Edit One for All: Interactive Batch Image Editing

Thao Nguyen Utkarsh Ojha Yuheng Li Haotian Liu Yong Jae Lee
University of Wisconsin–Madison
https://thaoshibe.github.io/edit-one-for-all

2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) | 979-8-3503-5300-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/CVPR52733.2024.00790

Abstract

In recent years, image editing has advanced remarkably. With increased human control, it is now possible to edit an image in a plethora of ways; from specifying in text what we want to change, to straight up dragging the contents of the image in an interactive point-based manner. However, most of the focus has remained on editing single images at a time. Whether and how we can simultaneously edit large batches of images has remained understudied. With the goal of minimizing human supervision in the editing process, this paper presents a novel method for interactive batch image editing using StyleGAN as the medium. Given an edit specified by users in an example image (e.g., make the face frontal), our method can automatically transfer that edit to other test images, so that regardless of their initial state (pose), they all arrive at the same final state (e.g., all facing front). Extensive experiments demonstrate that edits performed using our method have similar visual quality to existing single-image-editing methods, while having more visual consistency and saving significant time and human effort.

1. Introduction

Image editing has undergone a transformation in recent years with the aid of modern generative models. The process has been greatly democratized, where many of the sophisticated edits, which previously might have required hours and niche expertise, can now be completed with relative ease in a matter of minutes. For example, different types of learning based algorithms can be used for image correction/adjustment [1–12] and for manipulating the semantic contents of real images [13–22]. Moreover, there are many different ways to edit an image. For instance, a user can specify in text what changes they want - e.g., "*make the hair darker*" [14, 15, 23, 24], or they can *drag* the contents of the image in an interactive manner to shrink, enlarge, or move a part of an object [12, 16, 17, 19, 25].

A common theme across many of these works is that the edits are designed to work for *a single image* at a time. For example, given an image of a cat with open eyes, DragGAN [17] allows the user to drag the contents of *that particular image* so that both the eyes can be closed in the resulting cat image. But what if we wish to apply the same edit to *many* different types of cats so that we can close all of their eyes? While one could perform dragging on each cat separately, the whole process would be quite cumbersome (requiring lots of human annotation) and time intensive (requiring optimization for each image).

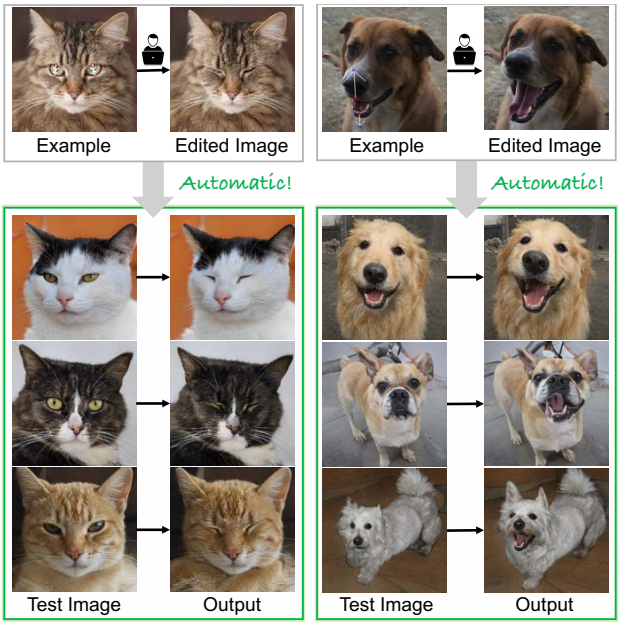

Figure 1. **Interactive Batch Image Editing.** Given a single user edited image, the goal is to automatically transfer that edit to new unseen images, so that all edited images end up with the same final state as the user edited image.

In this paper, we introduce the new problem of *Interactive Batch Image Editing*. Given a user-specified edit $I \rightarrow I'$, the goal is to automatically transfer that edit to new unseen images, so that all edited images end up with the same *final state* as I' (e.g., all cats with eyes closed) regardless of their original starting states (e.g., varying degrees of original eye openness); see Figure 1. Hence, a solution to this problem requires two components: (i) modeling the user edit in the example pair (I, I') so that it can be transferred to new images; and (ii) controlling the degree of the

2575-7075/24/$31.00 ©2024 IEEE
DOI 10.1109/CVPR52733.2024.00790

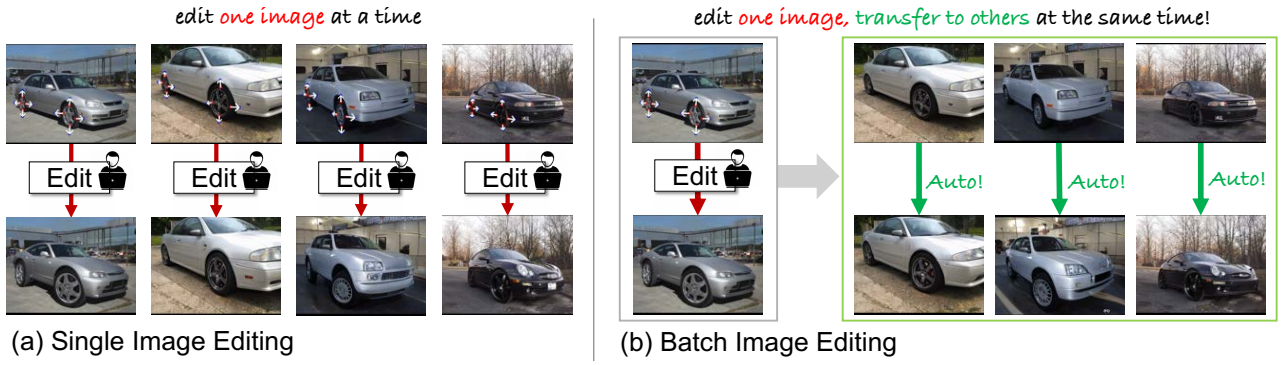

Figure 2. **Single Image Editing vs. Batch Image Editing.** (a) Prior work (e.g., [17, 19, 25]) focuses on single image editing. (b) We focus on batch image editing, where the user's edit on a single image is automatically transferred to new images, so that they all arrive at the same final state regardless of their initial starting state. In this way, we can achieve time speed up and reduce human effort in editing.

edit so that all edited images have the same final state.

