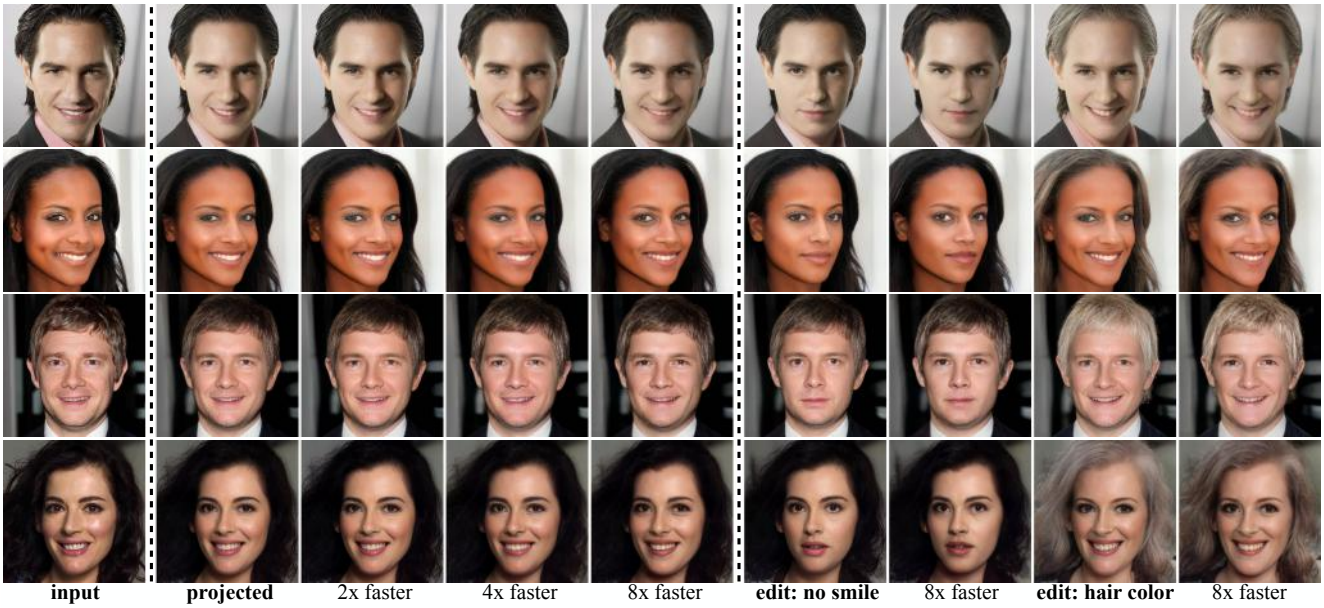
Anycost GANs for Interactive Image Synthesis and Editing

Ji Lin[1,2*] Richard Zhang[2] Frieder Ganz[2] Song Han[1] Jun-Yan Zhu[2,3]
[1]MIT [2]Adobe Research [3]CMU
{jilin, songhan}@mit.edu {rizhang, ganz}@adobe.com junyanz@cs.cmu.edu



(a) input images (b) Anycost GANs provide consistent outputs for projected latent code (c) Consistency remains after editing

Figure 1: Anycost GAN can be executed at flexible computation costs (fast preview with low cost and high-quality output with full cost), enabling interactive image editing with quick preview. The low-cost sub-generator produces consistent outputs compared to the full-cost generator during both image projection and latent code traversal, making the sub-generator an accurate proxy for various editing tasks (*e.g.*, no smile, changing hair color). Interactive demos are available here.

Abstract

Generative adversarial networks (GANs) have enabled photorealistic image synthesis and editing. However, due to the high computational cost of large-scale generators (e.g., StyleGAN2), it usually takes seconds to see the results of a single edit on edge devices, prohibiting interactive user experience. In this paper, inspired by quick preview features in modern rendering software, we propose Anycost GAN for interactive natural image editing. We train the Anycost GAN to support elastic resolutions and channels for faster image generation at versatile speeds. Running subsets of the full generator produce outputs that are perceptually similar to the full generator, making them a good proxy for quick preview. By using sampling-based multi-resolution training, adaptive-channel training, and a generator-conditioned discriminator, the anycost generator can be evaluated at various configurations while achieving better image quality compared to separately trained models. Furthermore, we develop new encoder training and latent code optimization techniques to encourage consistency between the different sub-generators during image projection. Anycost GAN can be executed at various cost budgets (up to 10× computation reduction) and adapt to a wide range of hardware and latency requirements. When deployed on desktop CPUs and edge devices, our model can provide perceptually similar previews at 6-12× speedup, enabling interactive image editing. The code and demo are publicly available.

1. Introduction

Generative Adversarial Networks (GANs) [17] have excelled at synthesizing diverse and realistic images from randomly sampled latent codes [42, 44, 45, 60, 6]. Furthermore, a user can transform the generated outputs (*e.g.*, add smiling to a portrait) by tweaking the latent code [64, 39, 44, 25, 68]. In real-world use cases, a user would often like to edit a natural image rather than generating random samples. To achieve this, one can project the image into the image manifold of GANs by finding a latent code that reconstructs the image,

* Part of the work done during an internship at Adobe Research.

and then modify the code to produce final outputs [82].

