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 [2] NVIDIA Research

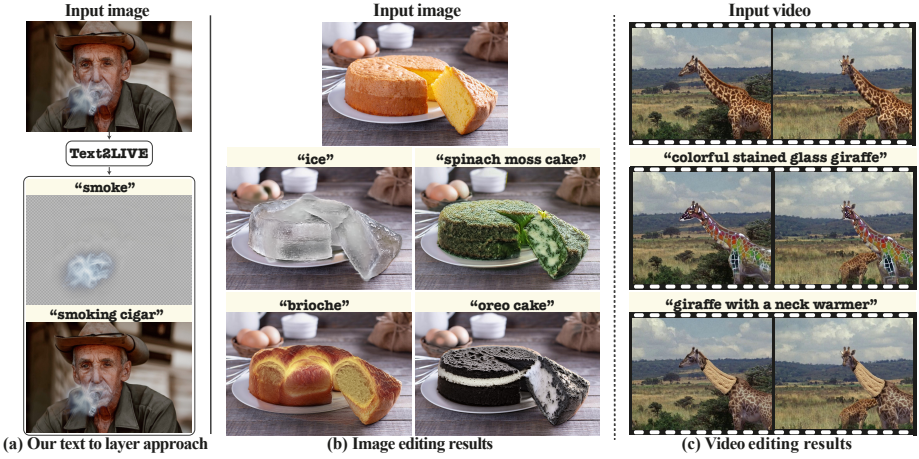

Fig. 1. Text2LIVE performs *semantic*, *localized* edits to real-world images (b), or videos (c). Our key idea is to generate an *edit layer*–RGBA image representing the target edit when composited over the original input (a). This allows us to use text to guide not only the final composite, but also 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 pre-trained generator.

Abstract. We present a method for zero-shot, text-driven appearance manipulation in natural images and videos. Given an input image or video and a target text prompt, our goal is to edit the appearance of existing objects (e.g., object's texture) or augment the scene with visual effects (e.g., smoke, fire) in a semantically meaningful manner. We train a generator using an *internal dataset* of training examples, extracted from a single input (image or video and target text prompt), while leveraging an *external* pre-trained CLIP model to establish 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 original input. This allows us to constrain the generation process and maintain high fidelity to the original input via novel text-driven losses that are applied directly to the edit layer. Our method neither relies on a pre-trained generator nor requires user-provided edit masks. We demonstrate localized, semantic edits on high-resolution natural images and videos across a variety of objects and scenes. Project page: *https://text2live.github.io/*

Keywords: text-guided image and video editing, appearance editing, CLIP

* Denotes equal contribution.

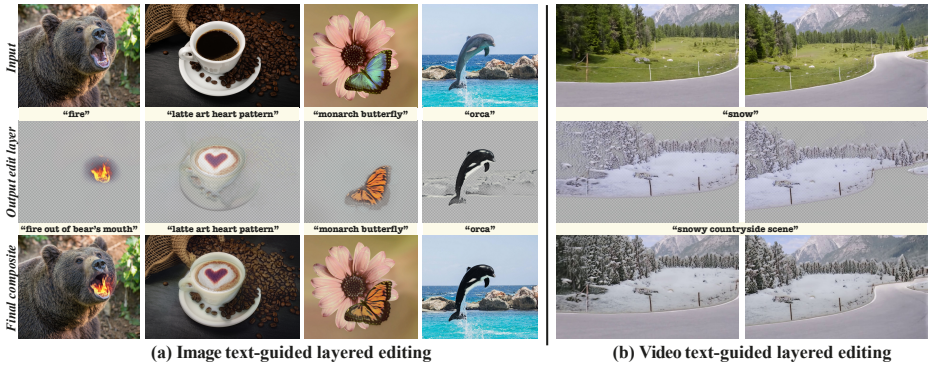

Fig. 2. Text2LIVE generates an edit layer (middle row), which is composited over the original input (bottom row). The text prompts expressing the target layer and the final composite are shown above each image. Our layered editing facilities a variety of effects including changing objects' texture or augmenting the scene with complex semi-transparent effects.

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 [16]. 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 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 full cake and the sliced piece in a *semantically-aware* manner. As seen, these properties hold across a variety of different edits.

