Supplementary Material of
Blending-NeRF: Text-Driven Localized Editing in Neural Radiance Fields

Hyeonseop Song[1*] Seokhun Choi[1*] Hoseok Do[1] Chul Lee[1] Taehyeong Kim[2†]

[1]AI Lab, CTO Division, LG Electronics, Republic of Korea
[2]Dept. of Biosystems Engineering, Seoul National University, Republic of Korea
{hyeonseop.song, seokhun.choi, hoseok.do, clee.lee}@lge.com taehyeong.kim@snu.ac.kr

A. Implementation Details

A.1. Dilatation Operations

We use CLIPseg [3] to extract the target regions for localized editing. Specifically, CLIPSeg takes an image of size 352×352 as input, so we resize the $S \times S$ size rendered original patch $I^o(\theta, \mathbf{p})$ to 352×352 before feeding it to CLIPSeg. We then estimate the target region M using the source text and the base text *'photo'*. The dilatation operations with a kernel of size 5×5 are applied to obtain positive (M_+) and negative target (M_-) regions with N_f and $2N_f$ times as shown in Figure 1. After this step, the regions are resized to fit the rendered image size $S \times S$ and the rest of the process is performed. We use the consistent notation M_+ and M_- for readability.

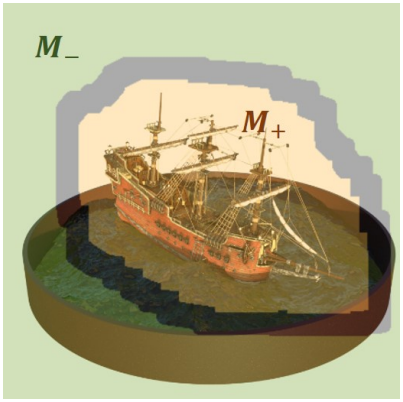

Figure 1: Positive and negative target regions ($N_f = 10$).

A.2. Hyperparameter for Region Loss

We observed that if adding densities is included in the editing operations, Blending-NeRF has difficulty making densities at initial when giving the same weights (*i.e.* $\lambda_+ = 1$) to the positive and negative regions in the loss $\mathcal{L}_{\text{region}}$ (see Eq. 12), especially for the small target region. To compensate for this, we set λ_+ by the ratio r as:

$$r = \max(30, (S^2 - \textit{area of } M_+)/(1 + \textit{area of } M_+)) \tag{1}$$

Note that this method is only used for object editing task that involves the adding density operation.

A.3. Patch Sampling for Training

Given the sampled camera pose $\mathbf{p}$, the rays are sampled at even intervals to make an image patch covering the entire extent of the image plane. Specifically, let the width and height of the image plane and the patch size be W, H, and S, respectively; the starting points along each axis are uniformly sampled by the following distributions:

$$\begin{aligned} &\mathcal{U}(0, \lfloor W/S \rfloor + (W \mod S) - 1) \\ &\mathcal{U}(0, \lfloor H/S \rfloor + (H \mod S) - 1). \end{aligned} \tag{2}$$

From these starting points, image patches are rendered at even intervals with horizontal stride $\lfloor W/S \rfloor$ and vertical stride $\lfloor H/S \rfloor$. Finally, we can obtain $S \times S$ image and opacity patches.

[*]Equal contribution.
[†]Corresponding author. Partially conducted at LG Electronics.

A.4. Training Time

We train our model with 3k iterations on tasks that only change color or remove densities. In the tasks of adding densities with color change and the tasks of adding only densities, we iteratively train our model for 4k and 5k, respectively. Our method takes about 12 minutes to train 1k iterations on a single NVIDIA RTX A5000. It takes slightly longer than CLIP-NeRF [5][1] which takes about 9 minutes to train 1k iterations due to our blending operations and additional objectives. Note that our approach can be extended to other more efficient 3D representation methods such as Instant-NGP [4]. The detailed experiments incorporated with Instant-NGP are described in Section E.

A.5. Annealing Thresholds

For the editable amount constraint, we anneal two thresholds: τ^{add} and τ^{change}. We use different target threshold values for each object editing task. Generally, τ^{add} is linearly annealed from 0.8 to the target value during the first 100 steps and remains the target value for the rest of the steps. Similarly, τ^{change} is annealed from 0.5–0.15 to the target value.

A.6. Editable NeRF Architecture

The detailed architecture of editable NeRF using residual blocks is shown in Figure 2. The editable NeRF extends NeRF to produce a density $\sigma^e \in [0, \infty)$ and color $\mathbf{c}^e \in [0, 1]^3$, in addition to two blending ratios $\beta^c \in [0, 1]$ and $\beta^\sigma \in [0, 1]$, respectively.