Despite its photorealistic results and versatile editing ability, modern deep generative models incur huge computational costs, prohibiting edge deployment. For example, the StyleGAN2 generator [45] consumes 144G MACs, 36× larger compared to ResNet-50 [27]. The expensive model often introduces a several-second delay for a single edit, leading to a sub-optimal user experience and shorter battery life when used on an edge device.

Modern 2D/3D content creation workflows, such as the preview rendering feature in Maya and Blender, as well as the playback feature in Adobe Premiere Pro, allow users to easily control the tradeoff between image quality and rendering speed. A user can turn off certain visual effects, reduce the resolution and fidelity, or use a fast method during user interaction. Once the edit is finalized, a user can use an expensive method with additional visual effects at a higher resolution. In rendering literature, this tradeoff can be easily achieved by reducing the number of sampled rays in ray/path tracing [40], or early stopping of iterative solvers in progressive radiosity [18, 13]. In this work, we aim to bring a smooth tradeoff between visual quality and interactivity to deep generative models.

We propose "Anycost" GANs for interactive image synthesis and editing. Our goal is to train a generator that can be executed at a wide range of computational costs while producing visually consistent outputs: we can first use a low-cost generator for fast, responsive previews during image editing, and then use the full-cost generator to render high-quality final outputs. We train the anycost generators to support *multi-resolution* outputs and *adaptive-channel* inference. The smaller generators are nested inside the full generator via weight-sharing. Supporting different configurations in one single generator introduces new challenges for minimax optimization, as the adversarial optimization involves too many players (different sub-generators). Vanilla GAN training methods fail catastrophically in this setting. To stabilize the training process, we propose to perform stage-wise training: first train sub-generators at multiple resolutions but with full channels, and then train sub-generators with reduced channels. To account for different sub-generators' capacities and architectures, we train a weight-sharing discriminator that is conditioned on the architectural information of the specific sub-generator.

We train Anycost GAN to support two types of channel configurations: uniform channel reduction ratio and flexible ratios per layer. The combined architecture space leads to high flexibility in terms of computation cost, containing sub-generators at $> 10\times$ computation difference. We can directly obtain low-cost generators by taking a subset of weights without any fine-tuning, which allows us to easily switch between quality and efficiency. To handle diverse hardware capacities, we use evolutionary search to automatically find the best sub-generator under different computational budgets, while achieving the best output consistency w.r.t. the full-cost generator (Figure 1b).

To better maintain consistency during the image projection process, we further propose consistency-aware encoder training and iterative optimization for image projection. We optimize the reconstruction loss, not only for the full-cost generator, but also for the sub-generators, which significantly improves the consistency during both image projection and editing steps (Figure 1c).

Our single, anycost generator can provide visually consistent outputs at various computation budgets. Compared to small generators of the same architecture or existing compression methods based on distillation, our method can provide better generation quality and higher consistency w.r.t. to the full generator. The generated outputs from generators at different costs also share high-level visual cues, for example, producing generated faces with consistent facial attributes (Table 4). Our method provides 12.2× speed-up on Xeon CPU and 8.5× speed up on Jetson Nano GPU for faster preview. Combined with consistency-aware image projection, our anycost generator maintains consistency after various editing operations (Figure 8) and offers an efficient and interactive image editing experience.

2. Related Work

Generative Adversarial Networks. GANs [17] have enabled photorealistic synthesis for many vision and graphics applications [47, 11, 38, 83, 62, 32, 15]. Some recent examples include high-resolution face synthesis [44, 45], ImageNet-scale class-conditional generation [6], and semantic photo synthesis [72, 61]. As image quality and model capacity have increased [17, 64, 77, 42, 76, 45, 41], so have computational costs and inference time. For example, it takes several seconds for a single forward pass of StyleGAN2 [45] on a desktop CPU or a tablet. While most research demos are running on the latest GPUs, deploying them on edge devices and laptops efficiently is critical for interactive editing applications. In this work, we tackle this problem by learning hardware-adaptive generators that work across a wide range of devices and latency constraints.

Model acceleration and dynamic networks. Efficient deep networks have enabled fast inference and reduced model sizes with a focus on image classifiers [24, 23, 80, 71, 22]. Commonly used approaches include pruning [29, 52, 58, 24, 23, 73], quantization [23, 80], knowledge distillation [31, 59, 10], efficient architecture design [12, 37, 35, 67], and autoML-based methods [28, 34, 71, 84, 55, 9, 20, 51]. Recently, several works adopt the above ideas to compress generative networks with a focus on image-conditional GANs [69, 49, 74, 70, 14], such as pix2pix [38] and CycleGAN [83]. The work most related to ours is Aguinaldo et al. [3], which adopts knowledge distillation to compress DC-

