --token_indices [3,7] --bbox [[93,4,447,232],[93,244,452,504]]
# GLIGEN
dict(
    prompt = "a red cube above the blue sphere in the supermarket",
    phrases = ['a red cube', 'a blue sphere'],
    locations = [ [0.25,0.1,0.75,0.5], [0.25,0.5,0.75,0.9] ],
    alpha_type = [0.3, 0.0, 0.7],
)
```

14.10 Two Objects: MultiDiffusion



Figure 26: Prompt: A **white dog** chasing after the **red ball**

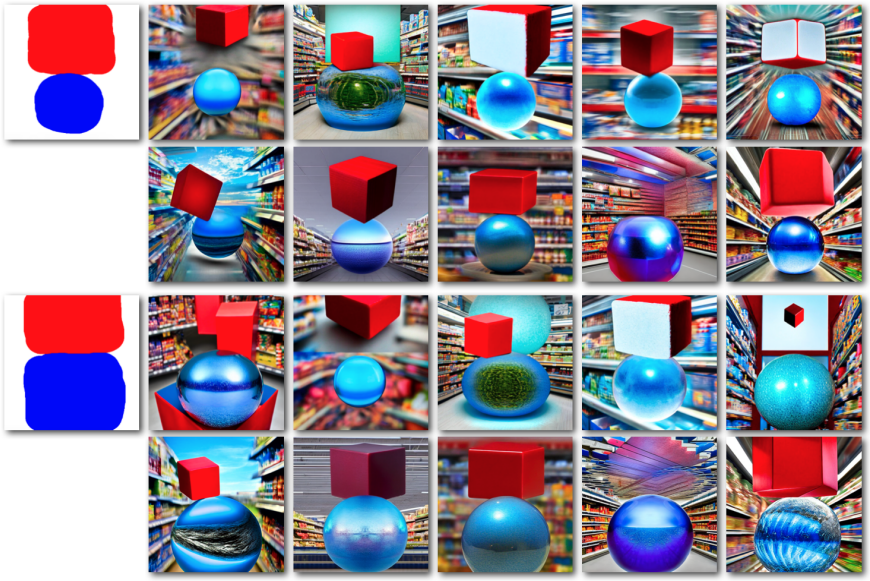


Figure 27: Prompt: A **red cube** above the **blue sphere** in the supermarket

Here we present the outcome from MultiDiffusion Bar-Tal et al. [2023]. Fig. 26 and Fig. 27 replicate the experiments previously conducted in Fig. 24 and Fig. 25. MultiDiffusion uses masks rather than boxes as its position guidance, so we provided both shaped masks and (approximate) boxes to provide a fair comparison. The method does not appear to closely follow the detailed mask, however. In their interface the color of the mask denotes a particular object but does not reflect the requested color of the object. The default demo parameters are used, with only the random seed changing. Please refer to the MultiDiffusion public Huggingface demo for more detail. [6]

[6] https://huggingface.co/spaces/weizmannscience/multidiffusion-region-based

14.11 Intersection over Union



Figure 28: Prompt: The **stone** on grass field in sunny day next to a tree

In the Fig. 28, we present Intersection over Union (IOU) results of the experiment depicted in Fig. 4. The green bounding box is the guidance (groundtruth) given to the algorithms, and the red bounding box is an estimate of the location of the synthesized object; this was produced using a simple box selection user interface by a person not associated with the project.

The averaged IOU values for BoxDiff, GLIGEN, and our DD approach are 0.51, 0.87, and **0.53**, respectively. Notably, the average IOU for GLIGEN is nearly twice as high as those for BoxDiff and DD. As seen in this example, it is important to emphasize that the IOU metric *does not correlate to the quality of the results* since our objective is not only to control the object placement but *also* generate natural images with plausible composition and interaction of the object and environment.

The command and the code snippet for BoxDiff and GLIGEN:

```
# BoxDiff
python run_sd_boxdiff.py --prompt "the stone on grass field in sunny day next to a tree" \
    --P 0.2 --L 1  --token_indices [2] --bbox [[256,256,512,512]]
# GLIGEN
dict(
    prompt = "the stone on grass field in sunny day next to a tree",
    phrases = ['stone'],
    locations = [[0.5, 0.5, 1.0, 1.0]],
    alpha_type = [0.3, 0.0, 0.7],)
```