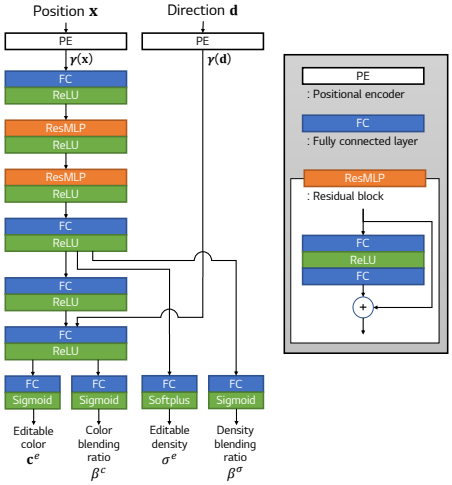

Figure 2: Editable NeRF architecture.

B. Ablation Study

Dilatation operations To investigate the effect of the number of dilatation operations used in localizing the target, we qualitatively compare the editing results with different N_f as shown in Figure 3. As shown in the upper row, the larger the number of dilatation operations, the larger the object is created as the target region grows. In the experiment that only changes the color, if N_f becomes too large, noise appears on the object as shown in the bottom row.

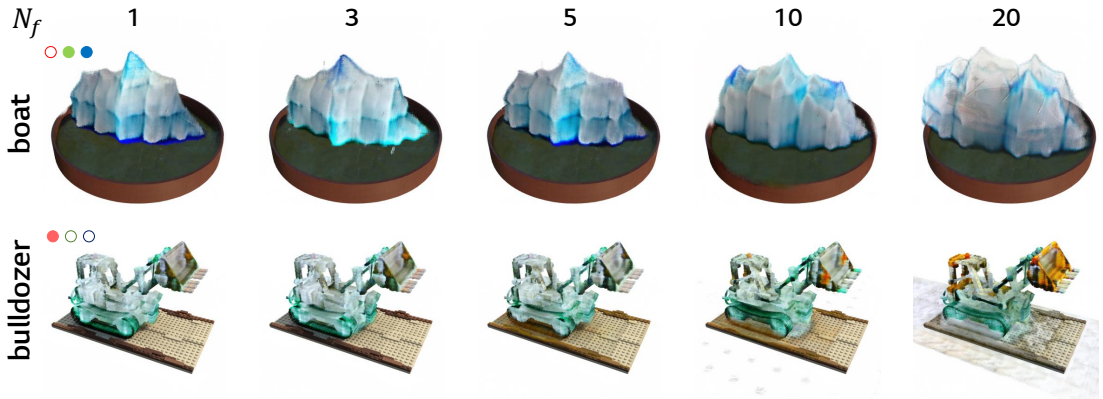

Figure 3: Ablation study on N_f. The *'boat'* object is edited to *'iceberg'* (upper row) with $\tau^{\text{add}} = 0.35$ and $\tau^{\text{remove}} = 0.05$, and the *'bulldozer'* is edited to *'marble bulldozer'* with $\tau^{\text{change}} = 0.2$ (bottom row).

Constraining the amount of editing We also analyze the effect of the thresholds τ^{add} and τ^{change} used in constraining the amount of editing. As shown in the upper row of Figure 4, the larger the threshold τ^{add} for adding densities, the denser the object is created. Similarly, in an experiment that only changes color, the amount of change in the object is limited by the threshold τ^{change}.