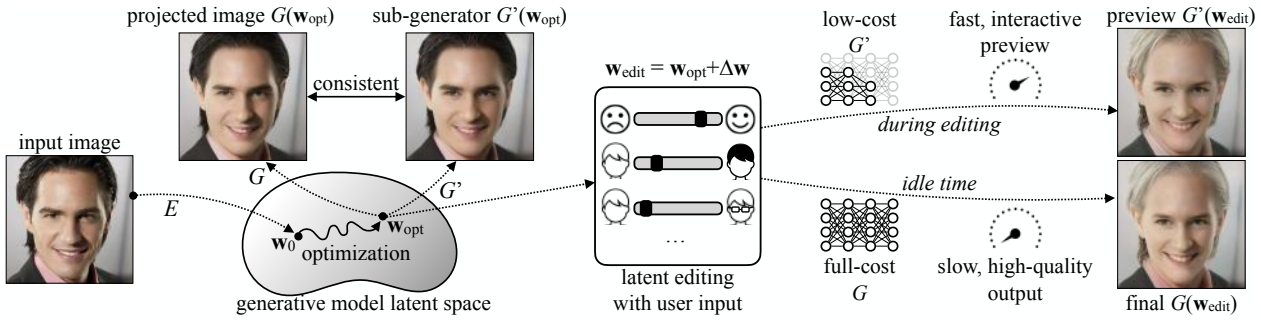


Figure 2: Anycost GAN for image synthesis and editing. Given an input image, we project it into the latent space with encoder E and backward optimization. We can modify the latent code with user input to edit the image. During editing, a sub-generator of small cost is used for fast and interactive preview; during idle time, the full cost generator renders the final, high-quality output. The outputs from the full and sub-generators are visually consistent during projection and editing.

GAN [64] on low-res images of MNIST, CIFAR, and CelebA datasets. A concurrent work [33] also explores channel reduction of unconditional GANs on low-resolution datasets. Our experiments show that knowledge distillation alone is insufficient to transfer the knowledge of a GAN on large-scale datasets (Figure 5). Compared to prior work, our method works well for large-scale, high-resolution unconditional GANs. More importantly, our method can produce output images with dynamic resolution and fidelity, not possible by the prior work [3]. We also include techniques to leverage the anycost generator in real image editing scenarios.

Image editing via GANs. There exist two major ways of using GANs for image editing: (1) conditional GANs [38, 83, 56, 57, 48], which learn to directly translate an input image into a target domain, and (2) image projection [82, 7], where the algorithm first projects a real image into the latent space of an unconditional GAN, modifies the latent code to achieve an edit, and synthesizes a new image accordingly. Recently, interest in projection-based editing has been revived due to the increasing quality of unconditional GANs. Several methods have been proposed, including choosing better or multiple layers to project and edit [1, 2, 19], fine-tuning network weights for each image [5], modeling image corruption and transformations [4, 36], and discovering meaningful latent directions [68, 16, 39, 25]. In this work, we aim to learn efficient generators for downstream image projection and editing, which offers unique challenges. Namely, we wish the projected latent code from a sub-generator provide a faithful "preview" of the full model. In addition, user edits with the sub-generator should also make the corresponding changes to the full model. We tackle these challenges using our new consistency-aware encoder training and optimization method.

3. Anycost GANs

We introduce the problem setting and potential challenges in Section 3.1. We then describe our training method for learning multi-resolution adaptive-channel generators in Section 3.2. In Section 3.3, we describe our consistency-aware encoder training and latent code optimization, designed for image projection and editing.

3.1. Problem Setup

An unconditional generator G learns to synthesize an image $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$ given a random noise $\mathbf{z} \in \mathcal{Z}$, where $\mathcal{Z}$ is a low-dimensional latent space. In this work, we study StyleGAN2 [45], which first learns a non-linear mapping $f : \mathcal{Z} \to \mathcal{W}+$ that produces $\mathbf{w} = f(\mathbf{z}), \mathbf{w} \in \mathcal{R}^{18 \times 512}$, and then generates the image from $\mathcal{W}+$ space: $\mathbf{x} = G(\mathbf{w})$ [1].

Anycost GAN enables a subsection of the generator G' to execute independently and produce a similar output $\mathbf{x}' = G'(\mathbf{w})$ to the full generator $G(\mathbf{w})$. We can then use $G'(\mathbf{w})$ for fast preview during interactive editing, and full $G(\mathbf{w})$ to render final high-quality outputs. The model can be deployed on diverse hardware (*e.g.*, GPUs, CPUs, smartphones), and users can choose between different preview qualities.

A natural choice for enabling diverse inference cost is to use different image resolutions, as done in StackGAN [78] and ProGAN [42]. However, reducing resolution from 1024×1024 to 256×256 only reduces the computation by 1.7×, despite 16× fewer pixels to synthesize*. Therefore, we need additional dimensions to further reduce the cost.

3.2. Learning Anycost Generators

We propose to learn an anycost generator to produce proxy outputs at diverse *resolutions* and *channel capacities*. The detailed architecture is shown in Figure 3.

Multi-resolution training. For elastic resolutions, the architectures of the ProGAN [42] and StyleGAN family [44, 45] already produce lower-res intermediate outputs, although the outputs do not look natural (Figure 4a). We can enable lower-resolution outputs by enforcing multi-scale

*1024-resolution StyleGAN2 has 144G MACs, while 256-resolution model has 85G MACs.

