Text2LIVE: Text-Driven Layered Image and Video Editing

Omer Bar-Tal[1(✉)], Dolev Ofri-Amar[1], Rafail Fridman[1], Yoni Kasten[2], and Tali Dekel[1]

[1] Weizmann Institute of Science, Rehovot, Israel
omer234@gmail.com
[2] NVIDIA Research, Tel Aviv, Israel

Abstract. We present a method for zero-shot, text-driven editing of natural images and videos. Given an image or a video and a text prompt, our goal is to edit the appearance of existing objects (e.g., texture) or augment the scene with visual effects (e.g., smoke, fire) in a semantic manner. We train a generator on an *internal dataset*, extracted from a single input, while leveraging an *external* pretrained CLIP model to impose our losses. Rather than directly generating the edited output, our key idea is to generate an *edit layer* (color+opacity) that is composited over the input. This allows us to control the generation and maintain high fidelity to the input via novel text-driven losses applied directly to the edit layer. Our method neither relies on a pretrained generator nor requires user-provided masks. We demonstrate localized, semantic edits on high-resolution images and videos across a variety of objects and scenes. Webpage: http://www.text2live.github.io.

Keywords: Text-guided image and video editing · CLIP

1 Introduction

Computational methods for manipulating the appearance and style of objects in natural images and videos have seen tremendous progress, facilitating a variety of editing effects to be achieved by novice users. Nevertheless, research in this area has been mostly focused in the Style-Transfer setting where the target appearance is given by a reference image (or domain of images), and the original image is edited in a global manner [14]. Controlling the localization of the edits typically involves additional input guidance such as segmentation masks. Thus, appearance transfer has been mostly restricted to global artistic stylization or to specific image domains or styles (e.g., faces, day-to-night, summer-to-winter). In this work, we seek to eliminate these requirements and enable more flexible and creative semantic appearance manipulation of real-world images and videos.

Inspired by the unprecedented power of recent Vision-Language models, we use simple text prompts to express the target edit. This allows the user to easily

O. Bar-Tal, D. Ofri-Amar and R. Fridman—Have contributed equally.

S. Avidan et al. (Eds.): ECCV 2022, LNCS 13675, pp. 707–723, 2022.
https://doi.org/10.1007/978-3-031-19784-0_41

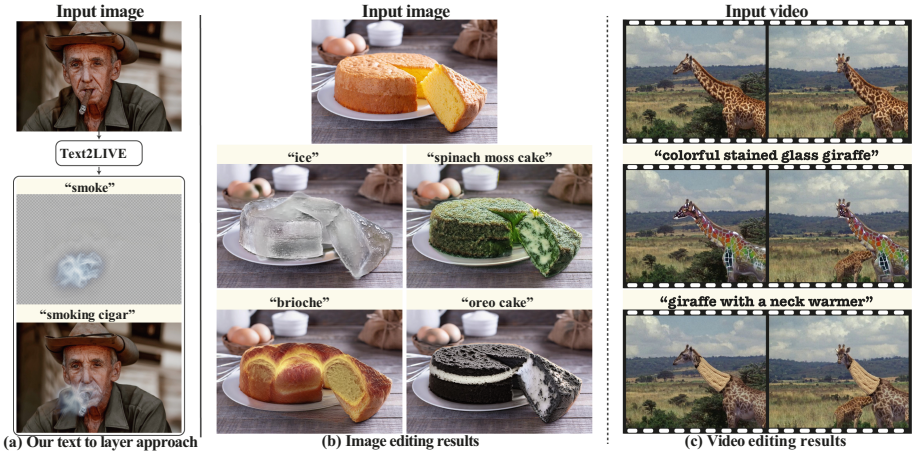

Fig. 1. Text2LIVE performs *semantic*, *localized* edits to real-world images (b), or videos (c). Our key idea is to generate an *edit layer*–RGBA image representing the target edit when composited over the input (a). This allows us to use text to guide not only the final composite, but also the edit layer itself (target text prompts are shown above each image). Our edit layers are synthesized by training a generator on a *single* input, without relying on user-provided masks or a pretrained generator.

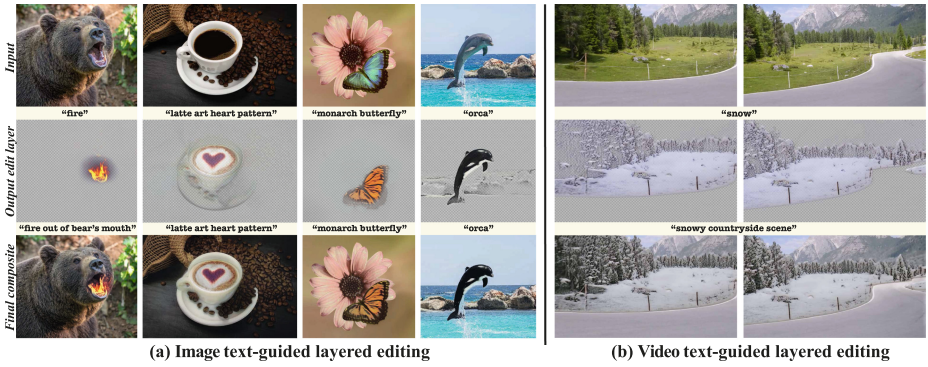

Fig. 2. Text2LIVE generates an edit layer, which is composited over the input. The text prompts expressing the target layer and the final composite are shown above each image. Our layered editing facilitates various effects including changing objects' texture or augmenting the scene with complex semi-transparent effects.

and intuitively specify the target appearance and the object/region to be edited. Specifically, our method enables *local*, *semantic* editing that satisfies a given target text prompt (e.g., Fig. 1 and Fig. 2). For example, given the cake image in Fig. 1(b), and the target text "oreo cake", our method automatically locates the cake region and synthesizes realistic, high-quality texture that combines naturally with the original image – the cream filling and the cookie crumbs "paint" the cake in a *semantically-aware* manner.