Our framework leverages the representation learned by a Contrastive Language-Image Pretraining (CLIP) model, which has been pre-trained on 400 million text-image examples [35]. The richness of the enormous visual and textual space spanned by CLIP has been demonstrated by various recent image editing methods (e.g., [2,3,11,12,33]). However, the task of editing *existing* objects in *arbi-*

trary, real-world images remains challenging. Most existing methods combine a pre-trained generator (e.g., a GAN or a Diffusion model) in conjunction with CLIP. With GANs, the domain of images is restricted and requires to invert the input image to the GAN's latent space —a challenging task by itself [49]. Diffusion models [13,45] 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 the following 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. For example, as seen in Fig. 2, we use text prompts to express not only the final edited image but also a 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 often consist of complex object and camera motion, which provide abundant information about the scene. Nevertheless, achieving consistent video editing is difficult and cannot be accomplished naïvely. We thus propose to decompose the video into a set of 2D *atlases* using [18]. Each atlas can be treated as a unified 2D image representing either a foreground object or the background throughout the video. This representation significantly simplifies the task of video editing: edits applied to a single 2D atlas are automatically mapped back to the entire video in a consistent manner. We demonstrate how to 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 [38,50,51,52], and editing [9,22,29]. ManiGAN [22] 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 [24]). DALL-E [36] 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 [30] takes this approach further, supporting both text-to-image generation and inpainting.

Instead of directly training a text-to-image generator, a recent surge of methods leverage a pre-trained generator, and use a pre-trained CLIP [35] to guide the generation process by text [3,12,25,33]. StyleCLIP [33] and StyleGAN-NADA [12] use a pre-trained StyleGAN2 [17] for image manipulation, by either controlling the GAN's latent code [33], or by fine-tuning the StyleGAN's output domain [12]. However, editing a real input image using these methods requires first tackling the GAN-inversion challenge [39,47]. 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. CLIP is used to guide the generation process of a pre-trained generator, e.g., VQ-GAN [10], or diffusion models [13,45]. [19] 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 [11,14,21,28] take a *test-time optimization* approach and leverage CLIP without using a pre-trained generator. For example, CLIPDraw [11] 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 [21] 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 input image and text. We draw inspiration from previous works that show the effectiveness of internal learning in the context of generation [42,44,48].

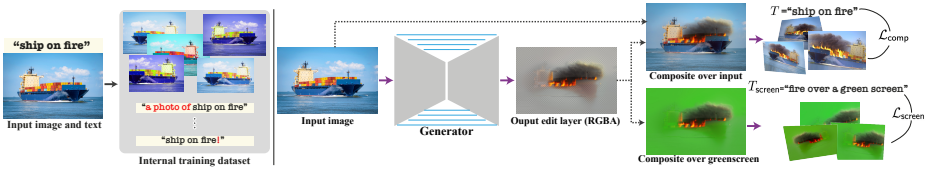

Fig. 3. *Image pipeline.* Our method consists of a generator trained on a single input image and target text prompts. *Left:* an *internal image-text dataset* of diverse training examples is created by augmenting both image and text (see Sec. 3.1). *Right:* Our generator takes as input an image and outputs 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 (Sec. 3.1)

Other works use CLIP to synthesize [14] or edit [28] 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 [18].

Consistent Video Editing. Existing approaches for consistent video editing can be roughly divided into: (i) propagation-based methods, which use keyframes [15,46] or optical flow [41] to propagate edits through the video, and (ii) video layering-based methods, in which a layered representation of the video is estimated and then edited [18,23,26,27,37]. For example, Lu et al. [26,27] estimate *omnimattes* – RGBA layers that contain a target subject along with their associated scene effects. Omnimattes facilitate a variety of video effects (e.g., object removal or retiming). However, since the layers are computed independently for each frame, it cannot support consistent propagation of edits across time. Kasten et al. [18] address this challenge by decomposing the video into unified 2D atlas layers (foreground and background). Edits applied to the 2D atlases are automatically mapped back to the video, thus achieving temporal consistency with minimal effort. In our work, we treat a pre-trained neural layered atlas model as a *video renderer* and leverage it for the task of 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 a pre-trained CLIP model, which is kept fixed and used to establish our losses [35]. Our task is ill-posed – numerous possible edits can satisfy the target text according to CLIP, some of which include noisy or undesired solutions [11,25]. Thus, controlling edits' localization and preserving the original

content are both pivotal components for achieving high-quality editing results. We tackle these challenges through 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"*, $T_{\mathsf{screen}} =$ *"fire over a green screen"*, and $T_{\mathsf{ROI}} =$ *"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) $\mathcal{L}_{\mathsf{comp}}$, which is the driving loss and encourages I_o to conform with T, (ii) $\mathcal{L}_{\mathsf{screen}}$, which serves as a direct supervision on the edit layer, and (iii) $\mathcal{L}_{\mathsf{structure}}$, a structure preservation loss w.r.t. I_s. Additionally, a regularization term $\mathcal{L}_{\mathsf{reg}}$ is used for controlling the extent of the edit by encouraging sparse alpha matte α. Formally,

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

where λ_g, λ_s, and λ_r control the relative weights between the terms, and are fixed throughout all our experiments (see Appendix A.3).

Composition loss. $\mathcal{L}_{\mathsf{comp}}$ 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 [33]:

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

where $\mathcal{L}_{\text{cos}} = \mathcal{D}_{\cos}\left(E_{\text{im}}(I_o), E_{\text{txt}}(T)\right)$ 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 [12,33] 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 Appendix A.2 for details.