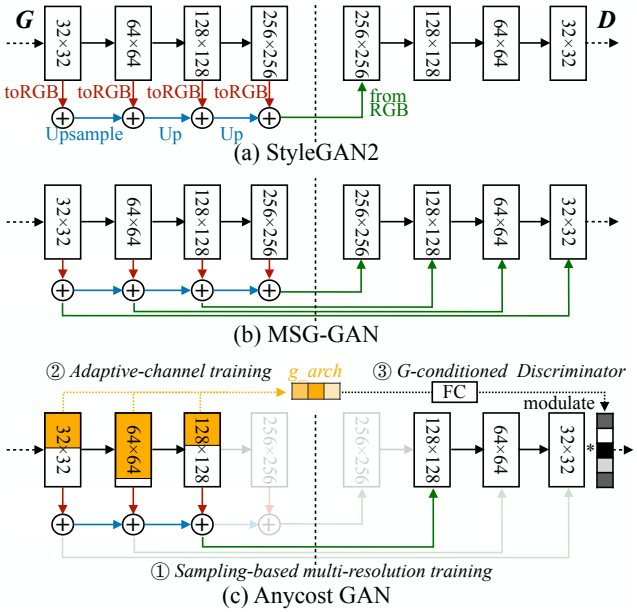


Figure 3: Anycost GAN synthesizes realistic images at a wide range of resolutions and model capacities via (1) sampling-based multi-res training, (2) adaptive-channel training, and (3) generator-conditioned discriminator.

training objectives (Figure 4b). Our generator gradually produces higher resolution outputs after each block g^k:

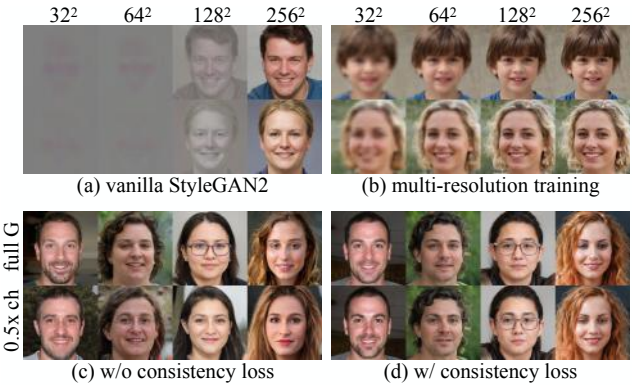
$$\tilde{\mathbf{x}} = G(\mathbf{w}) = g^K \circ g^{K-1} \circ ... \circ g^k \circ ... \circ g^2 \circ g^1(\mathbf{w}), \quad (1)$$

where K is the total number of network blocks. We use the intermediate low-res outputs for preview, and denote:

$$\tilde{\mathbf{x}}^k = G^k(\mathbf{w}) = g^k \circ g^{k-1} \circ ... \circ g^2 \circ g^1(\mathbf{w}), k \leq K, \quad (2)$$

resulting a series of outputs in increasing resolutions $[\tilde{\mathbf{x}}^1, ..., \tilde{\mathbf{x}}^K]$.

Existing work MSG-GAN [41] trains the generator to support different resolutions by using a discriminator taking images of all resolutions at the same time (Figure 3b). However, such an all-resolution training mechanism leads to lower quality results on large-scale datasets like FFHQ [45] (as measured by Fréchet Inception Distance, FID [30]), compared to training single-resolution models separately (Table 1). To overcome the image fidelity degradation when supporting multi-resolutions, we propose a *sampling-based* training objective, where a *single* resolution is sampled and trained at each iteration, both for the generator G and the discriminator D. As shown in Figure 3c, when sampling a lower resolution (*e.g.*, 128×128), the translucent parts are not executed. We use the intermediate output of G for a lower resolution. It is passed through a fromRGB convolution "readoff" layer to increase channels, and then fed to an intermediate layer of D. Our multi-resolution training



(a) vanilla StyleGAN2 (b) multi-resolution training

(c) w/o consistency loss (d) w/ consistency loss

Figure 4: (a) Intermediate layers do not output realistic images for StyleGAN2 baseline. (b) Multi-resolution training in our model enables photorealistic lower-resolution outputs at intermediate layers. (c) Without consistency loss, running the model at full and half-channel produces inconsistent images. (d) Adding consistency loss maintains output consistency (similar images per column).

objective is formulated as:

$$\mathcal{L}_{\text{multi-res}} = \mathbb{E}_{\mathbf{x},k}[\log D(\mathbf{x}^k)] + \mathbb{E}_{\mathbf{w},k}[\log(1 - D(G^k(\mathbf{w})))]. \quad (3)$$

We observe that the *sampling-based* training leads to better convergence: in fact, our multi-resolution model gives better FID at all resolutions, compared to single-resolution models trained at corresponding resolutions, as shown in Table 1.

Adaptive-channel training. To further enable our generator to run at different costs, we train the generator to support variable channels. For adaptive-channel training, we allow different channel number multipliers for each layer (either a *uniform* ratio, used across all layers or *flexible* ratios for each layer). For each training iteration, we randomly sample a channel multiplier configuration and update the corresponding subset of weights (yellow part in Figure 3c). We hope that the most "important" channels are preserved during sampling to minimize any degradation. To this end, we initialize our model using the multi-resolution generator from the previous stage and sort the channels of convolutional layers according to the magnitude of kernels, from highest to lowest. We always sample the most important αc ones according to the initial sorting, where $\alpha \in [0.25, 0.5, 0.75, 1]$ and c is the number of channels in the layer. The adaptive-channel training objective is written as follows:

$$\mathcal{L}_{\text{ada-ch}} = \mathbb{E}_{\mathbf{x},k}[\log D(\mathbf{x}^k)] + \mathbb{E}_{\mathbf{w},k,\mathbb{C}}[\log(1 - D(G^k_{\mathbb{C}}(\mathbf{w})))], \quad (4)$$

where $\mathbb{C}$ means the channel configurations for each layer.