[1] https://github.com/cassiePython/CLIPNeRF

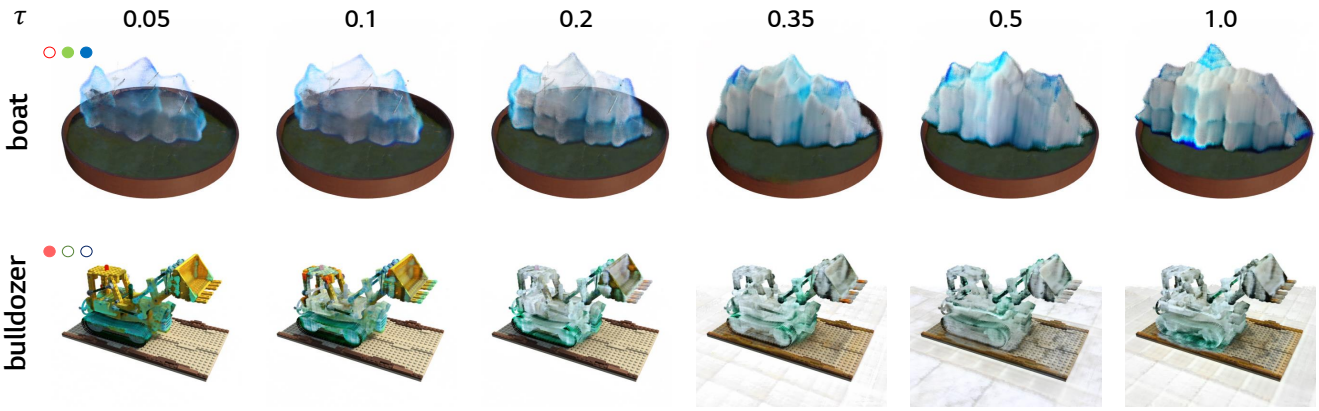

Figure 4: Ablation study on τ. The *'boat'* object is edited to *'iceberg'* (upper row) with $N_f = 10$ and $\tau^{\text{remove}} = 0.05$, and the *'bulldozer'* is edited to *'marble bulldozer'* with $N_f = 3$ (bottom row).

Image and text augmentations We augment text and images to improve the editing quality. Specifically, we augment original, editable, and blended images using differential [6] and random perspective augmentations in the same order as in [2]. Differential augmentations include color jittering, translation, and cut-out with the same setting as [6]. In the case of random perspective augmentation, we set the distortion scale as 0.4 with a probability of 0.5. From a rendered image, we obtain 24 augmented images, including the rendered image itself, and feed them to the image encoder of CLIP. We use the same templates for text augmentations as [1] before feeding the source and target text to the text encoder of CLIP. A random number of randomly selected text templates are applied to the source and target texts to form the augmented texts.

We analyze the effect of augmentations by comparing the results when each augmentation is removed from the overall methods. As shown in Figure 5, without augmenting the rendered images (w/o img aug), the added object has quality degradation, such as blurred and ambiguous boundaries. We also noticed that among differential and perspective augmentations, the former has more effect on the quality improvements (w/o diff). Similarly, without text augmentations (w/o text aug), there is a slight deterioration in quality.

Figure 5: Ablation study on augmentations. The *'boat'* object is edited to *'fantasy modern city'*.

C. User Study

We conducted a user survey to evaluate the performance of our approach against the other three baselines. We asked 30 users to evaluate 20 randomly selected pairs from target text and rendered image pairs in a total of 60 scenes. The rendered images consist of four edited images rendered using each model, including all baseline results and ours, in addition to the original image. For a fair comparison, we randomly shuffled the order of the edited images. Then, each user was asked to assign a score (1–10) for each edited image based on the following criterion: "How well is the image edited to match the target text relative to the amount the original object has changed?". We report the mean scores in Table 1. Our approach obtained the highest user score, demonstrating once again the super performance of Blending-NeRF for text-driven editing.

	CLIP-NeRF-*c*	CLIP-NeRF-*f*	CLIP-NeRF-*D*	Ours
score	3.04	3.99	4.80	**7.17**

Table 1: Human assessment result for comparison with baselines. We report the mean score indicating the precision of text-driven editing by users. Our method outperforms all baselines.

D. Experiments Using User-Provided Mask

Additionally, we introduce a method using a user-provided target region for localized editing. We use this method to address two issues related to the target region. The first is that the text-based image segmentation is not accurate as shown in Figure 6. The second issue is that users may not be able to specify the target region with a text prompt. To address these issues, we manually set the target region mask to be edited from three appropriate viewpoints as shown in Figure 7. Then, in the middle of training (i.e. once every 5 training iterations), we use the user-provided masks. Specifically, when the user-provided masks are given to remove noise as shown in 7-(a), the masks are used once out of 5 training iterations, and the existing segmentation results are used for the rest. On the other hand, if masks are provided to edit a specific region that cannot be specified by a text (*e.g.*, *'**over** the boat'*) as shown in 7-(b), image segmentation is not used. We present some editing results using two kinds of user-provided masks: masks for removing noise and masks for editing a specific area. First, as shown in Figure 8, by giving accurate masks for editing, the existing noise caused by inaccurate segmentation can be removed. Second, as shown in Figure 9, by only using masks specifying the target regions, Blending-NeRF can accurately change the colors of the object (*e.g.*, *'cymbals'* to *'cosmic cymbals'*) and add densities to the scene (*e.g.*, *'bulldozer'* to *'ruby in bulldozer bucket'*).