To model the user's edit, we build upon existing literature on Generative Adversarial Networks (GANs) [23, 26–29], and in particular StyleGAN [30, 31], which shows that their learned latent space emits globally linear editing directions where the edited attribute (e.g., eye closeness) varies linearly in magnitude along such a direction for all semantically related instances (e.g., all closed eyes cat). To try to discover the global direction corresponding to a user edit, we perform optimization in the latent space of StyleGAN2 [31] so that (i) the edited image with the discovered direction is visually similar to the original edited image I', and (ii) the linear correlation between the distance along the direction and the strength of the visual attribute is highest (e.g., going twice the amount in that direction closes a cat's eyes by twice the amount).

To control the degree of the edit so that all edited images end up with the same final state, we derive a closed-form solution. Taking motivation from [29], we model the attribute strength (e.g., how much the cat's eyes are open) as the distance along the normal vector from a hyperplane. By setting the example edited image as lying on that hyperplane (i.e., its distance is 0), the objective is to move all other images along that same direction so that their distances to the hyperplane also become 0. In this way, regardless of the start state of any image (e.g., varying degree of eye openness), their edited end states all become the same (e.g., closed eyes). We show that we can analytically compute the exact amount of traversal along the direction for any image with a closed-form solution. Editing images in this way helps improve the results visually, as the final state of any new image matches that of the edited example given by the user.

Our method works with various edits given by an editing framework, e.g., it can be a change given by a dragging operation [17], or it can be a change given by a text-based edit [14, 23] (e.g., "make eyes bigger"), as long as it is a direction compatible with a StyleGAN2 [31] model. We present qualitative and quantitative results in transferring edits for a variety of objects (cats/dogs/faces/humans/lions/arts/etc.), parts (mouths/ears/legs/etc.), and corresponding attributes (big eyes/short faces/pale skins/etc.). Importantly, we show that the final state of the edited images is comparable to the scenario in which a user performs the edit for each image separately. But since our method does this automatically (see Fig. 2), we show that it takes much less time (e.g., only 0.05s per image, compared to DragGAN [17] about 2s per image), and does not require laborious human annotation (e.g., only need 1 annotated image, compared to 4 required for DragGAN [17]). Finally, we show practical applications of batch image editing; e.g., changing wheel size in every photo of a car collection by editing just one.

In sum, our contributions are: (1) We introduce the novel problem of interactive batch image editing, wherein a user-driven edit is automatically transferred to other similar test images. (2) We study what *having the same final state* means in a geometrical sense, and propose a principled approach to achieving that in StyleGAN2 [31]'s latent space. (3) We show that our method works on a wide variety of domains e.g., cats, dogs, humans etc., taking significantly less time than editing each image individually while having similar visual quality and more visual consistency.

2. Related Work

Latent space of GANs. The capabilities of generative adversarial networks (GANs) have transformed drastically, from the first GAN [32] designed for simple image datasets like MNIST [33] to big and powerful models like BigGAN [34], Progressive GAN [35] and StyleGANs [30, 31] designed for much more complex datasets [36, 37]. Parallel to the efforts to enhance their capabilities, there has also been work done to better understand and make use of their learned latent space. Specifically, there is a line of work which tries to find interpretable *directions* in that space, so that moving in such a direction majorly changes only one discernible attribute in the image. Some methods [38, 39]

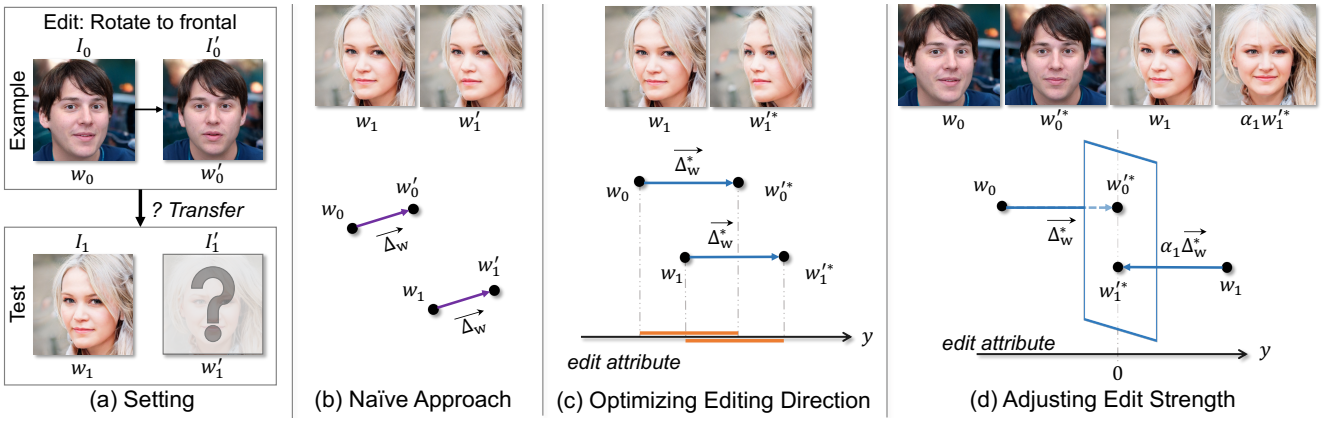

Figure 3. **Different editing strategies.** (a) Setting. (b) Naive Approach: The editing direction effective for an example may not generalize well to test images. (c) Optimizing Editing Direction: We optimize for a globally consistent direction that is effective for both example and test images. (d) Adjusting Editing Strength: Ensuring consistent final states requires adjusting the editing strength for each test image.

try to find such directions or particular activations using (self) supervised methods. Others [26–28] try getting rid of the need for supervision: discovering the directions representing the most important factors of variation, whatever they may be. Going beyond the $\mathcal{W}$ space, the authors in [40] explore the $\mathcal{W}+$ space, and show that it has even better disentanglement useful for spatial image editing. All of these hidden capabilities have made StyleGANs a useful tool for editing purposes. However, prior work mainly focuses on the effectiveness of the editing direction (e.g., whether the discovered direction can change yaw pose). The question of whether these directions can be applied to diverse images to yield consistent results (e.g., all faces facing frontal) and, if so, how to achieving such consistency, remains unanswered.