With such a learning objective, all the sub-networks with fewer channels can generate reasonable images. However, these images may be visually different compared to the full

network, failing to provide an accurate "preview" (Figure 4c). To keep the output consistent across different sub-networks, we add the following consistency loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{ada-ch}} + \mathbb{E}_{\mathbf{w},k,\mathbb{C}}[\ell(G^k_{\mathbb{C}}(\mathbf{w}), G(\mathbf{w}))], \quad (5)$$

where ℓ is a pre-defined distance metric. We use a combination of MSE loss and LPIPS loss [79], which empirically gives the most visually consistent results (Figure 4d). Note that all the sub-networks and the full network share weights across different channels and are jointly trained.

Generator-conditioned discriminator. Unlike conventional GAN training, our system trains many sub-generators of different channels and resolutions at the same time. We find that one single discriminator cannot provide good supervision for all sub-generators of different channel configurations, resulting in an inferior image fidelity/FID (Table 2). To handle the challenge, we design the discriminator to be *conditioned on the generator architecture*.

A straightforward method for generator-conditioning is to correspondingly shrink the channels of the discriminator with the generator. However, we find that such practice often sacrifices the quality of smaller channel settings, due to its reduced capacity of the discriminator ("reduced ch" in Table 2). Also, such a technique only supports uniform channel ratios. Instead, we take a learning-based approach to implement the conditioning. As shown in Figure 3c, we first encode the channel configuration in g_arch using a one-hot encoding (for each layer, we can choose one of the four ratios, forming a one-hot vector of length 4; we concatenate the vectors from all layers to form g_arch), which is passed through a fully connected layer to form the per-channel modulation. The feature map is modulated using the conditioned weight and bias before passing to the next layer. For real images, we randomly draw a g_arch vector. To stabilize training, we only apply the G-conditioned modulation units to the last two blocks of the discriminator.

Searching under different budgets. By training the generator to support flexible ratios for each layer, we support an exponential number of sub-generator architectures. At a given computation budget, selecting the proper sub-generator configuration is important for keeping generation quality and consistency. We use an evolutionary search algorithm [65, 8, 20] to find an effective sub-generator architecture under diverse resource budgets (*e.g.*, computation, latency). Given a certain budget, our evolutionary search minimizes the difference between the desired sub-generator and the full generator's outputs, measured by a perceptual loss. The details can be found in Section B.2.

3.3. Image Projection with Anycost Generators

To edit an existing image $\mathbf{x}$, we need to first project the image into the latent space of a generator [82] by solving $\mathbf{w}^* = \arg\min_{\mathbf{w}} \ell(G(\mathbf{w}), \mathbf{x})$, where we use a combination of LPIPS [79] and MSE loss for ℓ. We follow [1, 2] to project the image into the extended $\mathcal{W}+$ space, rather than the $\mathcal{Z}$ space due to its better expressiveness. We follow the two-step approach, as introduced in iGAN [82]: encoder-based initialization followed by gradient-based latent code optimization.

Consistency-aware image projection. *Encoder-based* projection directly trains an encoder E for projection by optimizing $E^* = \arg\min_E \mathbb{E}_{\mathbf{x}} \ell(G(E(\mathbf{x})), \mathbf{x})$ over many training images. For a specific image sample, we can further improve the results with *optimization-based* projection by solving $\mathbf{w}^* = \arg\min_{\mathbf{w}} \ell(G(\mathbf{w}), \mathbf{x})$ with iterative gradient descent. While our generator can produce consistent results across sub-generators for a *randomly* sampled latent code, the predicted/optimized latent codes $E(\mathbf{x})$ may not follow the prior distribution. As a result, the sub-generators may not produce consistent results on some optimized latent codes. Therefore, we modify the objects to produce a latent code that works for both the full generators as well as randomly sampled sub-generators as follows:

$$E^* = \arg\min_E \mathbb{E}_{\mathbf{x}}[\ell(G(E(\mathbf{x})), \mathbf{x}) + \alpha \mathbb{E}_{k,\mathbb{C}} \ell(G^k_{\mathbb{C}}(E(\mathbf{x})), \mathbf{x})] \quad (6)$$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} [\ell(G(\mathbf{w}), \mathbf{x}) + \alpha \mathbb{E}_{k,\mathbb{C}} \ell(G^k_{\mathbb{C}}(\mathbf{w}, \mathbf{x})] \quad (7)$$

We set hyper-parameter $\alpha = 1$ in our experiments.

Image editing with anycost generators. After projection, we can perform image editing by simply changing the latent code and synthesize a new one using $G(\mathbf{w} + \Delta\mathbf{w})$, where $\Delta\mathbf{w}$ is a vector that encodes a certain change. Several methods [68, 25] have been proposed to discover such latent directions that control certain aspects of the input (*e.g.*, smiling/non-smiling for faces, color, and shape for cars). To produce a preview with low latency, we can run $G^k_{\mathbb{C}}(\mathbf{w} + \Delta\mathbf{w})$. In experiments, we observe that as long as the initial projection is consistent across the full and sub-generators, the edited results are visually similar.