Our framework leverages the representation learned by a Contrastive Language-Image Pretraining (CLIP) model, which has been pretrained on 400 million text-image examples [32]. Various recent image editing methods [2,3,9,10,30] have demonstrated the richness of the visual and textual space spanned by CLIP. However, editing *existing* objects in *arbitrary, real-world* images remains challenging. Most existing methods combine a pre-trained generator (e.g., GAN/Diffusion model) in conjunction with CLIP. With GANs, the domain of images is restricted and requires inverting the input image to the GAN's latent space – a challenging task by itself [46]. Diffusion models [11,42] overcome these barriers but face an inherent trade-off between satisfying the target edit and maintaining high fidelity to the original content [2]. Furthermore, it is not straightforward to extend these methods to videos. In this work, we take a different route and propose to *learn a generator from a single input* – image or video and text prompts.

If no external generative prior is used, how can we steer the generation towards meaningful, high-quality edits? We achieve this via two key components: (i) we propose a novel text-guided *layered editing*, i.e., rather than directly generating the edited image, we represent the edit via an RGBA layer (color and opacity) that is composited over the input. This allows us to guide the content and localization of the generated edit via a novel objective function, including text-driven losses applied directly to the edit layer. As seen in Fig. 2, we use text prompts to express both the final edited image and the target effect (e.g., fire) represented by the edit layer. (ii) We train our generator on an *internal dataset* of diverse image-text training examples by applying various augmentations to the input image and text. We demonstrate that our internal learning approach serves as a strong regularization, enabling high quality generation of complex textures and semi-transparent effects.

We further take our framework to the realm of *text-guided video editing*. Real-world videos consist of complex objects and camera motion, providing abundant information about the scene. Yet, consistent video editing is difficult and cannot be achieved naïvely. We thus propose to decompose the video into a set of 2D *atlases* using [16]. Each atlas is a unified 2D image representing either a foreground object or the background throughout the video. This representation simplifies video editing: edits applied to a 2D atlas are consistently mapped back to the video automatically. Here, we extend our framework to perform edits in the atlas space while harnessing the rich information readily available in videos.

In summary, we present the following contributions:

- An end-to-end text-guided framework for performing localized, semantic edits of existing objects in real-world images.
- A novel layered editing approach and objective function that automatically guides the content and localization of the generated edit.
- We demonstrate the effectiveness of internal learning for training a generator on a single input in a zero-shot manner.
- An extension to video which harnesses the richness of information across time, and can perform consistent text-guided editing.

– We demonstrate various edits, ranging from changing objects' texture to generating complex semi-transparent effects, all achieved fully automatically across a wide-range of objects and scenes.

2 Related Work

Text-Guided Image Manipulation and Synthesis. There has been remarkable progress since the use of conditional GANs in both text-guided image generation [35,47–49], and editing [7,20,27]. ManiGAN [20] proposed a text-conditioned GAN for editing an object's appearance while preserving the image content. However, such multi-modal GAN-based methods are restricted to specific image domains and limited in the expressiveness of the text (e.g., trained on COCO [22]). DALL-E [33] addresses this by learning a joint image-text distribution over a massive dataset. While achieving remarkable text-to-image generation, DALL-E is not designed for editing existing images. GLIDE [28] takes this approach further, supporting both text-to-image generation and inpainting.

Instead of directly training a text-to-image model, a recent surge of methods leverage a pretrained generator, and use CLIP [32] for text-guided generation [3,10,23,30]. StyleCLIP [30] and StyleGAN-NADA [10] use a pretrained StyleGAN2 [15] for image manipulation, by either controlling the GAN's latent code [30], or by fine-tuning the StyleGAN's output domain [10]. However, editing a real input image using these methods requires first tackling the GAN-inversion challenge [36,44]. Furthermore, these methods can edit images from a few specific domains, and edit images in a *global* fashion. In contrast, we consider a different problem setting – *localized* edits that can be applied to real-world images spanning a variety of object and scene categories. A recent exploratory and artistic trend in the online AI community has demonstrated impressive text-guided image generation by steering the generation process of a pretrained VQ-GAN [8], or diffusion models [11,42] using CLIP. [17] takes this approach a step forward by optimizing the diffusion process itself. However, since the generation is *globally* controlled by the diffusion process, this method is not designed to support localized edits that are applied only to selected objects.

To enable region-based editing, user-provided masks are used to control the diffusion process for image inpainting [2]. In contrast, our goal is not to generate new objects but rather to manipulate the appearance of existing ones, while preserving the original content. Furthermore, our method is fully automatic and performs the edits directly from the text, without user edit masks.

Several works [9,12,19,26] take a *test-time optimization* approach and leverage CLIP without using a pre-trained generator. For example, CLIPDraw [9] renders a drawing that matches a target text by directly optimizing a set of vector strokes. To prevent adversarial solutions, various augmentations are applied to the output image, all of which are required to align with the target text in CLIP embedding space. CLIPStyler [19] takes a similar approach for *global* stylization. Our goal is to perform *localized* edits, which are applied only to specific objects. Furthermore, CLIPStyler optimizes a CNN that observes *only* the source image. In contrast, our generator is trained on an *internal dataset*, extracted from the

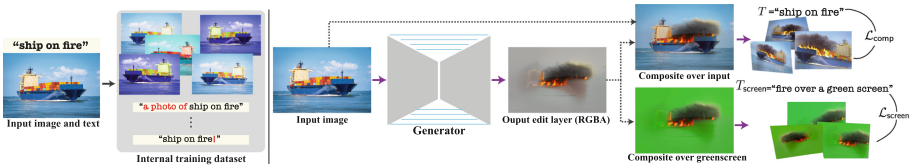

Fig. 3. *Image pipeline.* Our method consists of a generator trained on a single input image and target text prompts. *Left:* an *internal image-text dataset* of diverse training examples is created by augmenting both image and text (see Sect. 3.1). *Right:* Our generator takes as input an image and outputs an edit RGBA layer (color+opacity), which is composited over the input to form the final edited image. The generator is trained by minimizing several loss terms that are defined in CLIP space, and include: $\mathcal{L}_{\text{comp}}$, applied to the composite, and $\mathcal{L}_{\text{screen}}$, applied to the edit layer (when composited over a green background). We apply additional augmentations before CLIP (Sect. 3.1) (Color figure online)

input image and text. We draw inspiration from previous works that show the effectiveness of internal learning in the context of generation [38,40,45].