Image editing with generative models. Researchers have used StyleGAN for *real* image editing by designing an encoder to invert a real image into StyleGAN's latent space [41–43]. StyleCLIP [23] presents a way to perform image editing through a text-based interface by making use of the CLIP encoder [44]. Very recently, StyleGAN has also been used to perform point-based editing [17, 25] so as to move any start point to reach a target location in the image; thereby elongating, rotating, shifting the objects. Most work on GANs has been for single object category datasets, but with the rise of text-to-image Diffusion Models which can generate complex images [13, 45, 46], editing such images is now possible. Complex scenes can be edited either using text [14, 15, 21, 24, 47, 48], or using the same idea of point-based manipulation [16, 19]. There have been some works which discuss the possibility to transfer the edit of one image to another. EditGAN [49] uses segmentation mask manipulations to edit an image, but to successfully transfer them to an unseen image, it needs to do post hoc manipulation of the editing scale. RewriteGAN [50] involves editing one generated sample; e.g., adding a patch of a tree onto a church tower. Following this, the rules of the GAN are manipulated so that all churches have *some* tree on the top. Within diffusion models, visual prompting/ image analogies tries something similar, where users can define a triplet {before, after, test} to learn the edit and transfer it to a test image [51–55]. Our task is similar, in that we wish to transfer user-edits from the training example to new images, but it differs in one crucial aspect: along with transferring the edit, we wish to automatically learn its strength so that the edit produces the same final state for a new image.

3. Approach

Our focus is on the setting in which an image editing process edits an *attribute* of an image so that the attribute's *value* reaches a desired state; e.g., rotating the pose of a face looking sideways so that it faces front (attribute = face's pose, value = front).

With this view, we explain our framework for *batch image editing*, which can be broken down into two stages: (i) A user edits an image I_0 (e.g., using DragGAN [17]) to obtain an edited image I_0'. We describe in Sec. 3.1 how we capture this user edit $I_0 \rightarrow I_0'$, so that the same edit can be applied to new images. (ii) Next, we describe in Sec. 3.2, how for any new image, we apply the modeled edit by automatically adjusting its *strength* so that the attribute's value in this new image matches that of I_0' (e.g., any face, regardless of its initial pose, now faces front after the edit).

3.1. Modeling the User Edit

We start with an image I_0. This image could be a real image or a generated one. Either way, we get its latent representation in the $\mathcal{W}$ space of a StyleGAN2 [31] model G, so that $I_0 = G(w_0)$. (For real images we can use GAN-inversion techniques [56].) The user, with the help of an image editing framework (e.g., DragGAN [17] or InstructPix2Pix [14]), edits I_0 to manipulate one or more of its attributes. The edit maps to the $\mathcal{W}$ space as $w_0 \rightarrow w_0'$. The

resulting edited image can thus be recovered as follows: $I'_0 = G(w'_0)$. Fig. 3(a) shows an example of the original and edited image pair, (I_0, I'_0), where the user intended to turn the face forward.

Now, given the user edit $I_0 \rightarrow I'_0$, we wish to capture it in a way that can be applied to new images in a generalizable manner; i.e., the application of the edit changes the *same* property in a new image. This is where a nice property of GANs, and in particular StyleGANs [30, 31] comes in useful. It has been shown in prior works [26–28] that it is possible to discover directions with such properties (using supervised as well as unsupervised methods) in the learned $\mathcal{W}$ space. In particular, it is possible to find directions (Δ_w) that are *globally consistent*. Taking motivation from [29], we define a globally consistent direction Δ_w as the following: for any arbitrary w, moving along Δ_w, $w \rightarrow w + \Delta_w$ (i) changes the same attribute, and (ii) by the same amount.

To make the precise user edit $I_0 \rightarrow I'_0$ applicable to other images, it needs to be captured as a globally consistent direction. The naive way to represent that edit will be through a simple difference in the $\mathcal{W}$ space: $\Delta_w = w'_0 - w_0$. However, empirically, and as we show in Fig. 3(b), applying this Δ_w to the latent code w_1 corresponding to a new image I_1 *does not* always result in the same change; while $I_0 \rightarrow I'_0$ results in a pose change of $\sim$30°degrees in yaw, $I_1 \rightarrow I'_1$ does not seem to change the same attribute or at least not by the same amount.

Hence, our goal is to represent the $I_0 \rightarrow I'_0$ edit through a $\mathcal{W}$ space direction that can better satisfy both the properties of a globally consistent direction. For this task, we first take motivation from LARGE [29] to introduce a mathematical view of what those directions mean.

Globally consistent direction. Let's say that Δ_g is one such direction. Along with a bias term b, we can define a hyperplane as follows:

$$w \cdot \Delta_g + b = 0 \tag{1}$$

That is, any point w which lies on the hyperplane will satisfy this equation. The authors in [29] argued that for such a hyperplane, whose normal vector is a globally consistent direction (Δ_g), the distance of an arbitrary point w from that hyperplane $(w \cdot \Delta_g)$ will be linearly correlated to the actual attribute (y) that results in the generated image:

$$y = a \times (w \cdot \Delta_g) + b \tag{2}$$

Here, a and b are unknown linear coefficients. For example, if Δ_g corresponds to *change in pose*, then all the front facing people will lie at the same distance d from the above hyperplane. Because of this, any hyperplane defined with respect to a globally consistent direction can be viewed as a semantic hyperplane. For simplicity, we can set $d = 0$ for the edited image I'_0.

Given that the original Δ_w may not always be globally consistent, we aim to discover, through optimization, a different direction Δ^*_w which produces a similar editing effect, but is more likely to be a globally consistent one. We initialize Δ^*_w with 0's, and design two objective functions to optimize it. First, to make sure that it produces a similar effect as the original direction, we use an image reconstruction loss so that the edited image produced by the new edit matches the original edit given by the user:

$$\mathcal{L}_{img} = \|G(w_0 + \Delta_w) - G(w_0 + \Delta^*_w)\|_2 \tag{3}$$

Additionally, user can provide the real value of distance d, otherwise we set the distance of the edited image from the hypothetical hyperplane to be 0 (similar to [29]). We constrain the learned Δ^*_w to follow this property for better interpretability. That is, the new direction should be such that when we traverse the original latent code in that direction, $w_0 \rightarrow w_0 + \Delta^*_w$, the resulting image should lie at the hyperplane defined by that direction. Setting $b = 0$ in Eq. 1, we minimize the edited point's distance from the hyperplane:

$$\mathcal{L}_{att} = |(w_0 + \Delta^*_w) \cdot \Delta^*_w| \tag{4}$$

The overall loss function for optimizing the new direction Δ^*_w is a weighted sum of the two objectives: $\mathcal{L}_\Delta = \mathcal{L}_{img} + \lambda \mathcal{L}_{att}$. We set λ to 0.02 in all experiments to balance the magnitude of $\mathcal{L}_{img}$ and $\mathcal{L}_{att}$.