4. Experiments

4.1. Setup

We conduct experiments on both FFHQ [44] (resolution 1024) and LSUN car dataset [75] (resolution 512) due to the large scale and high resolution. Our generators are based on StyleGAN2 [45] (config-F). We train our models to support four resolutions from the highest (*e.g.*, 1024, 512, 256, 128 for FFHQ), since lower resolutions are too blurry for a high-quality preview. For dynamic channels, we support multipliers ranging [0.25, 0.5, 0.75, 1] w.r.t. the original channel numbers. For fast ablation studies, we train models

Table 1: FID-70k on FFHQ of different multi-resolution training techniques. Our sampling-based technique can train one model that produces multiple resolution outputs with higher image quality (measured by FID [30]) compared to single resolution training. The models are trained with half channels (Config-E) for faster ablation.

Resolution	1024	512	256	128	64	32
Single-resolution	3.25	4.17	3.76	4.04	3.32	2.41
MSG-GAN [41]	-	-	4.79	6.34	2.7	3.04
Ours (low res)	-	-	3.49	**3.26**	**2.52**	**2.18**
Ours (high res)	**2.99**	**3.08**	**3.35**	3.98	-	-

Table 2: FIDs on FFHQ at different resolutions and channels. All the settings except "vanilla" only train one generator and evaluate it at multiple configurations. We mark a better FID green and a worse FID red compared to multi-G baseline. Conditioned discriminator ("conditioned") provides the best FID over different channel widths and resolutions. The model is based on Config-E for faster ablation.

FID-70k↓	resolution 256				resolution 128			
	1×	0.75×	0.5×	0.25×	1.0×	0.75×	0.5×	0.25×
vanilla	3.80	4.64	6.20	10.39	4.04	4.99	5.78	11.15
same D	3.63	3.91	5.41	14.01	7.25	6.81	5.92	7.57
reduced ch	3.67	4.35	5.82	10.62	3.09	3.65	4.74	8.82
conditioned (Ours)	3.73	3.86	4.64	8.06	3.30	3.28	3.69	5.55

on 256×256 images using config-E (half channels). More experimental details are provided in Section B.

For adaptive channel support, we consider two settings: (1) training with uniform channel ratios (*i.e.*, using the same channel reduction ratios for all layers); (2) training with flexible channel ratios for each layer. For the latter setting, after GAN training, we leverage evolutionary search to find the best channel & resolution configurations under certain computation budgets. The details of the evolutionary search can be found in Section B.2.

4.2. Ablation studies

We first show how we improve the image quality while supporting different resolutions and channels with ablation studies. The results are evaluated on FFHQ dataset.

Multi-resolution training. We compare the results of sampling-based multi-resolution training against single-resolution training and MSG-GAN [41] in Table 1. With our technique, we can train one generator that generates multi-resolution outputs at a better quality (lower FID) for both high-resolution (1024) and lower-resolution (256) settings, even compared to specialized single-resolution models. Compared to MSG-GAN [41], which trains all resolutions at each iteration, our method consistently gains better FIDs. We hypothesize that feeding images of all resolutions to the discriminator poses a stricter requirement for the generator (all the outputs have to be realistic to fool the discriminator), breaking the balance between the generator and the discriminator.

G-conditioned discriminators for dynamic channels. Using one fixed discriminator is not enough to handle a set of generators at different channel capacities and resolutions. We use a simpler setting for ablation by only supporting four uniform channel reduction ratios, *i.e.*, using the same channel multiplier for all layers. As shown in Table 2, using the same discriminator for all sub-generators (denoted as "same D") cannot consistently match the FIDs compared to single generators trained for a specific resolution and channel width (denoted as "vanilla"). It also leads to an unstable FID distribution (*e.g.*, for resolution 128, 1.0× channel gives worse FID compared to 0.5×, despite increased computational resources), since a single discriminator can only provide suitable supervisions for a small subset of sub-generators. To improve the performance of each generator, we implement the discriminator to be *conditioned* on the generator architecture. A straight-forward method is to also reduce the channels of the discriminator using the same ratio as the generator (denoted as "reduced ch" in Table 2). This improves the FIDs under some conditions and makes the FIDs monotonic as a function of computation (wider sub-generators give better FID). However, the narrow generators (*e.g.*, 0.25×) have degraded FIDs due to limited discriminator capacity. This also does not work for non-uniform channel ratio settings, where each layer can use a different channel ratio, leading to exponential combinations. Instead, our conditioned discriminator (denoted as "conditioned") uses a *learned* modulation according to the generator architecture (represented as an architecture vector), without reducing the discriminator capacity. For different channels and resolutions, it provides consistently better FIDs compared to the multi-generator baseline, and also improves over the reduced-channel method.

Outperforming compression baselines. One baseline for fast generation preview is to train a small model that mimics the full generator and use the small model for interactive image editing. However, this method does not allow flexible model capacity to fit diverse hardware platforms and latency requirements. The small models also have inferior generation quality and consistency w.r.t. to the full generator, compared to our anycost generator.