Other works use CLIP to synthesize [12] or edit [26] a single 3D representation (NeRF or mesh). The unified 3D representation is optimized through a differentiable renderer: CLIP loss is applied across different 2D rendered viewpoints. Inspired by this approach, we use a similar concept to edit videos. In our case, the "renderer" is a layered neural atlas representation of the video [16].

Consistent Video Editing. Existing approaches for consistent video editing can be roughly divided into (i) propagation-based, which use keyframes [13,43] or optical flow [37] to propagate edits throughout the video, and (ii) video layering-based, in which a video is decomposed into layers that are then edited [16,21,24,25,34]. Lu et al. [24,25] estimate *omnimattes* – RGBA layers depicting a target object and its associated scene effects. Omnimattes facilitate various video effects (e.g., object removal). However, the layers are computed independently for each frame, hence cannot support consistent edit propagation across time. Kasten et al. [16] address this challenge by decomposing a video into unified 2D atlas layers (foreground/background). Edits applied to the 2D atlases are automatically mapped back to the video, achieving temporal consistency with minimal effort. Here, we treat the neural layered atlas model as a *video renderer*, leveraging it for text-guided video editing.

3 Text-Guided Layered Image and Video Editing

We focus on semantic, localized edits expressed by simple text prompts. Such edits include changing objects' texture or semantically augmenting the scene with complex semi-transparent effects (e.g., smoke, fire). To this end, we harness the potential of learning a generator from a *single input* image or video while leveraging CLIP, which is kept fixed and used to establish our losses [32]. Our task is ill-posed – numerous possible edits can satisfy the target text according to CLIP, some of which are noisy or undesired [9,23]. Thus, controlling edits'

localization and preserving the original content are essential for achieving high-quality editing. We tackle these challenges via the following key components:

1. *Layered editing.* Our generator outputs an RGBA layer that is composited over the input image. This allows us to control the content and spatial extent of the edit via dedicated losses applied directly to the edit layer.
2. *Explicit content preservation and localization losses.* We devise new losses using the internal spatial features in CLIP space to preserve the original content, and to guide the localization of the edits.
3. *Internal generative prior.* We construct an internal dataset of examples by applying augmentations to the input image/video and text. These augmented examples are used to train our generator, whose task is to perform text-guided editing on a larger and more diverse set of examples.

3.1 Text to Image Edit Layer

As illustrated in Fig. 3, our framework consists of a generator G_θ that takes as input a source image I_s and synthesizes an *edit layer*, $\mathcal{E} = \{C, \alpha\}$, which consists of a color image C and an opacity map α. The final edited image I_o is given by compositing the edit layer over I_s:

$$I_o = \alpha \cdot C + (1 - \alpha) \cdot I_s \tag{1}$$

Our main goal is to generate $\mathcal{E}$ such that the final composite I_o would comply with a target text prompt T. In addition, generating an RGBA layer allows us to use text to further guide the generated content and its localization. To this end, we consider a couple of auxiliary text prompts: T_{screen} which expresses the target *edit layer*, when composited over a green background, and T_{ROI} which specifies a region-of-interest in the source image, and is used to initialize the localization of the edit. For example, in the *Bear* edit in Fig. 2, $T =$ *"fire out of the bear's mouth"*, $T_{\text{screen}} =$ *"fire over a green screen"*, and $T_{\text{ROI}} =$ *"mouth"*. We next describe in detail how these are used in our objective function.

Objective Function. Our novel objective function incorporates three main loss terms, all defined in CLIP's feature space: (i) $\mathcal{L}_{\text{comp}}$, which is the driving loss and encourages I_o to conform with T, (ii) $\mathcal{L}_{\text{screen}}$, which serves as a direct supervision on the edit layer, and (iii) $\mathcal{L}_{\text{structure}}$, a structure preservation loss w.r.t. I_s. Additionally, a regularization term $\mathcal{L}_{\text{reg}}$ is used for controlling the extent of the edit by encouraging sparse alpha matte α. Formally,

$$\mathcal{L}_{\text{Text2LIVE}} = \mathcal{L}_{\text{comp}} + \lambda_g \mathcal{L}_{\text{screen}} + \lambda_s \mathcal{L}_{\text{structure}} + \lambda_r \mathcal{L}_{\text{reg}}, \tag{2}$$

where λ_g, λ_s, and λ_r control the relative weights between the terms, and are fixed throughout all our experiments (see Supplementary Materials on our website - SM).

Composition Loss. $\mathcal{L}_{\text{comp}}$ reflects our primary objective of generating an image that matches the target text prompt and is given by a combination of a *cosine distance* loss and a *directional* loss [30]:

$$\mathcal{L}_{\text{comp}} = \mathcal{L}_{\cos}(I_o, T) + \mathcal{L}_{\text{dir}}(I_s, I_o, T_{\text{ROI}}, T), \tag{3}$$

where $\mathcal{L}_{\cos} = \mathcal{D}_{\cos}(E_{\text{im}}(I_o), E_{\text{txt}}(T))$ is the cosine distance between the CLIP embeddings for I_o and T. Here, E_{im}, E_{txt} denote CLIP's image and text encoders, respectively. The second term controls the direction of edit in CLIP space [10,30] and is given by: $\mathcal{L}_{\text{dir}} = \mathcal{D}_{\cos}(E_{\text{im}}(I_o) - E_{\text{im}}(I_s), E_{\text{txt}}(T) - E_{\text{txt}}(T_{\text{ROI}}))$.

Similar to most CLIP-based editing methods, we first augment each image to get several different views and calculate the CLIP losses w.r.t. each of them separately, as in [2]. This holds for all our CLIP-based losses. See SM for details.

Screen Loss. The term $\mathcal{L}_{\text{screen}}$ serves as a direct text supervision on the edit layer $\mathcal{E}$. We draw inspiration from chroma keying [4]–a well-known technique by which a solid background (often green) is replaced by an image in a post-process. Chroma keying is extensively used in image and video post-production, and there is high prevalence of online images depicting various visual elements over a green background. We thus composite the edit layer over a green background I_{green} and encourage it to match the template $T_{\text{screen}} :=$ "{ } *over a green screen*", (Fig. 3):

$$\mathcal{L}_{\text{screen}} = \mathcal{L}_{\cos}(I_{\text{screen}}, T_{\text{screen}}) \tag{4}$$

where $I_{\text{screen}} = \alpha \cdot C + (1 - \alpha) \cdot I_{\text{green}}$.