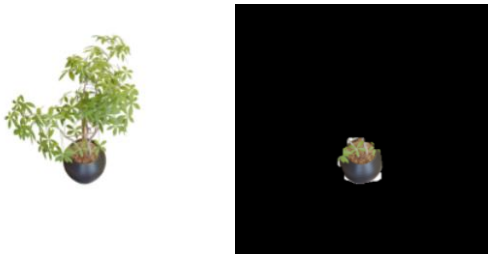

Figure 6: Inaccurate segmentation result for an image and a queried text *'brown-jar'*. This incorrect segmentation may reduce the quality of localized object editing.

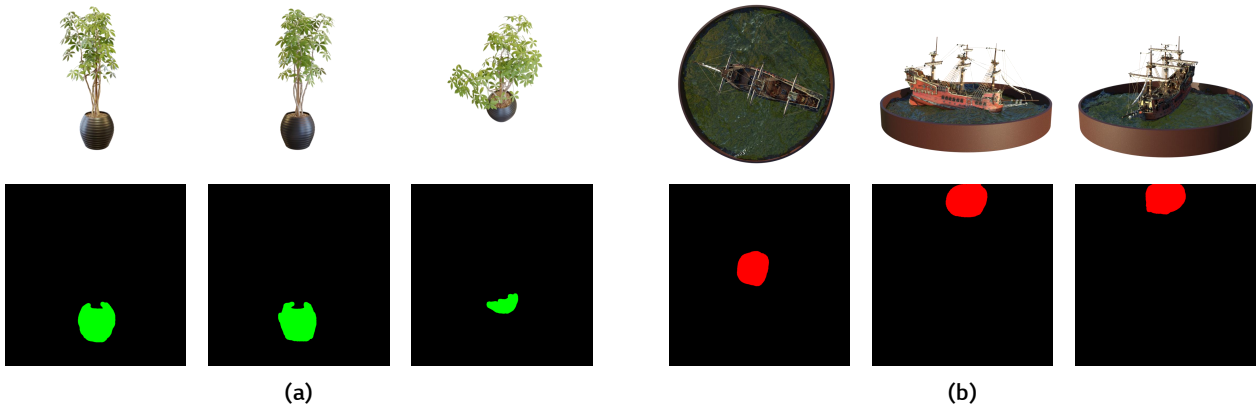

Figure 7: Users can utilize the user-provided image masks (a) to remove noise, or (b) to edit a specific region on the 3D objects.

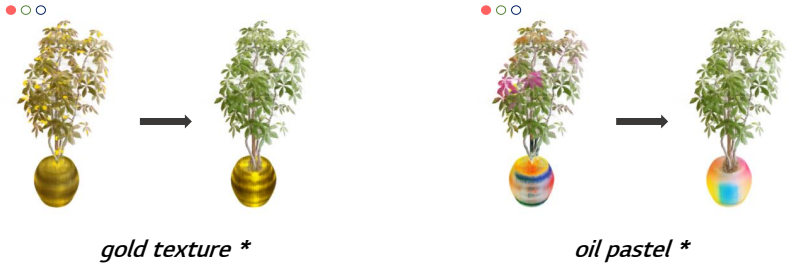

Figure 8: Examples of removing noise. By using user-provided masks, existing noise can be removed.

Figure 9: Examples of object editing using user-provided masks. If it is difficult to specify the target region with the source text, a user-provided mask can be used.

E. Applying Blending-NeRF to Instant-NGP

Our novel layered architecture and blending operations for 3D object editing can be applied to other neural scene representations such as Instant-NGP. To demonstrate our approach can be incorporated with other 3D representation methods, we further experimented using Instant-NGP as a backbone which utilizes multi-resolution hash encoding to represent high-frequency details of a scene with low computational cost.

We used the PyTorch and CUDA based reimplementation[2] of Instant-NGP, and the default settings in the codes were used. We applied the regularization loss $\mathcal{L}_{\text{reg}}$ at the start of training and set its weight as $\lambda_3 = 1.0$. The initial learning rate was set to 1×10^{-3}, and the rest of the hyperparameters were used the same. We trained all the edited scenes for 2k iterations with the patch size $S = 128$, taking about 9 minutes to edit each scene on a single NVIDIA RTX A5000. Figure 10 shows the locally edited 3D objects on the *ship* scene with a wide range of target texts. The experimental results confirm that our approach can be incorporated with the other 3D representation methods.