We compare the FID-computation trade-off with compression-based baselines in Figure 5a. "Vanilla" means single models trained from scratch. We compare with [3], which uses a distillation-based method for *unconditional* GAN compression ("Distill"), and also adapt a general CNN

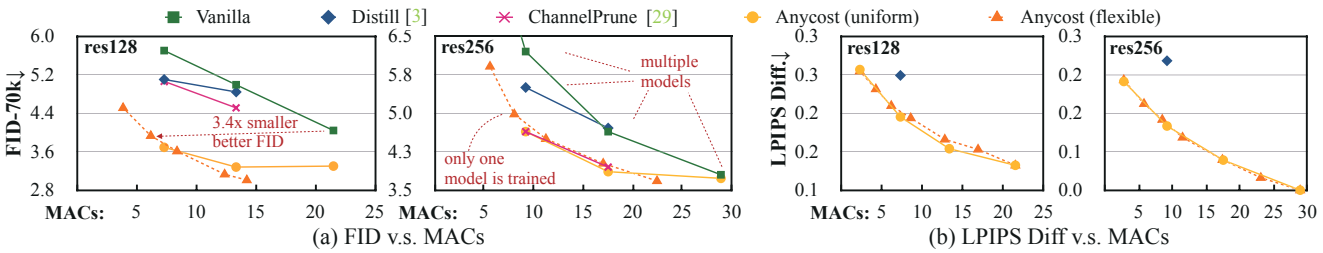


Figure 5: Anycost GAN outperforms existing compression baselines [3, 29] at various computation budgets, despite only training a single, flexible generator across computation budgets.

compression method [29] to GAN ("ChannelPrune"). Since the FID cannot be computed across different resolutions, we provide the results for resolution 256 and 128 separately (we can search over all resolutions and channels when optimizing for consistency, like in Figure 7). For anycost GAN, we provide the results of two settings: uniform channel ratio (denoted as "Anycost (uniform)") and flexible channel ratios (denoted as "Anycost (flexible)"). Anycost GAN outperforms existing compression methods at a *wide range* of computation budgets, despite using only a *single* generator, achieving lower FID/difference at the same computation. The "uniform" and "evolve" settings achieve a similar trade-off curve. However, with evolutionary search, we can support more fine-grained computation budgets to fit more deployment scenarios.

Apart from generation quality, our sub-generators also provide more consistent outputs w.r.t. to the full generator (Figure 5b). We measure the LPIPS difference [79] between the generated images from sub-networks and the full network. Compared to distillation-based training, our model reduces the LPIPS difference by half. For models trained from scratch, we cannot report LPIPS consistency, as the models are independently trained.

4.3. Anycost Image Synthesis

In this section, we provide the results of Anycost GAN trained on high-resolution images (1024×1024 for FFHQ [45] and 512×384 for LSUN Car [75]). The models are trained with Config-F [45] for high-quality synthesis.

Qualitative results. In Figure 6, we show several samples from our anycost generators (uniform channel setting). Despite only training a single generator, Anycost GAN maintains output consistency across different resolutions and different channel widths compared to the full generator's output, providing a fast preview when a user is exploring the latent space or applying editing operations. The samples are generated with truncation rate $\psi = 0.5$.

We also provide the results of anycost generator (flexible channels +evolutionary search setting) in Figure 1 and Figure 7. With evolutionary search, we can provide sub-generators at fine-grained computation budgets at 2-10× reduction. The sub-generators share high consistency for the projected and edited images, paving a way for fast editing preview.

Table 3: Anycost GANs achieves similar or better FIDs/path lengths at various channel widths compared to StyleGAN2, despite training a single, flexible generator.

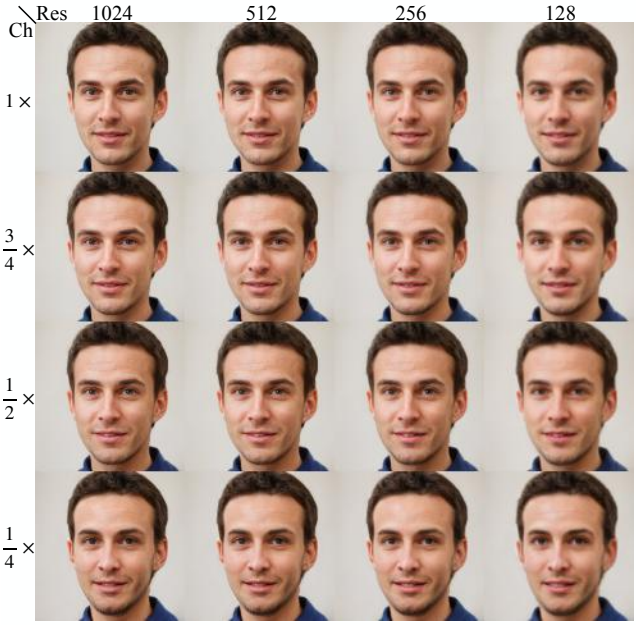
		FID-50k↓				Path length↓	
Channels:		1.0×	0.75×	0.5×	0.25×	1.0×	0.5×
FFHQ 1024	StyleGAN2	2.84	-	3.31	-	145.0	124.5
	Anycost	2.77	3.05	3.28	5.01	144.2	147.2
Car 512	StyleGAN2	2.32	-	3.19	-	415.5	471.2
	Anycost	2.38	2.46	2.61	3.69	380.1	430.0

Table 4: The generated outputs of the full and sub-generators have high *attribute consistency*, validated by match rates. The chosen attributes have high inconsistency for two separately trained generators ("Separate").