A nice property of this loss is that it allows intuitive supervision on a desired *effect*. For example, when generating semi-transparent effects, e.g., *Bear* in Fig. 2, we can use this loss to focus on the fire regardless of the image content by using $T_{\text{screen}} =$ "fire over a green screen". Unless specified otherwise, we plug in T to our screen text template in all our experiments. Similar to the composition loss, we first apply augmentations on the images before feeding to CLIP.

Structure Loss. We want to allow substantial texture and appearance changes while preserving the objects' original spatial layout, shape, and perceived semantics. While various perceptual content losses have been proposed in the context of style transfer, most of them use features extracted from a pretrained VGG [41]. Instead, we define our loss in CLIP feature space. This allows us to impose additional constraints to the resulting internal CLIP representation of I_o. Inspired by classical and recent works [18,39,45], we adopt the *self-similarity* measure. Specifically, we feed an image into CLIP's ViT and extract its K spatial tokens from the deepest layer. The self-similarity matrix, denoted by $S(I) \in \mathbb{R}^{K \times K}$, is used as structure representation. Each matrix element $S(I)_{ij}$ is defined by:

$$S(I)_{ij} = 1 - \mathcal{D}_{\cos}\left(\mathbf{t}^i(I), \mathbf{t}^j(I)\right) \tag{5}$$

where $\mathbf{t}_i(I) \in \mathbb{R}^{768}$ is the i^{th} token of image I.

The term $\mathcal{L}_{\text{structure}}$ is defined as the Frobenius norm distance between the self-similarity matrices of I_s, and I_o:

$$\mathcal{L}_{\text{structure}} = \|S(I_s) - S(I_o)\|_F \tag{6}$$

Sparsity Regularization. To control the spatial extent of the edit, we encourage the output opacity map to be sparse. Following [24,25], we define the sparsity loss as a combination of L_1- and L_0-approximation regularization terms:

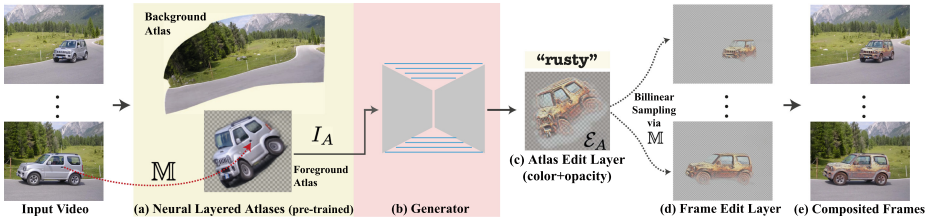

Fig. 4. *Video pipeline.* (a) a pretrained and fixed *neural layered atlas* model [16] is used as a "video renderer", which consists of: a set of 2D atlases, mapping functions from pixels to the atlases (and per-pixel fg/bg opacity values). Our framework takes in an atlas I_A and a target text prompt (e.g., "rusty car"), and trains a generator (b) to output an atlas edit layer $\mathcal{E}_A$ (c). The edited atlas (d) is rendered to frames using the pre-trained mapping network $\mathbb{M}$, and then (e) composited over the original video.

$$\mathcal{L}_{\text{reg}} = \gamma||\alpha||_1 + \Psi_0(\alpha) \tag{7}$$

where $\Psi_0(x) \equiv 2\text{Sigmoid}(5x) - 1$ is a smooth L_0 approximation that penalizes non zero elements. We fix γ in all our experiments.

Bootstrapping. To achieve accurate localized effects without user-provided edit mask, we apply a text-driven *relevancy* loss to initialize our opacity map. Specifically, we use Chefer et al. [5] to automatically estimate a *relevancy map*[1] $R(I_s) \in [0,1]^{224\times224}$ which roughly highlights the image regions that are most relevant to a given text T_{ROI}. We use $R(I_s)$ to initialize α by minimizing:

$$\mathcal{L}_{\text{init}} = \text{MSE}\left(R(I_s), \alpha\right) \tag{8}$$

Note that the relevancy maps are noisy, and only provide a rough estimation for the region of interest (Fig. 8(c)). Thus, we anneal this loss during training (see implementation details in SM). By training on diverse internal examples along with the rest of our losses, our framework dramatically refines this rough initialization, and produces accurate and clean opacity (Fig. 8(d)).

Training Data. Our generator is trained from scratch for each input (I_s, T) using an *internal dataset* of diverse image-text training examples $\{(I_s^i, T^i)\}_{i=1}^N$ that are derived from the input (Fig. 3 left). Specifically, each training example (I_s^i, T^i) is created by randomly applying a set of augmentations to I_s and T. Image augmentations include global crops, color jittering, and flip; text augmentations are sampled from predefined text templates (e.g., *"a photo of* { }*"*); see details in SM. The vast space of all augmentation combinations provides a rich and diverse dataset for training. The task is now to learn *one* mapping function G_θ for the *entire dataset*, posing a strong regularization. Specifically, for each individual example, G_θ has to generate a plausible edit layer $\mathcal{E}^i$ from I_s^i such that the composited image is well described by T^i. We demonstrate the effectiveness of our *internal learning* approach compared to test-time optimization in Sect. 4.

[1] [5] works with 224×224 images, so we resize I_s and α before applying loss (8).

3.2 Text to Video Edit Layer

A natural question is whether our image framework can be applied to videos. The key additional challenge is achieving a temporally consistent result. Naïvely applying our image framework on each frame independently yields unsatisfactory jittery results (see Sect. 4). To enforce temporal consistency, we utilize the Neural Layered Atlases (NLA) method [16], as illustrated in Fig. 4(a). We next provide a brief review of NLA and discuss in detail our extension to videos.