Now, in Fig. 3 (b) and (c), we compare the difference in image editing results using two different directions when applied to the same new image (I_1): $w_1 \rightarrow w_1 + \Delta_w$ vs. $w_1 \rightarrow w_1 + \Delta^*_w$. As discussed before, in this case, the editing using the naively computed direction (Δ_w) is not able to accurately capture the pose rotation the way it did for the user edited example $(I_0 \rightarrow I'_0)$. On the other hand, with Δ^*_w, we see that the pose of the woman changes by a similar amount as the edited example. However, since the original pose of the woman (I_1) was not the same as the pose of the man (I_0), the new edited image (I'_1) still does not arrive at the same final state as I'_0. Therefore, our next goal is to figure out how to scale the learned direction so that I'_1 does arrive at the same final state as I'_0.

3.2. Adjusting Editing Strength for New Images

Given the optimized Δ^*_w and some new image $I_i = G(w_i)$, we wish to edit it in the following way:

$$w'_i = w_i + \alpha_i n \tag{5}$$

where α_i is the editing strength computed separately for each image and $n = \Delta^*_w / ||\Delta^*_w||$ is a unit vector in the direction of Δ^*_w. Fig. 3 (d) illustrates a geometric perspective that we will use to compute α_i. First, we see Δ^*_w represented as a vector and its corresponding hyperplane that is normal to it. Next, we depict the new latent point w_i. The

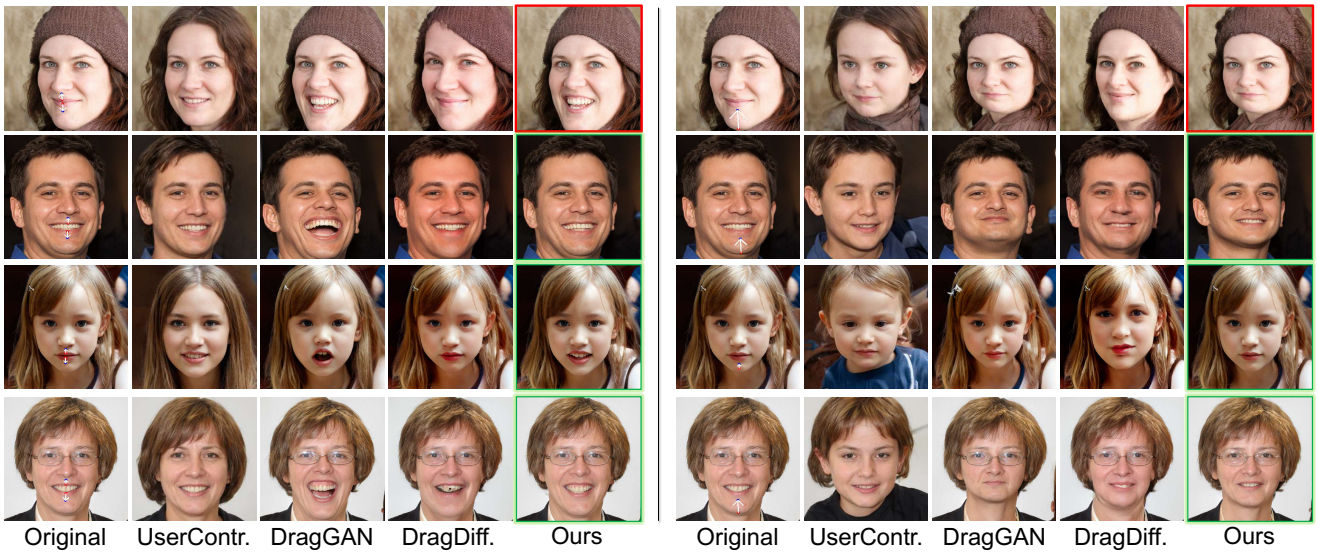

Figure 4. Qualitative comparisons between dragging baselines. For ours, green bounding box indicates automatic transfer from the red bounding box example in the first row (i.e., no point annotation needed!).

goal is to move it along the Δ_w^* direction so that it arrives *onto* the hyperplane. Through this depiction, α_i can be understood as the distance of w_1 from the hyperplane. We can get the distance by projecting $(w_0' - w_i)$ in the direction of n. Therefore, $\alpha_i = (w_0' - w_i) \cdot n$.

Since for each image, the only unique computation that needs to be done is the calculation of α_i, we can see why our method will be much faster than, for example, annotating every image and running the DragGAN [17] optimization each of those times. We will show in our experiments the significant difference in time taken by our method compared to single image editing baselines.

Importantly, there is an added benefit to computing α's in this way. Let's say the goal of the editing process is to rotate n faces and make them frontal. After completion, all of them do become frontal, each with their own editing strengths $\{\alpha_1, \alpha_2, ..., \alpha_n\}$ computed using the above formula. Suppose the user now wants the same faces facing a bit left instead. To do this, the user *does not* need to re-annotate the original training example and run the optimization one more time. In an interactive manner, they can simply scale the α for the training example to match the desired edit, and all other α's can be automatically recomputed.

4. Experiments

We study how well our method models and transfers the edits from an example, and how efficient it is compared to single image editing baselines using their official code.

Categories. We evaluate on a variety of domains: Human faces (FFHQ) [30], AFHQ Cats, Dogs [57], MetFaces [58], Human bodies [59]. For each domain, we use the corresponding pretrained StyleGAN model to perform editing.

4.1. Qualitative results

We start with qualitative comparisons with (i) interactive point-dragging, and (ii) text-based image editing baselines.

Interactive point based editing. We compare against DragGAN [17], UserControllableLT [25], and DragDiffusion [16]. For each baseline, we, as a user, desire to edit a bunch of images to have the same final state for an attribute. Fig. 4 shows the results of editing two kinds of edits to four images (leftmost column). In the left case, the goal is to make everyone smile by the same degree. In the right, it is to vertically compress everyone's face by the same degree. For all three baselines, we manually annotate points for each test image. For our method, we only annotate points and perform DragGAN-based edit on one image (top row), then automatically transfer the edit to the three other test images.