Screen loss. The term $\mathcal{L}_{\text{screen}}$ serves as a direct text supervision on the generated 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 text-template $T_{\text{screen}} :=$ "{ } *over a green screen*", (Fig. 3):

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

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 pre-trained VGG model. 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 [20,43,48], we adopt the *self-similarity* measure. Specifically, we feed an image into CLIP's ViT encoder 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:

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

Sparsity regularization. To control the spatial extent of the edit, we encourage the output opacity map to be sparse. We follow [26,27] and define the sparsity

loss term as a combination of L_1- and L_0-approximation regularization terms:

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

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. [6] to automatically estimate a *relevancy map*[1] $R(I_s) \in [0,1]^{224\times 224}$ which roughly highlights the image regions that are most relevant to a given text T_{ROI}. We use the relevancy map 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 Appendix A.3). 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 generated by randomly applying a set of augmentations to I_s and to T. The image augmentations include global crops, color jittering, and flip, while text augmentations are randomly sampled from a predefined text template (e.g., *"a photo of"*+T); see Appendix A.2 for details. The vast space of all combinations between these augmentations provides us with a rich and diverse dataset for training. The task is now to learn *one* mapping function G_θ for the *entire dataset*, which poses a strong regularization on the task. 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 the test-time optimization approach in Sec. 4.

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 Sec. 4). To enforce temporal consistency, we utilize the Neural Layered Atlases (NLA) method [18], as illustrated in Fig. 4(a). We next provide a brief review of NLA and discuss in detail how 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

[1] [6] can only work with 224×224 images, so we resize both I_s and α to 224×224 before applying the loss of (8)

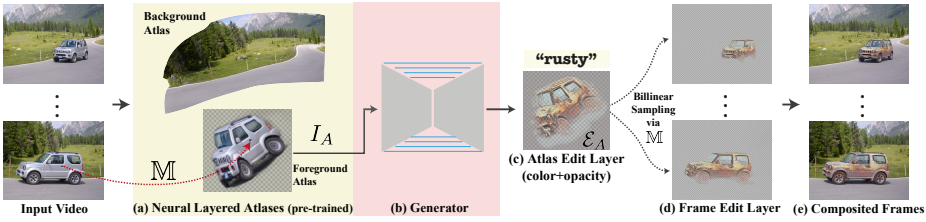

Fig. 4. *Video pipeline.* (a) a pre-trained and fixed *layered neural atlas* model [18] is used as a "video renderer", which consists of: a set of 2D atlases, mapping functions from pixels to the atlases (and per-pixel fg/bg opacity values). (b) Our framework trains a generator that takes a chosen (discretized) atlas I_A as input and a target text prompt (e.g., "rusty car"), and outputs (c) an atlas edit layer $\mathcal{E}_A$. (d) The edited atlas is rendered back to frames using the pre-trained mapping network $\mathbb{M}$, and then (e) composited over the original video.

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 pixel is also fed to an MLP that predicts the opacity value of the foreground in each position. The predicted UV coordinates are then fed into an atlas network $\mathbb{A}$, which outputs the 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. We refer the reader to [18] for full details.

Importantly, NLA enables consistent video editing: the continuous atlas (foreground or background) is first discretized to a fixed resolution image (e.g., 1000×1000 px). The user can directly 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 original frames, using the predicted UV mappings and foreground opacity. In this work, we are interested in generating atlas edits in a *fully automatic* manner, solely guided by text. **Text to Atlas Edit Layer.** Our video framework leverages NLA as a "video renderer", as illustrated in Fig. 4. Specifically, given a pre-trained 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 atlas as input and generates an atlas edit layer $\mathcal{E}_A = \{C_A, \alpha_A\}$. Note that since