Preliminary: Neural Layered Atlases. NLA provides a unified 2D parameterization of a video: the video is decomposed into a set of 2D atlases, each can be treated as a 2D image, representing either one foreground object or the background throughout the entire video. An example of foreground and background atlases are shown in Fig. 4. For each video location $p = (x, y, t)$, NLA computes a corresponding 2D location (UV) in each atlas, and a foreground opacity value. This allows to reconstruct the original video from the set atlases. NLA comprises of several Multi-Layered Perceptrons (MLPs), representing the atlases, the mappings from pixels to atlases and their opacity. More specifically, each video location p is first fed into two mapping networks, $\mathbb{M}_b$ and $\mathbb{M}_f$:

$$\mathbb{M}_b(p) = (u_b^p, v_b^p), \qquad \mathbb{M}_f(p) = (u_f^p, v_f^p) \tag{9}$$

where (u_*^p, v_*^p) are the 2D coordinates in the background/foreground atlas space. Each video location is also fed to an MLP that predicts its foreground opacity. The predicted UV coordinates are then fed into an atlas network $\mathbb{A}$ that outputs RGB colors in each location. Thus, the original RGB value of p can be reconstructed by mapping p to the atlases, extracting the corresponding atlas colors, and blending them according to the predicted opacity. See [16] for full details.

Importantly, NLA enables consistent video editing: the continuous atlas (foreground/background) is first discretized to a fixed resolution image (e.g., 1000×1000 px). The user can edit the discretized atlas using image editing tools (e.g., Photoshop). The atlas edit is then mapped back to the video, and blended with the frames using the predicted UV mappings and foreground opacity. Here, we are interested in generating atlas edits in a *fully automatic* manner, solely via text.

Text to Atlas Edit Layer. Our video framework leverages NLA as a "video renderer", as illustrated in Fig. 4. Specifically, given a pretrained and fixed NLA model for a video, our goal is to generate a 2D *atlas edit layer*, either for the background or foreground, such that when mapped back to the video, each of the rendered frames would comply with the target text.

Similar to the image framework, we train a generator G_θ that takes a 2D discretized atlas I_A and generates an atlas edit layer $\mathcal{E}_A = \{C_A, \alpha_A\}$. The pretrained UV mapping $\mathbb{M}$ is used to bilinearly sample $\mathcal{E}_A$ to map it to each frame:

$$\mathcal{E}_\mathrm{t} = \mathrm{Sampler}(\mathcal{E}_A, \mathcal{S}) \tag{10}$$

$\mathcal{S} = \{\mathbb{M}(p) \mid p = (\cdot, \cdot, t)\}$ is the set of UV coordinates corresponding to frame t. The edited video is obtained by blending $\mathcal{E}_t$ with the frames, following [16].

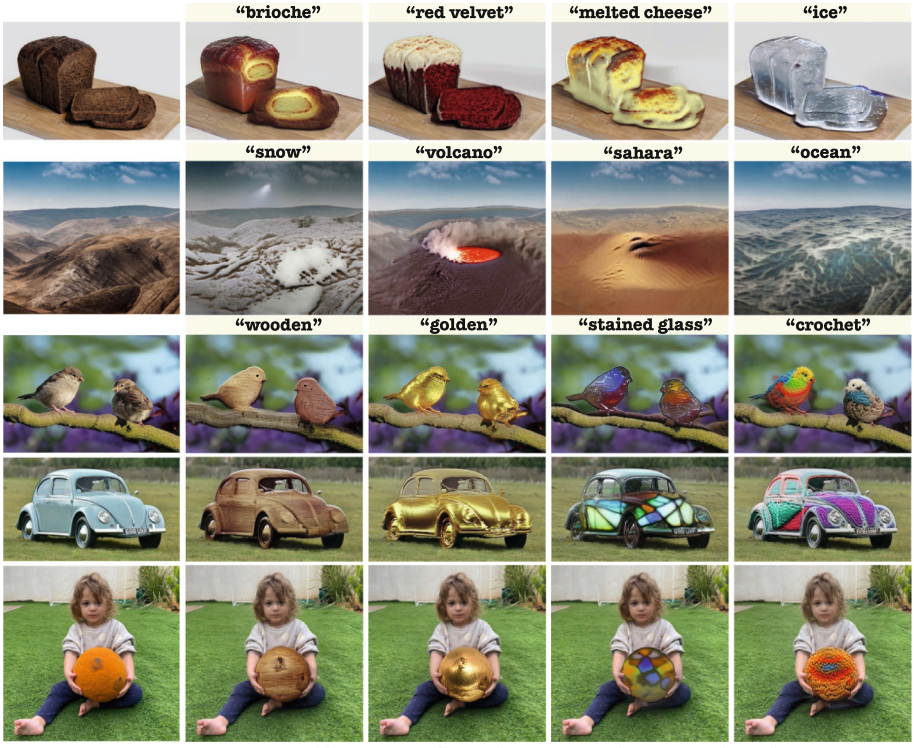

Fig. 5. *Text2LIVE image results.* Across rows: different images, across columns: different target edits. All results are produced fully automatically w/o any input masks.

Training. A straightforward approach to train G_θ is to treat I_A as an image and plug it into our image framework (Sect. 3.1). This will ensure temporal consistency, yet it has two main drawbacks: (i) the atlas often non-uniformly distorts the original structures (see Fig. 4), which may lead to low-quality edits, (ii) solely using the atlas while ignoring the video frames, disregards the abundant, diverse information available in the video such as different viewpoints or non-rigid object deformations, which can serve as "natural augmentations" to our generator. We overcome these drawbacks by mapping the atlas edit back to the video and applying our losses on the resulting edited *frames* with the same objective as in Eq. 2; we construct an internal dataset from the atlas.