For the smiling case (left), we notice that when DragGAN drags the upper/lower lips up/down respectively, it can either make people smile (1st, 2nd and 4th rows), or it can make people look shocked (blonde girl; 3rd row). Both edits are technically correct based on lips movement, but it won't match the user expectation of making everyone smile by the same degree. Since all four images are edited independently, this is not unexpected. The other two baselines sometimes have issues introducing the same edit across different images; e.g., UserControllableLT can make the blonde girl smile a little bit, but not the other images. In general, we notice in our experiments that to introduce the same exact edit that we want, we often need to play around with some hyperparameters (e.g., # of iterations). In comparison, our method produces edited images that are smiling more equally (left) and have been compressed by a similar

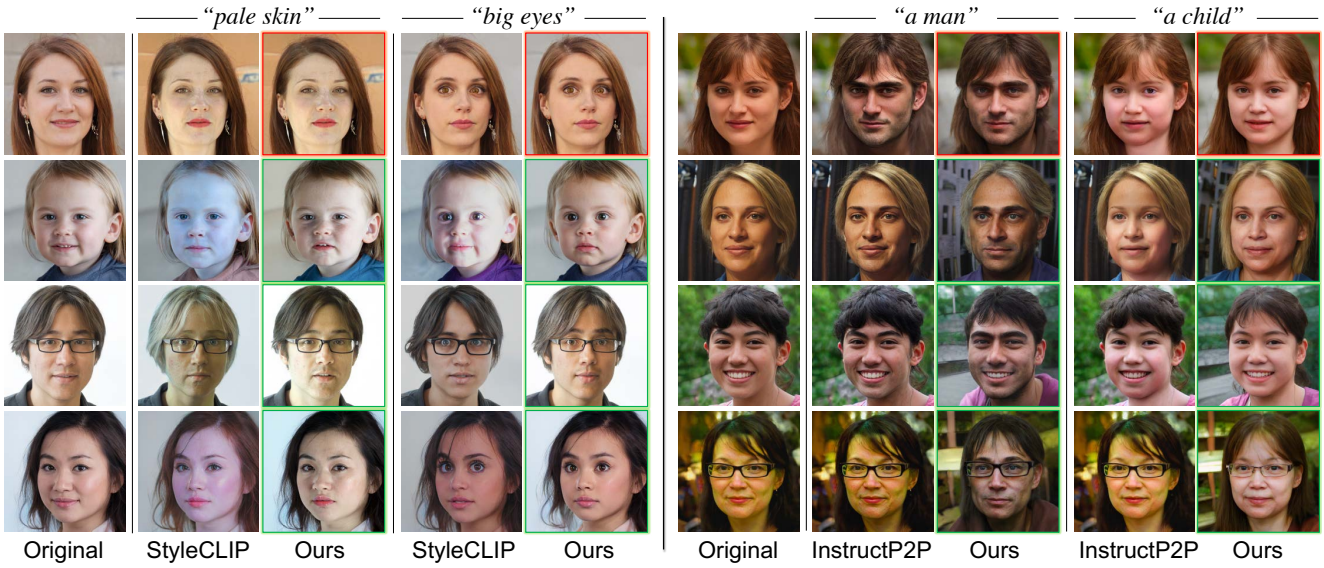

Figure 5. Qualitative comparisons to text-guided baselines. Ours transfers the edit from example (red), to other test images (green).

amount (right), and with much less human effort.

Text-driven image editing. We consider two baselines: StyleCLIP [23] and InstructPix2Pix [14]. Each takes an image and text as input to produce an edited image: $I \rightarrow I'$. Fig. 5 shows the results of Ours vs. StyleCLIP (left) and Ours vs. InstructPix2Pix (right). (For the latter, we invert the output of InstructPix2Pix into StyleGAN's latent space using [42] for our method.) In each case, for the baselines, we use the same text prompt, e.g., "A photo of a person with pale skin" for each edited image (in four rows). For ours, we capture the edit from the first editing result of a baseline (first row), and then automatically transfer the edit to the remaining three images.

Our goal with this particular setup is to test (i) how consistently the baselines introduce the edit denoted by text to different images (e.g., do all people become equally pale?), and (ii) how consistently our method captures and transfers the *particular* edit of the first example to the rest of the images, irrespective of how good that first edit was. For (i), we find that StyleCLIP sometimes has issues; e.g., the *paleness* of the edited face in first example is different from the second (first vs second rows). For InstructPix2Pix, sometimes it can give consistent results; the *childness* of different faces seems similar. But, it can have consistency issues in other examples; e.g., the *type/strength* of maleness introduced is different in each image. And this is where, we believe, the utility of our method lies: if one desires to edit every face to be *male* in a particular way (i.e., to the same degree) as the first example, e.g., thick eyebrows & light beard, our method has an advantage.

Finally, we show our method's results for non-facial domains in Fig. 6. For the lion examples (top left), we see that it can preserve not just the type of edit, e.g., dragging to make the lion roar, but also the strength of the roar. The strength being preserved can be observed more easily for the human body poses (bottom left), where we see that the extent of legs split, hand movements, is consistent enough to almost align the edited test images with the user edited one. Overall, results on these diverse set of domains highlight an observation that many kinds of edits can be thought of as a combination of {type, strength}, both of which can indeed be captured and transferred according to our needs.

Method	Point Dragging		Time		Anno.
	Dist.	FID	1 img	1k imgs	# imgs
UserControl. [25]	13.64	25.32	0.03s	30s	1000
DragGAN [17]	4.165	9.28	2s	33.33m	1000
DragDiffusion [16]	26.56	36.55	60s	16.67h	1000
Ours	9.467	9.35	2s	82s	1

Table 1. Time is estimated for 1 point drag, without human annotation time. (82s includes 2s to perform edit on the example, 30s to optimize the editing direction, and 50s to transfer the edit to 1000 test images.) Our method requires only one image annotation in total, while the baselines need one annotation per test image.