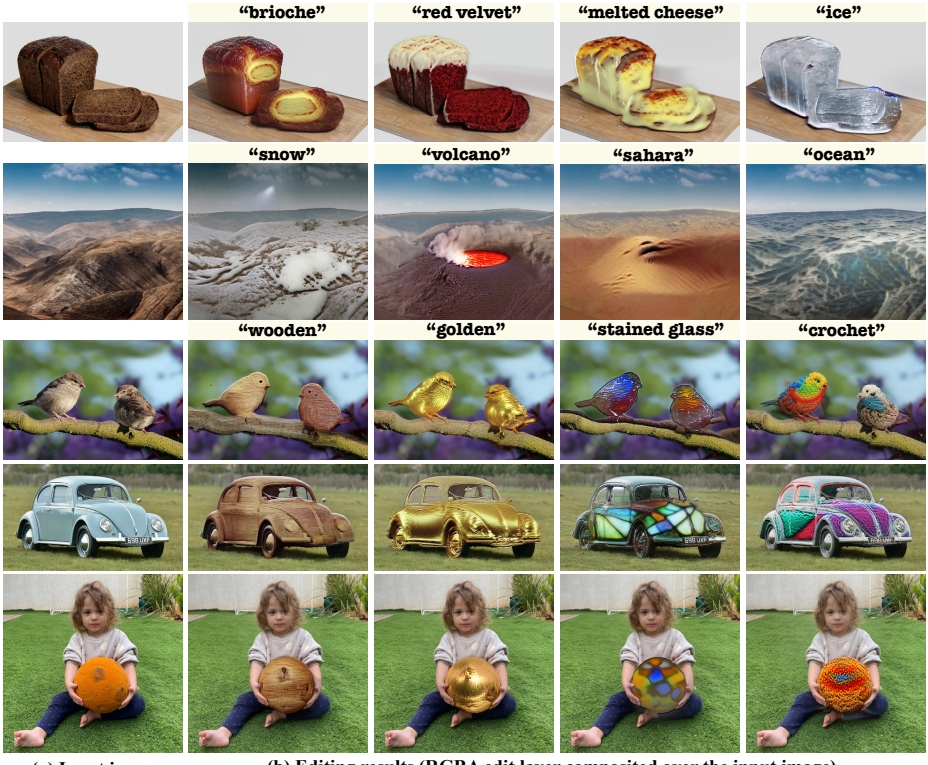

Fig. 5. *Text2LIVE image results.* Across rows: different images, across columns: different target edits. All results are produced fully automatically w/o any input masks.

G_θ is a CNN, we work with a discretized atlas, denoted as I_A. The pre-trained UV mapping, denoted by $\mathbb{M}$, is used to bilinearly sample $\mathcal{E}_A$ to map it to each frame:

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

where $\mathcal{S} = \{\mathbb{M}(p) \mid p = (\cdot, \cdot, t)\}$ is the set of UV coordinates that correspond to frame t. The final edited video is obtained by blending $\mathcal{E}_t$ with the original frames, following the same process as done in [18].

Training. A straightforward approach for training G_θ is to treat I_A as an image and plug it into our image framework (Sec. 3.1). This approach will result in a temporally consistent result, 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*. Similar to the image case, we use the same objective function (Eq. 2), and construct an internal dataset directly from the atlas for training.

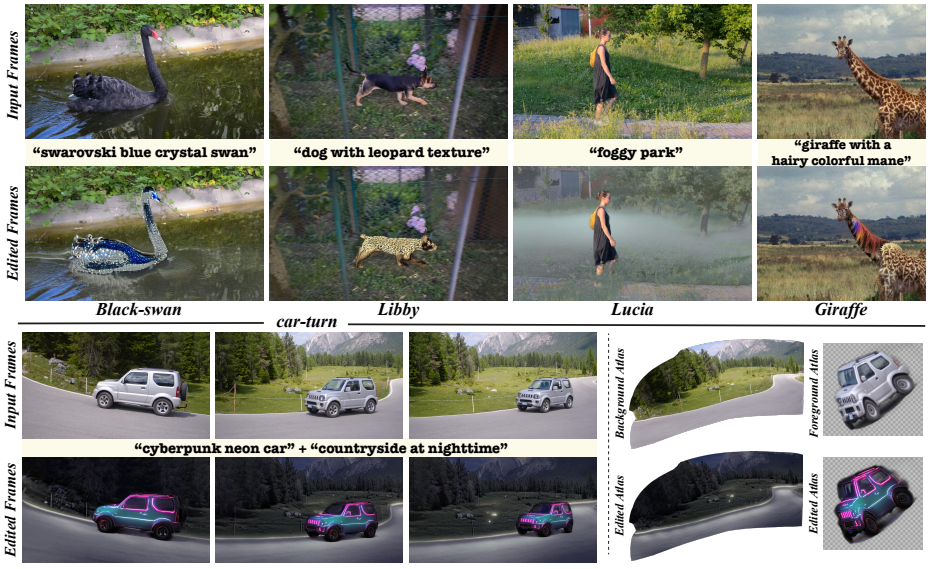

Fig. 6. *Text2LIVE video results.* A representative frame from the original and edited videos are shown for each example, along with the target text prompt. In *car-turn*, both foreground and background atlases are edited sequentially (see Sec. 4). The original and edited atlases are shown on the right.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

Fig. 7. *Comparison to baselines.* A couple of inputs are plugged into different image manipulation methods: *cake* image, shown in Fig. 1, using "oreo cake"; and *birds*, shown in Fig. 5, using "golden birds". (a) manually created masks (shown in red over the input) are provided to (b-c) the inpainting methods as additional inputs, while the rest of the methods are mask-free. Our results are shown in Fig. 1, and Fig. 5.

More specifically, a training example is constructed by first extracting a crop from I_A. To ensure we sample informative atlas regions, we first 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 Appendix A.4 for full 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 coordinates corresponding to the frame crop. Finally, we apply $\mathcal{L}_{\text{Text2LIVE}}$ from Eq. 2, where $I_s = I_t$ and $\mathcal{E} = \mathcal{E}_t$.