More specifically, a training example is constructed by first extracting a crop from I_A. To ensure we sample informative atlas regions, we randomly crop a video segment in both space and time, and then map it to a corresponding atlas crop I_{Ac} using $\mathbb{M}$ (see SM for technical details). We then apply additional augmentations to I_{Ac} and feed it into the generator, resulting in an edit layer $\mathcal{E}_{Ac} = G_\theta(I_{Ac})$. We then map $\mathcal{E}_{Ac}$ and I_{Ac} back to the video, resulting in frame edit layer $\mathcal{E}_t$, and a reconstructed foreground/background crop I_t. This is done by bilinearly sampling $\mathcal{E}_{Ac}$ and I_{Ac} using Eq. (10), with $\mathcal{S}$ as the set of UV

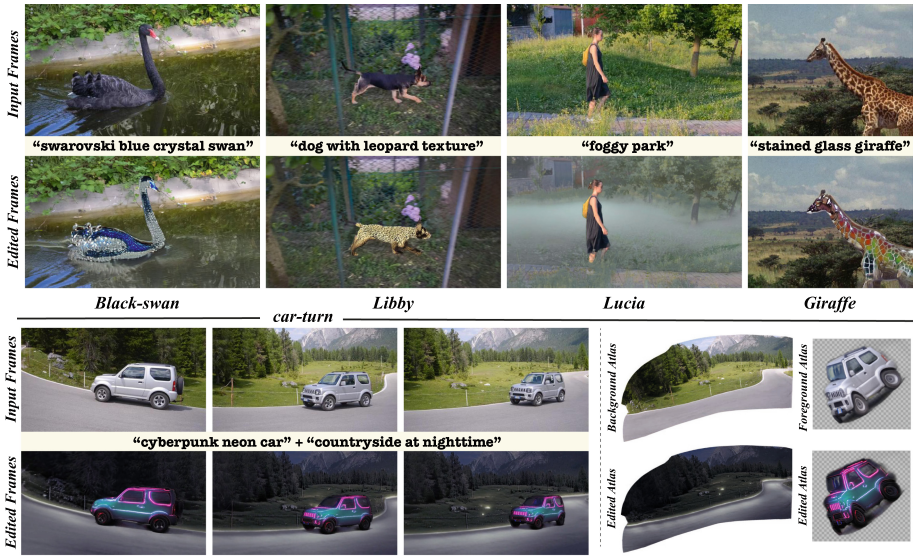

Fig. 6. *Text2LIVE video results.* A representative frame from the original and edited videos are shown for each example, along with the target text prompt. In *car-turn*, both foreground and background atlases are edited sequentially (see Sect. 4). The original and edited atlases are shown on the right. Full video results are included in the SM.

coordinates corresponding to the frame crop. Finally, we apply $\mathcal{L}_{\text{Text2LIVE}}$ from Eq. 2, where $I_s = I_t$ and $\mathcal{E} = \mathcal{E}_t$.

4 Results

We tested our method across various real-world, high-resolution images and videos. The image set contains 35 images collected from the web, spanning various object categories, including animals, food, landscapes, and more. The video set contains 7 videos from DAVIS [31]. We applied our method for various edits, ranging from text prompts that describe the texture/materials of specific objects to edits that express complex scene effects (e.g., smoke, fire, clouds). Sample results can be seen in Fig. 1, Fig. 2, and Fig. 5 for images, and Fig. 6 for videos (see SM for more). As seen, our method successfully generates photorealistic textures that are "painted" over the target objects in a semantically aware manner. Edits are accurately localized, even under partial occlusions, multiple objects (last and third row of Fig. 5) and complex scene composition (the dog in Fig. 2). Our method successfully augments the scene with complex semi-transparent effects without changing irrelevant content (see Fig. 1).

4.1 Comparison to Prior Work

To the best of our knowledge, there is no existing method tailored for our task: text-driven *semantic*, *localized* editing of *existing* objects in *real-world* images

Fig. 7. *Comparison to baselines.* Different text-guided image editing methods are applied on several images: *cake* image using "oreo cake" (Fig. 1); and *birds* using "golden birds" (Fig. 5). Manually created masks (a) are provided to the inpainting methods (b-c); the other methods are mask-free. Our results are shown in Fig. 1, and Fig. 5.

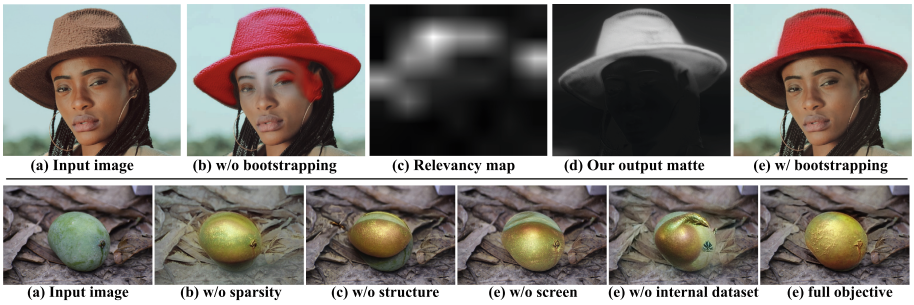

Fig. 8. *Top:* We illustrate the effect of our relevancy-based bootstrapping for (a) using "red hat" as the target edit. (b) w/o bootstrapping our edited image suffers from color bleeding. Our alpha-matte is initialized to capture the hat ($T_{\text{ROI}} =$ "hat") using the raw relevancy map, which only provides a very rough supervision (c); during training, our method dramatically refines the matting (d). *Bottom:* We ablate each of our loss terms and the effect of internal learning ("mango" to "golden mango"). See Sect. 4.3.

and videos. We illustrate the key differences between our method and several prominent text-driven image editing methods. We consider those that can be applied to a similar setting to ours: editing real-world images without restriction to specific domains. Inpainting methods: Blended-Diffusion [2] and GLIDE [28], require user-provided editing mask. CLIPStyler, which performs image stylization; Diffusion+CLIP [1], and VQ-GAN+CLIP [6]: both combine CLIP with either a pretrained VQ-GAN or a Diffusion model. See SM for qualitative comparison to StyleGAN text-guided editing methods [10,30].

Figure 7 shows representative results (more in SM). As seen, none of these methods are designed for our task. The inpainting methods (b-c), even when supplied with tight edit masks, generate new content in the masked region rather than changing the texture of the existing one. CLIPStyler (d) modifies the image in a *global* artistic manner rather than performing *local* semantic editing (e.g., the background in both examples changed entirely, regardless of the image content). Diffusion+CLIP [1] (e) can often synthesize high-quality images, but with either low fidelity to the target text or the input image content (see more in SM).

Table 1. *AMT surveys evaluation (see Sect. 4).* We report the percentage of judgments in our favor (mean, std). Our method outperforms all baselines.