4.2. Quantitative results

Next, perform quantitative experiments following the setup proposed in [16, 17, 19]. We first randomly sample 10 facial test images and pair them with a randomly sampled target face image. We use dlib-ml [60] to detect keypoints in the test and target images. Our goal is to see if the keypoints in the test images can be moved to the target locations specified by keypoints in the target image. For the baselines [16, 17, 25], we perform dragging for each test image. For our method, we perform dragging on *one* test image, and then transfer the edit to the remaining nine images. This is repeated 100 times; i.e., each time a random target image is

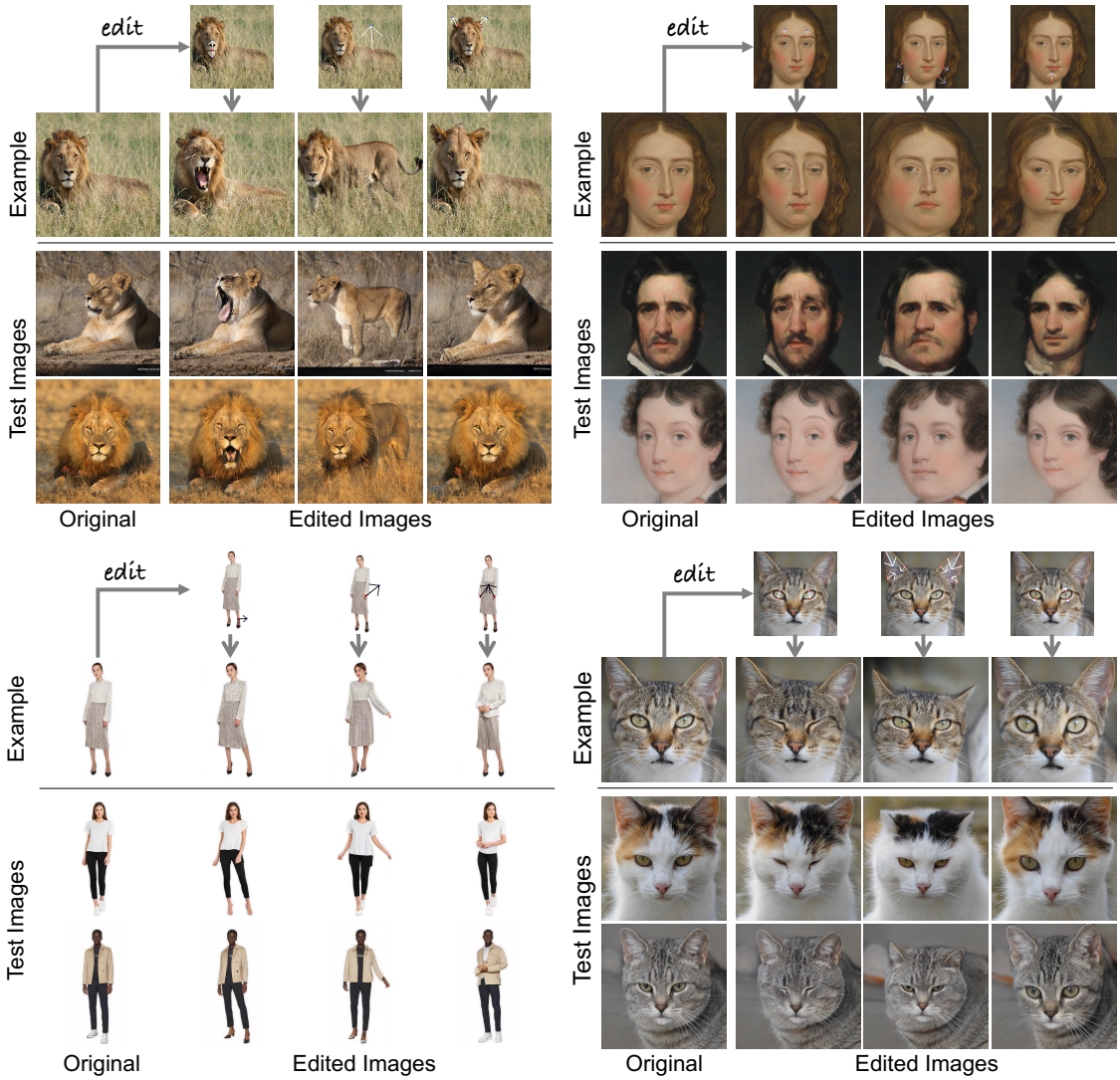

Figure 6. Additional qualitative results on various domains.

paired with 10 random test images.

We report the Euclidean distance between the keypoints of edited and target images in Table 1. Even without requiring annotations for every test image, our method can move the points very close to the target; almost as close as DragGAN, which requires annotation each time. We also compute FID [61] between the original test images and their edited versions to ensure that our method does not distort the image; and our quality is comparable to DragGAN.

5. Deeper Analysis

We perform deeper analysis on the usefulness of our batch image editing framework by focusing on a specific edit: face pose rotation. In particular, if we wish to rotate many faces to front, how do we visualize the improvements brought by different components of our method?

Fig. 7 depicts the setup. We use DragGAN to perform dragging on one image (top; $w_0 \rightarrow w_0'$). We then transfer the edit to 1000 other test images (two shown in rows 2-3). We see that *naively* applying $\Delta_w = w_0' - w_0$ from DragGAN to test images cannot make them frontal (0°). Using our optimized Δ_w^* (without dynamic scaling) does help it bring closer to facing front. Does this mean that Δ_w^* might be a more *global* direction? We study this in Fig. 8 (a), where we visualize the degree to which the correlation property of global directions (Eq. 2) holds for Δ_w and Δ_w^* individually. We see that there is indeed a better correlation (R^2) between the distances of latents from hyperplanes and the Yaw degree of resulting images. (We predict Yaw using 6DRepNet [62].) However, as we discussed before, Δ_w^* in itself is insufficient: to completely bring the facial pose to front, we need to scale it with corresponding α's to bring them much closer to front (Fig. 7, 4th column). To study this in a more systematic way, we visualize the effect on all 1000 test images. Similar to Fig. 8 (a), we visualize

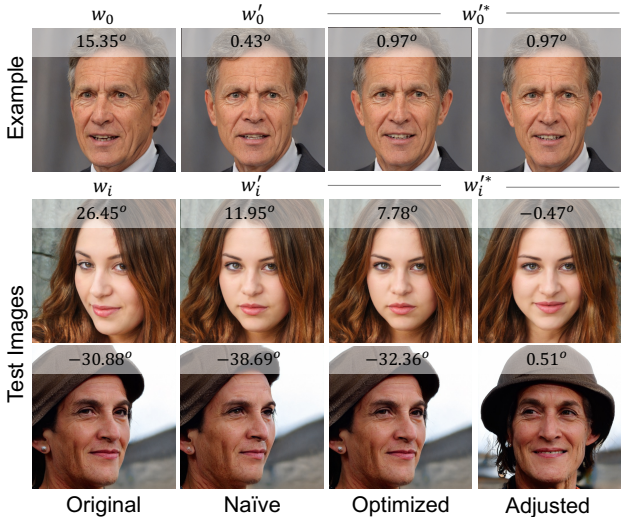

Figure 7. Effect of adjusted and optimized editing for test images. Yaw degree of each image is provided on top of each images.