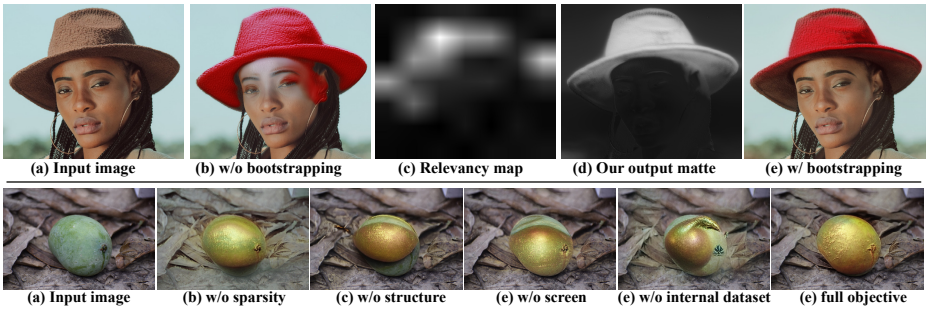

Fig. 8. *Top:* We illustrate the effect of our relevancy-based bootstrapping for image (a) using "red hat" as the target edit. (b) w/o bootstrapping our edited image suffers from color bleeding. When initializing our alpha-matte to capture the hat (T_{ROI} ="hat"), an accurate matting is achieved (d-e). Notably, the raw relevancy map provides very rough supervision (c); during training, our method dramatically refines it (d). *Bottom:* We ablate each of our loss terms and the effect of internal learning ("mango" to "golden mango"). See Sec. 4.4.

4 Results

4.1 Qualitative evaluation

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 others. The video set contains seven videos from DAVIS dataset [34]. We applied our method using various target edits, ranging from text prompts that describe the texture/materials of specific objects, to edits that express complex scene effects such as smoke, fire, or clouds. Sample examples for the inputs along with our results can be seen in Fig. 1, Fig. 2, and Fig. 5 for images, and Fig. 6 for videos. The full set of examples and results are included in the Supplementary Materials (SM). As can be seen, in all examples, our method successfully generates photorealistic textures that are "painted" over the target objects in a semantically aware manner. For example, in *red velvet* edit (first row in Fig. 5), the frosting is naturally placed on the top. In *car-turn* example (Fig. 6), the neon lights nicely follow the car's framing. In all examples, the edits are accurately localized, even under partial occlusions, multiple objects (last row and third row of Fig. 5) and complex scene composition (the dog in Fig. 2). Our method successfully augments the input scene with complex semi-transparent effects without changing irrelevant content in the image (see Fig. 1).

4.2 Comparison to Prior Work

To the best of our knowledge, there is no existing method tailored for solving our task: text-driven *semantic*, *localized* editing of *existing* objects in *real-world* images 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 that are not restricted to specific domains. Inpainting methods: Blended-Diffusion [2] and GLIDE [30], both require user-provided editing mask. CLIPStyler, which performs image stylization, and Diffusion+CLIP [1], and VQ-GAN+CLIP [7]: two baselines that combine CLIP with either a pre-trained VQ-GAN or a Diffusion model. In the SM, we also include additional qualitative comparison to the StyleGAN text-guided editing methods [33,12].

Fig. 7 shows representative results, and the rest are included in the SM. As can be 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 modifies the image in a *global* artistic manner, rather than performing *local* semantic editing (e.g., the background in both examples is entirely changed, regardless of the image content). For the baselines (d-f), Diffusion+CLIP [1] can often synthesize high-quality images, but with either low-fidelity to the target text (e), or with low-fidelity to the input image content (see many examples in SM). VQ-GAN+CLIP [7] fails to maintain fidelity to the input image and produces non-realistic images (f). Our method automatically locates the cake region and generates high-quality texture that naturally combines with the original content.

4.3 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 [20,31]. Participants are shown a reference image and a target editing prompt, along with two alternatives: our result and another baseline result. We consider from the above baselines those not requiring user-masks. The participants are asked: *"Which image better shows objects in the reference image edited according to the text"*. We perform the survey using a total of 82 image-text combinations. We collected 12,450 user judgments w.r.t. prominent text-guided image editing methods. Table 1 reports the percentage of votes in our favor. As seen, 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 the video-editing by comparing our video method against: (i) *Atlas Baseline:* feeding the discretized 2D Atlas to our single-image method (Sec. 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: our result and a baseline. The participants are asked *"Choose the video that has better quality and better*

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

Table 1. *AMT surveys evaluation (see Sec. 4).* We compare to prominent (mask-free) image baselines (left), and demonstrate the effectiveness of our design choices in the video framework compared to alternatives (right). We report the percentage of judgments in our favor (mean, std). Our method outperforms all baselines.