Image baselines			Video baselines	
CLIPStyler	VQ-GAN+CLIP	Diffusion+CLIP	Atlas baseline	Frames baseline
0.85 ± 0.12	0.86 ± 0.14	0.82 ± 0.11	0.73 ± 0.14	0.74 ± 0.15

VQ-GAN+CLIP [6] (f) fails to maintain fidelity to the input image and produces non-realistic images. Our method automatically locates the cake region and generates high-quality texture that naturally combines with the original content.

4.2 Quantitative Evaluation

Comparison to Image Baselines. We conduct an extensive human perceptual evaluation on Amazon Mechanical Turk (AMT). We adopt the Two-alternative Forced Choice (2AFC) protocol suggested in [18,29]. Participants are shown a reference image and a target editing prompt, along with two alternatives: ours and a baseline. We consider from the above baselines those not requiring user masks. Participants were asked: *"Which image better shows objects in the reference image edited according to the text"*. We collected 12,450 user judgments using 82 image-text pairs across several prominent text-guided image editing methods. Table 1 reports the percentage of votes in our favor. Our method outperforms all baselines by a large margin, including those using a strong generative prior.

Comparison to Video Baselines. We quantify the effectiveness of our key design choices for our video framework by comparing our video method against (i) *Atlas Baseline:* feeding the discretized 2D Atlas to our single-image method (Sect. 3.1), and using the same inference pipeline illustrated in Fig. 4 to map the edited atlas back to frames. (ii) *Frames Baseline:* treating all video frames as part of a single *internal dataset*, used to train our generator; at inference, we apply the trained generator independently to each frame.

We conduct a human perceptual evaluation in which we provide participants a target editing prompt and two video alternatives: ours and a baseline. Participants were asked *"Choose the video that has better quality and better represents the text"*. We collected 2,400 user judgments over 19 video-text combinations. Table 1 reports the percentage of votes in our favor. We first note that the *Frames baseline* produces temporally inconsistent edits. The *Atlas baseline* produces temporally consistent results yet struggles to generate high-quality textures and often produces blurry results. These observations support our hypotheses mentioned in Sect. 3.2 (see SM for visual comparisons).

4.3 Ablation Study

Figure 8 (top) illustrates the effect of our relevancy-based bootstrapping (Sect. 3.1). As seen, it allows us to achieve accurate object mattes, which significantly improves the rough, inaccurate relevancy maps.

Fig. 9. *Limitations.* CLIP often exhibits biases towards certain visual elements. Our method is designed to edit existing objects, thus new objects may not be visually pleasing. Yet, the desired edit can often be achieved by using more specific text (left).

We ablate the loss terms in our objective by qualitatively comparing our results obtained with our full objective (Eq. 2) and with a specific loss removed (Fig. 8). Without $\mathcal{L}_{\text{reg}}$, the output matte does not accurately capture the mango, resulting in a global color shift around it. Without $\mathcal{L}_{\text{structure}}$, the model outputs an image with the desired appearance but fails to preserve the mango shape. Without $\mathcal{L}_{\text{screen}}$, the object segmentation is noisy (color bleeding from the mango), and the texture quality is degraded. Lastly, we consider a test-time optimization baseline by not using our internal dataset but rather feeding G_θ the same input at each training step. As seen, this baseline results in lower-quality edits.

4.4 Limitations

We noticed that for some edits, CLIP exhibits a very strong bias towards a specific solution. For example, as seen in Fig. 9, given an image of a cake, the text "birthday cake" is highly associated with candles. Our method is not designed to significantly deviate from the original layout and to create new objects, thus generates unrealistic candles. Nevertheless, in many cases the desired edit can be achieved by using more specific text. For example, the text "moon" guides the generation toward a crescent. Instead, "a bright full moon" steers the generation toward a full moon (Fig. 9 left). Finally, as noted by prior works (e.g., [26]), we also noticed that slightly different text prompts describing similar concepts may lead to slightly different flavors of edits.

Our video framework relies on the pretrained NLA model. Thus, we are restricted to examples where NLA works well, as artifacts in the atlas representation can propagate to our edited video. An exciting avenue of future research may include fine-tuning the NLA representation jointly with our model.

5 Conclusion

We considered a new problem setting in the context of zero-shot text-guided editing: semantic, localized editing of real-world images and videos. Addressing this task requires careful control of several aspects of the editing: localization, preservation of the original content, and visual quality. We proposed to generate text-driven edit layers that allow us to tackle these challenges, without using a pretrained generator in the loop. We further demonstrated how to adopt our framework, with minimal changes, for consistent text-guided video editing. We

believe that the key principles exhibited in the paper hold promise for leveraging large-scale multi-modal networks in tandem with an internal learning approach.

Acknowledgements. We thank Kfir Aberman, Lior Yariv, Shai Bagon for reviewing early drafts; Narek Tumanyan for assisting with the user evaluation. This project received funding from the Israeli Science Foundation (grant 2303/20).

References

1. Disco Diffusion. https://colab.research.google.com/github/alembics/disco-diffusion/blob/main/Disco_Diffusion.ipynb
2. Avrahami, O., Lischinski, D., Fried, O.: Blended diffusion for text-driven editing of natural images. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) (2022)
3. Bau, D., et al.: Paint by word. arXiv preprint arXiv:2103.10951 (2021)
4. Brinkmann, R.: The Art and Science of Digital Compositing: Techniques for Visual Effects, Animation and Motion Graphics. Morgan Kaufmann, Burlington (2008)
5. Chefer, H., Gur, S., Wolf, L.: Generic attention-model explainability for interpreting bi-modal and encoder-decoder transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2021)
6. Crowson, K.: VQGAN+CLIP. https://colab.research.google.com/github/justinjohn0306/VQGAN-CLIP/blob/main/VQGAN%2BCLIP(Updated).ipynb
7. Dong, H., Yu, S., Wu, C., Guo, Y.: Semantic image synthesis via adversarial learning. In: Proceedings of the IEEE International Conference on Computer Vision, ICCV (2017)
8. Esser, P., Rombach, R., Ommer, B.: Taming transformers for high-resolution image synthesis. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
9. Frans, K., Soros, L., Witkowski, O.: CLIPDraw: exploring text-to-drawing synthesis through language-image encoders. arXiv preprint arXiv:2106.14843 (2021)
10. Gal, R., Patashnik, O., Maron, H., Chechik, G., Cohen-Or, D.: StyleGAN-NADA: CLIP-guided domain adaptation of image generators. arXiv preprint arXiv:2108.00946 (2021)
11. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Advances in Neural Information Processing Systems (NeurIPS) (2020)
12. Jain, A., Mildenhall, B., Barron, J.T., Abbeel, P., Poole, B.: Zero-shot text-guided object generation with dream fields. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) (2022)
13. Jamriška, O., et al.: Stylizing video by example. ACM Trans. Graph. **38**, 1–11 (2019)
14. Jing, Y., Yang, Y., Feng, Z., Ye, J., Yu, Y., Song, M.: Neural style transfer: a review. IEEE Trans. Visual Comput. Graphics **26**(11), 3365–3385 (2019)
15. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of StyleGAN. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
16. Kasten, Y., Ofri, D., Wang, O., Dekel, T.: Layered neural atlases for consistent video editing. ACM Trans. Graph. (TOG) **40**(6), 1–12 (2021)
17. Kim, G., Ye, J.C.: DiffusionCLIP: text-guided image manipulation using diffusion models. arXiv preprint arXiv:2110.02711 (2021)