Method	MAE	Time complexity	
		Prepare	Inference
Random	11.295 ± 8.972	-	-
DragGAN [17] + GANgealing [63]	8.141 ± 7.221	30s	2000s
Ours	2.120 ± 1.818	32s	50s

Table 2. Ablation Study: DragGAN + GANgealing.

distance-to-hyperplane vs. Yaw degree for the original test images (blue) and edited images (red) in Fig. 8 (b). We can see that the variation in Yaw for the edited images *collapses* at around 0°; i.e., they mostly face front as we would like.

We also compute the time it takes to perform editing in Table 1, last three columns. Our method only requires annotating one image, while the baselines require annotating each image (e.g., 1000 total in this case). Note the times do not include human annotation time.

Other ways to automate batch image editing. One baseline for batch image editing (i.e., so that all edited images achieve the same final state) is shown in Fig. 9. After the user annotates the source/target points in the example image, we use GANgealing [63] to transfer the points to corresponding locations in each new test image. The edited images are then obtained using DragGAN (DragGAN + GANgealing). We compare our method to this baseline when the user edits one face image to become front (Yaw=0°) and transferring it to 1000 other test images.

Results are shown in Table 2. We report mean absolute error (MAE) between the Yaw degree of the edited image and its ideal frontal image (Yaw=0°). 'Random' shows the variations in Yaw for the original images (before editing). While 'DragGAN + GANgealing' does help in reducing the variation (8.14 < 11.29), our edited images are much more *frontal*, with an MAE of 2.12. This is because the baseline has no way to bring all edited images to the same final state. On top of that, since our method does not rely on

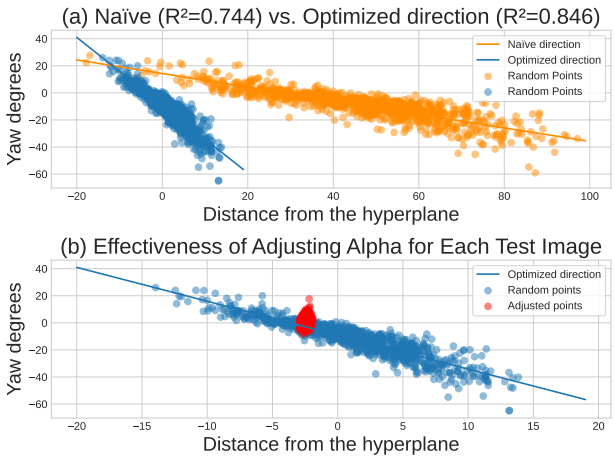

Figure 8. (a) After optimization, the editing direction is more linearly correlated with the yaw attribute. (b) With automatically adjusted editing scale for each test image.

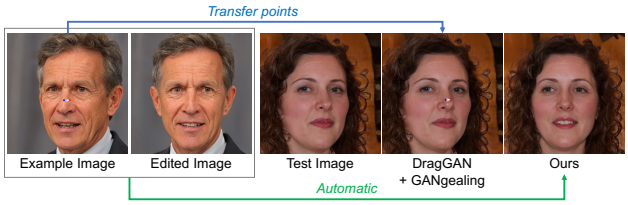

Figure 9. Ablation study set up. We use GANgealing [63] to transfer the annotated points from example to test image.

additional information about keypoints (which 'DragGAN + GANgealing' does), we believe that it is better suited for batch image editing for many kinds of domains (e.g., Fig. 6), where we might not have such information.

6. Conclusion

We introduced the problem of interactive batch image editing. Given a user edit in an example image, our approach automatically transfers that edit to other test images, maintaining a consistent final state of the edit across images. Extensive experiments demonstrated that our method produces comparable quality to state-of-the-art single-image-editing methods while saving significant time and human effort. We are currently limited to StyleGAN models. Extending this problem and solution to diffusion-based models for more edits types would be an exciting future direction.

Acknowledgement

This work was supported in part by NSF CAREER IIS2150012, Adobe Data Science award, Sony Focused Research award, and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2022-00187238, Development of Large Korean Language Model Technology for Efficient Pretraining).

References

[1] Man M. Ho and Jinjia Zhou. Deep preset: Blending and retouching photos with color style transfer. In *WACV*, 2021. 1

[2] Mahmoud Afifi, Konstantinos G. Derpanis, Bjorn Ommer, and Michael S. Brown. Learning multi-scale photo exposure correction. In *CVPR*, 2021.

[3] Jong-Hyeon Baek, DaeHyun Kim, Su-Min Choi, Hyo-jun Lee, Hanul Kim, and Yeong Jun Koh. Luminance-aware color transform for multiple exposure correction. In *ICCV*, 2023.

[4] Wenyan Cong, Jianfu Zhang, Li Niu, Liu Liu, Zhixin Ling, Weiyuan Li, and Liqing Zhang. Dovenet: Deep image harmonization via domain verification. In *CVPR*, 2020.

[5] Yi-Hsuan Tsai, Xiaohui Shen, Zhe Lin, Kalyan Sunkavalli, Xin Lu, and Ming-Hsuan Yang. Deep image harmonization, 2017.

[6] Zonghui Guo, Haiyong Zheng, Yufeng Jiang, Zhaorui Gu, and Bing Zheng. Intrinsic image harmonization. In *CVPR*, 2021.

[7] Jun Ling, Han Xue, Li Song, Rong Xie, and Xiao Gu. Region-aware adaptive instance normalization for image harmonization. In *CVPR*, 2021.

[8] Yifan Jiang, He Zhang, Jianming Zhang, Yilin Wang, Zhe Lin, Kalyan Sunkavalli, Simon Chen, Sohrab Amirghodsi, Sarah Kong, and Zhangyang Wang. Ssh: A self-supervised framework for image harmonization. In *ICCV*, 2021.

[9] Wenyan Cong, Xinhao Tao, Li Niu, Jing Liang, Xuesong Gao, Qihao Sun, and Liqing Zhang. High-resolution image harmonization via collaborative dual transformations. 2022.