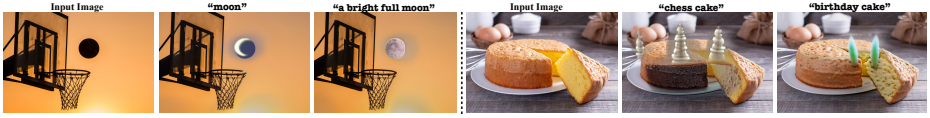

Fig. 9. *Limitations.* CLIP often exhibit strong association between text and certain visual elements such as the shape of objects (e.g., "moon" with crescent shape), or additional new objects (e.g., "birthday cake" with candles). As our method is designed to edit existing objects, generating new ones may not lead to a visually pleasing result. However, often the desired edit can be achieved by using more specific text (left).

represents the text". We collected 2,400 user judgments over 19 video-text combinations and report the percentage of votes in favor of the complete model in table 1. We first note that the *Frames baseline* produces temporally inconsistent edits. As expected, the *Atlas baseline* produces temporally consistent results. However, it struggles to generate high-quality textures and often produces blurry results. These observations support our hypotheses mentioned in Sec. 3.2. We refer the reader to the SM for visual comparisons.

4.4 Ablation Study

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

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

4.5 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 strongly associated with candles. Our method is not designed to significantly deviate from the input image layout and to create new objects, and 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 towards a crescent. By using the text "a bright full moon" we can steer the generation towards a full moon (Fig. 9 left). Finally, as acknowledged by prior works (e.g., [28]), we also noticed that slightly different text prompts describing similar concepts may lead to slightly different flavors of edits.

On the video side, our method assumes that the pre-trained NLA model accurately represents the original video. 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 existing objects within real-world images and videos. Addressing this task requires careful control of several aspects of the editing: the edit localization, the 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 pre-trained generator in the loop. We further demonstrated how to adopt our image framework, with only minimal changes, to perform 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.

6 Acknowledgments

We thank Kfir Aberman, Lior Yariv, Shai Bagon, and Narek Tumanayan for their insightful comments. We thank Narek Tumanayan for his help with the baselines comparison. 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., Andonian, A., Cui, A., Park, Y., Jahanian, A., Oliva, A., Torralba, A.: 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 (2008)
5. Caron, M., Touvron, H., Misra, I., Jegou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. ICCV (2021)
6. 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)
7. Crowson, K.: VQGAN+CLIP, https://colab.research.google.com/github/justinjohn0306/VQGAN-CLIP/blob/main/VQGAN%2BCLIP(Updated).ipynb
8. Defazio, A., Jelassi, S.: Adaptivity without compromise: a momentumized, adaptive, dual averaged gradient method for stochastic optimization. arXiv preprint arXiv:2101.11075 (2021)
9. 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)
10. Esser, P., Rombach, R., Ommer, B.: Taming transformers for high-resolution image synthesis. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
11. Frans, K., Soros, L., Witkowski, O.: Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. arXiv preprint arXiv:2106.14843 (2021)
12. 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)
13. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Advances in Neural Information Processing Systems (NeurIPS) (2020)
14. 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)
15. Jamriška, O., Šárka Sochorová, Texler, O., Lukáč, M., Fišer, J., Lu, J., Shechtman, E., Sýkora, D.: Stylizing video by example. ACM Transactions on Graphics (2019)
16. Jing, Y., Yang, Y., Feng, Z., Ye, J., Yu, Y., Song, M.: Neural style transfer: A review. IEEE transactions on visualization and computer graphics (2019)
17. 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)
18. Kasten, Y., Ofri, D., Wang, O., Dekel, T.: Layered neural atlases for consistent video editing. ACM Transactions on Graphics (TOG) (2021)
19. Kim, G., Ye, J.C.: Diffusionclip: Text-guided image manipulation using diffusion models. arXiv preprint arXiv:2110.02711 (2021)
20. 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)

21. Kwon, G., Ye, J.C.: Clipstyler: Image style transfer with a single text condition. arXiv preprint arXiv:2112.00374 (2021)
22. 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)
23. Lin, S., Fisher, M., Dai, A., Hanrahan, P.: Layerbuilder: Layer decomposition for interactive image and video color editing. arXiv preprint arXiv:1701.03754 (2017)
24. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision (ECCV) (2014)
25. 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)
26. Lu, E., Cole, F., Dekel, T., Xie, W., Zisserman, A., Salesin, D., Freeman, W.T., Rubinstein, M.: Layered neural rendering for retiming people in video. ACM Trans. Graph. (2020)
27. 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)
28. Michel, O., Bar-On, R., Liu, R., Benaim, S., Hanocka, R.: Text2mesh: Text-driven neural stylization for meshes. arXiv preprint arXiv:2112.03221 (2021)
29. 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)
30. Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., Chen, M.: Glide: Towards photorealistic image generation and editing with text-guided diffusion models. arXiv preprint arXiv:2112.10741 (2021)
31. Park, T., Zhu, J., Wang, O., Lu, J., Shechtman, E., Efros, A.A., Zhang, R.: Swapping autoencoder for deep image manipulation. In: Advances in Neural Information Processing Systems (NeurIPS) (2020)
32. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
33. 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)
34. 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)
35. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. In: Proceedings of the 38th International Conference on Machine Learning (ICML) (2021)

36. Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., Sutskever, I.: Zero-shot text-to-image generation. In: Proceedings of the 38th International Conference on Machine Learning (ICML) (2021)
37. Rav-Acha, A., Kohli, P., Rother, C., Fitzgibbon, A.W.: Unwrap mosaics: a new representation for video editing. ACM Trans. Graph. (2008)
38. 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)
39. Richardson, E., Alaluf, Y., Patashnik, O., Nitzan, Y., Azar, Y., Shapiro, S., Cohen-Or, D.: 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)
40. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. Springer (2015)
41. Ruder, M., Dosovitskiy, A., Brox, T.: Artistic style transfer for videos. In: Pattern Recognition - 38th German Conference (GCPR) (2016)
42. 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)
43. 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)
44. 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)
45. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: 9th International Conference on Learning Representations, (ICLR) (2021)
46. Texler, O., Futschik, D., Kučera, M., Jamriška, O., Šárka Sochorová, Chai, M., Tulyakov, S., Sýkora, D.: Interactive video stylization using few-shot patch-based training. ACM Transactions on Graphics (2020)
47. Tov, O., Alaluf, Y., Nitzan, Y., Patashnik, O., Cohen-Or, D.: Designing an encoder for stylegan image manipulation. ACM Transactions on Graphics (TOG) (2021)
48. Tumanyan, N., Bar-Tal, O., Bagon, S., Dekel, T.: Splicing vit features for semantic appearance transfer. arXiv preprint arXiv:2201.00424 (2022)
49. Xia, W., Zhang, Y., Yang, Y., Xue, J.H., Zhou, B., Yang, M.H.: Gan inversion: A survey. arXiv preprint arXiv:2101.05278 (2021)
50. Xu, T., Zhang, P., Huang, Q., Zhang, H., Gan, Z., Huang, X., He, X.: 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)
51. Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., Metaxas, D.N.: Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2017)
52. Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., Metaxas, D.N.: Stackgan++: Realistic image synthesis with stacked generative adversarial networks. IEEE Transactions on Pattern Analysis and Machine Intelligence (2019)

A Implementation Details

We provide implementation details for our architecture and training regime.

A.1 Generator Network Architecture

We base our generator G_θ network on the `U-Net` architecture [40], with a 7-layer encoder and a symmetrical decoder. All layers comprise 3×3 Convolutional layers, followed by `BatchNorm`, and `LeakyReLU` activation. The intermediate channels dimensions is 128. In each level of the encoder, we add an additional 1×1 Convolutional layer and concatenate the output features to the corresponding level of the decoder. Lastly, we add a 1×1 Convolutional layer followed by `Sigmoid` activation to get the final RGB output.

A.2 Internal Dataset (Sec. 3.1)

We apply data augmentations to the source image and target text (I_s, T) to create multiple *internal examples* $\{(I_s^i, T^i)\}_{i=1}^N$. Specifically, at each training step, we apply a random set of image augmentations to I_s, and augment T using a pre-defined set of text templates, as follows:

Image augmentations

- Random spatial crops: 0.85 and 0.95 of the image size in our image and video frameworks respectively.
- Random scaling: aspect ratio preserved scaling, of both spatial dimensions by a random factor, sampled uniformly from the range $[0.8, 1.2]$.
- Random horizontal-flipping is applied with probability p=0.5.
- Random color jittering: we jitter the global brightness, contrast, saturation and hue of the image.

Text augmentations and the target text prompt T We compose T with a random text template, sampled from of a pre-defined list of 14 templates. We designed our text-templates that does not change the semantics of the prompt, yet provide variability in the resulting CLIP embedding e.g.:

- "photo of {}."
- "high quality photo of {}."
- "a photo of {}."
- "the photo of {}."
- "image of {}."
- "an image of {}."
- "high quality image of {}."
- "a high quality image of {}."
- "the {}."
- "a {}."
- "{}."

– "{}"
– "{}!"
– "{}..."

At each step, one of the above templates is chosen at random and the target text prompt T is plugged in to it and forms our augmented text. By default, our framework uses a single text prompt T, but can also support multiple input text prompts describing the same edit, which effectively serve as additional text augmentations (e.g., "crochet swan", and "knitted swan" can both be used to describe the same edit).