[2] https://github.com/kwea123/ngp_pl

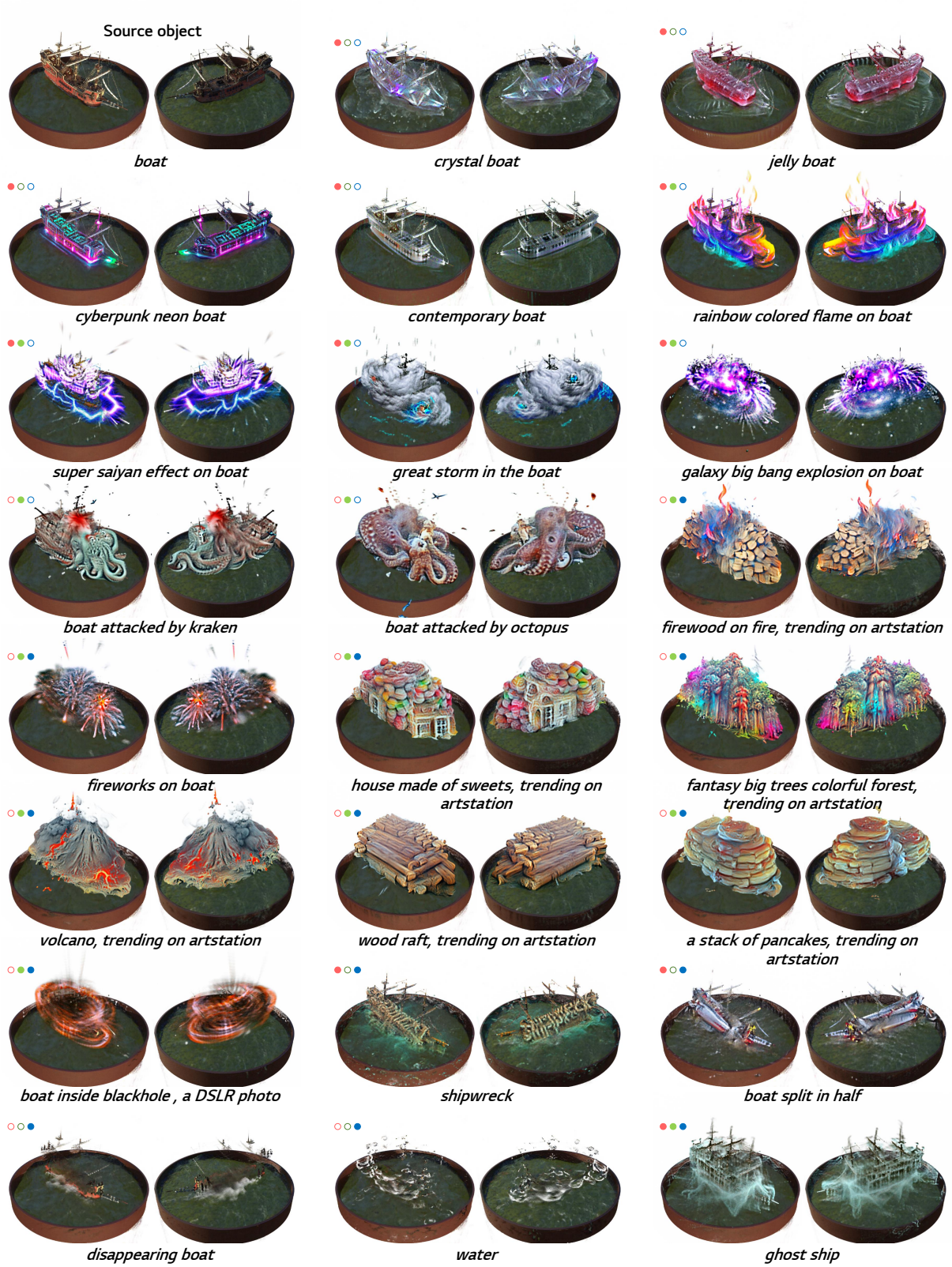

Figure 10: Editing results of our method integrated with Instant-NGP [4]. The source object (*'boat'*) at the top left is edited into each object using the target text at the bottom of each scene. For example, the result in the last column of the second row is edited from *'boat'* to *'rainbow colored flame on boat'*, combining color change (●) and density addition (●).

● : changing colors ● : adding densities ● : removing densities.

References

[1] Omer Bar-Tal, Dolev Ofri-Amar, Rafail Fridman, Yoni Kasten, and Tali Dekel. Text2live: Text-driven layered image and video editing. *arXiv preprint arXiv:2204.02491*, 2022. 3

[2] Han-Hung Lee and Angel X Chang. Understanding pure clip guidance for voxel grid nerf models. *arXiv preprint arXiv:2209.15172*, 2022. 3

[3] Timo Lüddecke and Alexander Ecker. Image segmentation using text and image prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7086–7096, June 2022. 1

[4] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 2, 6

[5] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3835–3844, 2022. 2

[6] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 3