[10] Jeya Maria Jose Valanarasu, He Zhang, Jianming Zhang, Yilin Wang, Zhe Lin, Jose Echevarria, Yinglan Ma, Zijun Wei, Kalyan Sunkavalli, and Vishal M Patel. Interactive portrait harmonization. 2022.

[11] Lingzhi Zhang, Tarmily Wen, and Jianbo Shi. Deep image blending. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 231–240, 2020.

[12] Dave Epstein, Allan Jabri, Ben Poole, Alexei A. Efros, and Aleksander Holynski. Diffusion self-guidance for controllable image generation. *arXiv preprint arXiv:2306.00986*, 2023. 1

[13] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. 2021. 1, 3

[14] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. 2022. 1, 2, 3, 6

[15] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *ICLR*, 2022. 1, 3

[16] Yujun Shi, Chuhui Xue, Jiachun Pan, Wenqing Zhang, Vincent YF Tan, and Song Bai. Dragdiffusion: Harnessing diffusion models for interactive point-based image editing. 2023. 1, 3, 5, 6

[17] Xingang Pan, Ayush Tewari, Thomas Leimkühler, Lingjie Liu, Abhimitra Meka, and Christian Theobalt. Drag your gan: Interactive point-based manipulation on the generative image manifold. In *SIGGRAPH*, 2023. 1, 2, 3, 5, 6, 8

[18] Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. Diffedit: Diffusion-based semantic image editing with mask guidance, 2022.

[19] Chong Mou, Xintao Wang, Jiechong Song, Ying Shan, and Jian Zhang. Dragondiffusion: Enabling drag-style manipulation on diffusion models. 2023. 1, 2, 3, 6

[20] Binxin Yang, Shuyang Gu, Bo Zhang, Ting Zhang, Xuejin Chen, Xiaoyan Sun, Dong Chen, and Fang Wen. Paint by example: Exemplar-based image editing with diffusion models. *arXiv preprint arXiv:2211.13227*, 2022.

[21] *Adding Conditional Control to Text-to-Image Diffusion Models*, 2023. 3

[22] *GLIGEN: Open-Set Grounded Text-to-Image Generation*, 2023. arXiv:2301.07093. 1

[23] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In *ICCV*, 2021. 1, 2, 3, 6

[24] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *CVPR*, 2023. 1, 3

[25] Yuki Endo. User-controllable latent transformer for stylegan image layout editing. *Computer Graphics Forum*, 2022. 1, 2, 3, 5, 6

[26] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. In *NeurIPS*, 2020. 2, 3, 4

[27] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *CVPR*, 2020.

[28] Yujun Shen and Bolei Zhou. Closed-form factorization of latent semantics in gans. In *CVPR*, 2021. 3, 4

[29] Yotam Nitzan, Rinon Gal, Ofir Brenner, and Daniel Cohen-Or. Large: Latent-based regression through gan semantics. In *CVPR*, 2022. 2, 4

[30] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. 2019. 2, 4, 5

[31] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *CVPR*, 2020. 2, 3, 4

[32] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 2

[33] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998. 2

[34] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv*, 2018. 2

[35] Terro Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*, 2018. 2

[36] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 2

[37] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. 2015. 2

[38] Ali Jahanian, Lucy Chai, and Phillip Isola. On the "steerability" of generative adversarial networks. In *ICLR*, 2020. 2

[39] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 2020. 2

[40] Zongze Wu, Dani Lischinski, and Eli Shechtman. Stylespace analysis: Disentangled controls for stylegan image generation. In *ICLR*, 2020. 3

[41] Tengfei Wang, Yong Zhang, Yanbo Fan, Jue Wang, and Qifeng Chen. High-fidelity gan inversion for image attribute editing. In *CVPR*, 2022. 3

[42] Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. Designing an encoder for stylegan image manipulation. In *ACM Transactions on Graphics*, 2021. 6

[43] Daniel Roich, Ron Mokady, Amit H Bermano, and Daniel Cohen-Or. Pivotal tuning for latent-based editing of real images. *ACM Trans. Graph.*, 2021. 3

[44] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. 2021. 3

[45] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. 2021. 3

[46] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2020. 3

[47] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. *arXiv preprint arXiv:2211.09794*, 2022. 3

[48] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022. 3

[49] Huan Ling, Karsten Kreis, Daiqing Li, Seung Wook Kim, Antonio Torralba, and Sanja Fidler. Editgan: High-precision semantic image editing. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 3

[50] David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. Rewriting a deep generative model. In *ECCV*, 2020. 3

[51] Amir Bar, Yossi Gandelsman, Trevor Darrell, Amir Globerson, and Alexei A. Efros. Visual prompting via image inpainting. 2022. 3

[52] Thao Nguyen, Yuheng Li, Utkarsh Ojha, and Yong Jae Lee. Visual instruction inversion: Image editing via visual prompting. In *NeurIPS*, 2023.

[53] Yasheng Sun, Yifan Yang, Houwen Peng, Yifei Shen, Yuqing Yang, Han Hu, Lili Qiu, and Hideki Koike. Imagebrush: Learning visual in-context instructions for exemplar-based image manipulation. 2023.

[54] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 2023.

[55] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. Visual attribute transfer through deep image analogy. 2017. 3

[56] Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, and Ming-Hsuan Yang. Gan inversion: A survey. *TPAMI*, 2022. 3

[57] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *CVPR*, 2020. 5

[58] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data, 2020. 5

[59] Jianglin Fu, Shikai Li, Yuming Jiang, Kwan-Yee Lin, Chen Qian, Chen-Change Loy, Wayne Wu, and Ziwei Liu. Stylegan-human: A data-centric odyssey of human generation. *arXiv preprint*, arXiv:2204.11823, 2022. 5

[60] Davis E. King. Dlib-ml: A machine learning toolkit. *J. Mach. Learn. Res.*, 2009. 6

[61] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Gunter Klambauer, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017. 7

[62] Thorsten Hempel, Ahmed A. Abdelrahman, and Ayoub Al-Hamadi. 6d rotation representation for unconstrained head pose estimation. In *ICIP*, 2022. 7

[63] William Peebles, Jun-Yan Zhu, Richard Zhang, Antonio Torralba, Alexei Efros, and Eli Shechtman. Gan-supervised dense visual alignment. In *CVPR*, 2022. 8