18. Kolkin, N.I., Salavon, J., Shakhnarovich, G.: Style transfer by relaxed optimal transport and self-similarity. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
19. Kwon, G., Ye, J.C.: CLIPstyler: image style transfer with a single text condition. arXiv preprint arXiv:2112.00374 (2021)
20. Li, B., Qi, X., Lukasiewicz, T., Torr, P.H.: ManiGAN: text-guided image manipulation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
21. Lin, S., Fisher, M., Dai, A., Hanrahan, P.: LayerBuilder: layer decomposition for interactive image and video color editing. arXiv preprint arXiv:1701.03754 (2017)
22. Lin, T.-Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
23. Liu, X., Gong, C., Wu, L., Zhang, S., Su, H., Liu, Q.: FuseDream: training-free text-to-image generation with improved CLIP+GAN space optimization. arXiv preprint arXiv:2112.01573 (2021)
24. Lu, E., et al.: Layered neural rendering for retiming people in video. ACM Trans. Graph. (2020)
25. Lu, E., Cole, F., Dekel, T., Zisserman, A., Freeman, W.T., Rubinstein, M.: Omnimatte: associating objects and their effects in video. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
26. Michel, O., Bar-On, R., Liu, R., Benaim, S., Hanocka, R.: Text2Mesh: text-driven neural stylization for meshes. arXiv preprint arXiv:2112.03221 (2021)
27. Nam, S., Kim, Y., Kim, S.J.: Text-adaptive generative adversarial networks: manipulating images with natural language. In: Advances in Neural Information Processing Systems (NeurIPS) (2018)
28. Nichol, A., et al.: GLIDE: towards photorealistic image generation and editing with text-guided diffusion models. arXiv preprint arXiv:2112.10741 (2021)
29. Park, T., et al.: Swapping autoencoder for deep image manipulation. In: Advances in Neural Information Processing Systems (NeurIPS) (2020)
30. Patashnik, O., Wu, Z., Shechtman, E., Cohen-Or, D., Lischinski, D.: StyleCLIP: text-driven manipulation of StyleGAN imagery. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2021)
31. Pont-Tuset, J., Perazzi, F., Caelles, S., Arbeláez, P., Sorkine-Hornung, A., Van Gool, L.: The 2017 DAVIS challenge on video object segmentation. arXiv preprint arXiv:1704.00675 (2017)
32. Radford, A., et al.: Learning transferable visual models from natural language supervision. In: Proceedings of the 38th International Conference on Machine Learning (ICML) (2021)
33. Ramesh, A., et al.: Zero-shot text-to-image generation. In: Proceedings of the 38th International Conference on Machine Learning (ICML) (2021)
34. Rav-Acha, A., Kohli, P., Rother, C., Fitzgibbon, A.W.: Unwrap mosaics: a new representation for video editing. ACM Trans. Graph. (2008)
35. Reed, S.E., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H.: Generative adversarial text to image synthesis. In: Proceedings of the 33rd International Conference on Machine Learning (ICML) (2016)
36. Richardson, E., et al.: Encoding in style: a StyleGAN encoder for image-to-image translation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021)

37. Ruder, M., Dosovitskiy, A., Brox, T.: Artistic style transfer for videos. In: Rosenhahn, B., Andres, B. (eds.) GCPR 2016. LNCS, vol. 9796, pp. 26–36. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45886-1_3
38. Shaham, T.R., Dekel, T., Michaeli, T.: SinGAN: learning a generative model from a single natural image. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2019)
39. Shechtman, E., Irani, M.: Matching local self-similarities across images and videos. In: 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) (2007)
40. Shocher, A., Bagon, S., Isola, P., Irani, M.: InGAN: capturing and retargeting the "DNA" of a natural image. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV) (2019)
41. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
42. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: 9th International Conference on Learning Representations (ICLR) (2021)
43. Texler, O., et al.: Interactive video stylization using few-shot patch-based training. ACM Trans. Graph. **39**(4), 73:1 (2020)
44. Tov, O., Alaluf, Y., Nitzan, Y., Patashnik, O., Cohen-Or, D.: Designing an encoder for StyleGAN image manipulation. ACM Trans. Graph. (TOG) **40**(4), 1–14 (2021)
45. Tumanyan, N., Bar-Tal, O., Bagon, S., Dekel, T.: Splicing ViT features for semantic appearance transfer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2022)
46. Xia, W., Zhang, Y., Yang, Y., Xue, J.H., Zhou, B., Yang, M.H.: GAN inversion: a survey. arXiv preprint arXiv:2101.05278 (2021)
47. Xu, T., et al.: AttnGAN: fine-grained text to image generation with attentional generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
48. Zhang, H., et al.: StackGAN: text to photo-realistic image synthesis with stacked generative adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2017)
49. Zhang, H., et al.: StackGAN++: realistic image synthesis with stacked generative adversarial networks. IEEE Trans. Pattern Anal. Mach. Intell. **41**(8), 1947–1962 (2019)