A.3 Training Details

We implement our framework in PyTorch [32] (code will be made available). As described in Sec. 3, we leverage a pre-trained CLIP model [35] to establish our losses. We use the `ViT-B/32` pretrained model (12 layers, 32x32 patches), downloaded from the official implementation at GitHub. We optimize our full objective (Eq. 2, Sec. 3.1), with relative weights: $\lambda_g = 1$, $\lambda_s = 2$ (3 for videos), $\lambda_r = 5 \cdot 10^{-2}$, ($5 \cdot 10^{-4}$ for videos) and $\gamma = 2$. For bootstrapping, we set the relative weight to be 10, and for the image framework we anneal it linearly throughout the training. We use the MADGRAD optimizer [8] with an initial learning rate of $2.5 \cdot 10^{-3}$, weight decay of 0.01 and momentum 0.9. We decay the learning rate with an exponential learning rate scheduler with gamma = 0.99 (gamma = 0.999 for videos), limiting the learning rate to be no less than 10^{-5}. Each batch contains (I_s^i, T^i) (see Sec. 3.1), the augmented source image and target text respectively. Every 75 iterations, we add $\{I_s, T\}$ to the batch (i.e., do not apply augmentations). The output of G_θ is then resized down to 224[px] maintaining aspect ratio and augmented (e.g., geometrical augmentations) before extracting CLIP features for establishing the losses. We enable feeding to CLIP arbitrary resolution images (i.e., non-square images) by interpolating the position embeddings (to match the size of spatial tokens of a the given image) using bicubic interpolation, similarly to [5].

Training on an input image of size 512×512 takes ~ 9 minutes to train on a single GPU (NVIDIA RTX 6000) for a total of 1000 iterations. Training on one video layer (foreground/background) of 70 frames with resolution 432×768 takes $\sim$60 minutes on a single GPU (NVIDIA RTX 8000) for a total of 3000 iterations.

A.4 Video Framework

We further elaborate on the framework's details described in Sec. 3.2 of the paper.

Atlas Pre-processing. Our framework works on a discretized atlas, which we obtain by rendering the atlas to a resolution of 2000×2000 px. This is done as in [18], by querying the pre-trained atlas network in uniformly sampled UV locations. The neural atlas representation is defined within the [-1,1] continuous

space, yet the video content may not occupy the entire space. To focus only on the used atlas regions, we crop the atlas prior to training, by mapping all video locations to the atlas and taking their bounding box. Note that for foreground atlas, we map only the foreground pixels in each frame, i.e., pixels for which the foreground opacity is above 0.95; the foreground/background opacity is estimated by the pre-trained neural atlas representation.

Training. As discussed in Sec. 3.2 in the paper, our generator is trained on atlas crops, yet our losses are applied to the resulting edited frames. In each iteration, we crop the atlas by first sampling a video segment of 3 frames and mapping it to the atlas. Formally, we sample a random frame t and a random spatial crop size (W, H) where its top left coordinate is at (x, y). As a result we get a set of cropped (spatially and temporally) video locations:

$$\mathcal{V} = \{p = (x + j, y + i, t + m) \quad \text{s.t.} \quad 0 \leq j < W, 0 \leq i < H, m \in \{-k, 0, k\}\} \tag{11}$$

where $k = 2$ is the offset between frames.

The video locations set $\mathcal{V}$ is then mapped to its corresponding UV atlas locations: $\mathcal{S}_\mathcal{V} = \mathbb{M}(\mathcal{V})$, where $\mathbb{M}$ is a pre-trained mapping network. We define the atlas crop I_{Ac} as the minimal crop in the atlas space that contains all the mapped UV locations:

$$I_{Ac} = \begin{Bmatrix} I_A[u, v] & s.t. & \min(\mathcal{S}_\mathcal{V}.u) \leq u \leq \max(\mathcal{S}_\mathcal{V}.u) \\ & & \min(\mathcal{S}_\mathcal{V}.v) \leq v \leq \max(\mathcal{S}_\mathcal{V}.v), \end{Bmatrix} \tag{12}$$

We augment the atlas crop I_{Ac} as well as the target text T, as described in Sec. A.2 herein to generate an internal training dataset. To apply our losses, we map back the atlas edit layer to the original video segment and process the edited frames the same way as in the image framework: resizing, applying CLIP augmentations, and applying the final loss function of Eq. 2 in Sec. 3.1 in the paper. To enrich the data, we also include one of the sampled frame crops as a direct input to G and apply the losses directly on the output (as in the image case). Similarly to the image framework, every 75 iterations we additionally pass the pair $\{I_A, T\}$, where I_A is the entire atlas (without augmentations, and without mapping back to frames). For the background atlas, we first downscale it by three due to memory limitations.

Inference. As described in Sec. 3.2, at inference time, the entire atlas I_A is fed into G_θ results in $\mathcal{E}_A$. The edit is mapped and combined with the original frames using the process that is described in [18](Sec. 3.4, Eq. (15),(16)). Note that our generator operates on a single atlas. To produce foreground and background edits, we train two separate generators for each atlas.