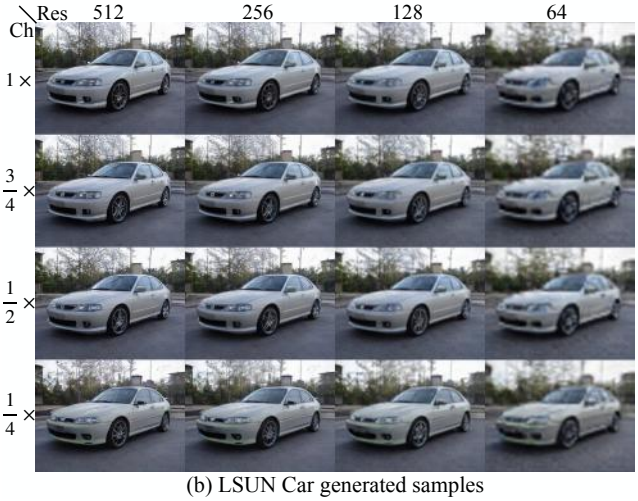
	Smiling	BlackHair	Eyeglass	StraightHair	Earrings
Separate	55.8%	62.7%	83.4%	55.8%	50.6%
0.5× ch	98.1%	97.0%	99.9%	95.9%	94.1%
0.25× ch	96.9%	95.5%	99.8%	93.6%	88.4%

Quantitative results. We also provide the quantitative results of the high-resolution images. The high-resolution FIDs and path lengths [44] are shown in Table 3. We only provide the results of uniform channel setting, since the flexible setting shares a similar FID *vs.* computation trade-off (Figure 5). Compared to StyleGAN2, Anycost GANs achieves similar or better FIDs and path lengths at various channel widths, despite only one generator is trained. Anycost GANs enjoy better flexibility for inference and also better synthesis quality compared to the baselines.

Attribute consistency. Our low-cost results not only preserve the visual consistency (LPIPS difference), but also de-



(a) FFHQ generated samples



(b) LSUN Car generated samples

Figure 6: **Anycost GAN (uniform)** maintains output consistency across different resolutions and channels widths. Zoom in for better view. For faces (a), our method preserves the prominent facial structure such as age, hair style, and pose. Some small details such as wrinkles are omitted. For cars (b), our method succeeds to keep the color, shape, and pose. Some small details are omitted such as license plates.

pict similar visual attributes as the full results do. We check the binary attributes (*e.g.*, is smiling, is wearing eyeglasses) of images generated by sub-generators ($0.5\times$ and $0.25\times$ channels) and the full generator. Following [44], we trained the attribute classifier to predict 40 attributes on CelebA-HQ for evaluation. We compare the prediction results on 10,000 randomly generated images and report the match rates in Table 4. For all the categories, we can achieve $> 95\%$ consistency with $0.5\times$ channel (except for "Earrings", which

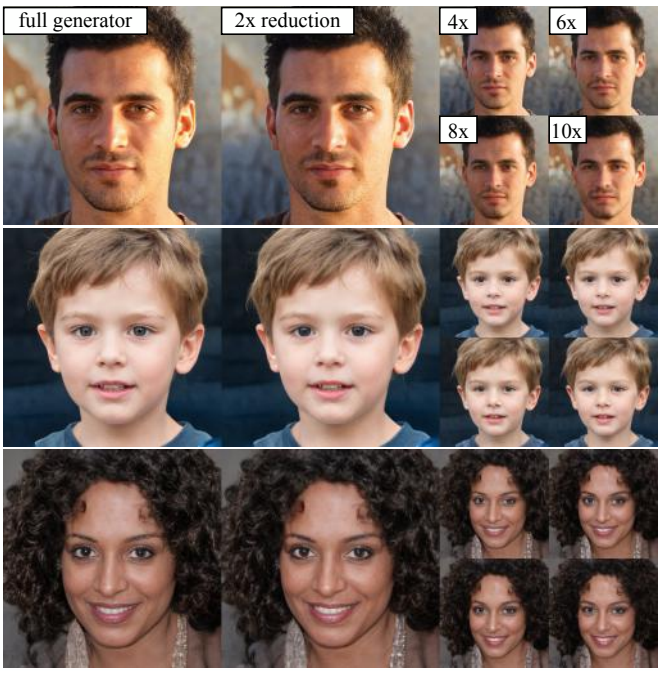


Figure 7: **Anycost GAN (flexible)** maintains output consistency across *fine-grained* computation reductions. Zoom in for better view. To save space, we display some images in a smaller size. Check Figure 15 for a larger view.

Table 5: Inference FPS & speed up rates on different devices.

FLOPs reduction	1×	2×	4×	6×	8×	10×
Xeon CPU FPS	0.63	1.78	5.94	6.24	7.48	7.35
speed up rate	1×	2.8×	9.5×	10.0×	11.9×	11.7×
Nano GPU FPS	0.65	1.17	2.77	3.59	4.07	4.1
speed up rate	1×	1.8×	4.2×	5.5×	6.2×	6.3×

is challenging due to its small size), which is higher than the accuracy of attribute classifiers on CelebA [21], indicating that our model preserves high-level visual attributes across full generators and sub-generators. The attributes in Table 4 are chosen due to their high inconsistency between two separately trained generators ("Separate").

Latency reduction. We measure the latency speed-up on both desktop CPU (Intel Xeon Silver) and mobile GPU (NVIDIA Jetson Nano) in Table 5. Anycost generators can generate consistent previews at $11.9\times$ walltime speed-up on Xeon CPU and $8.5\times$ speed-up on Jetson Nano. Interestingly, the model achieves a super-linear speed-up on Intel CPU. We hypothesize that the activation and weight sizes of the full-cost generator exceed the cache on CPU (16.5MB), leading to increased cache miss rate and worse inference efficiency.

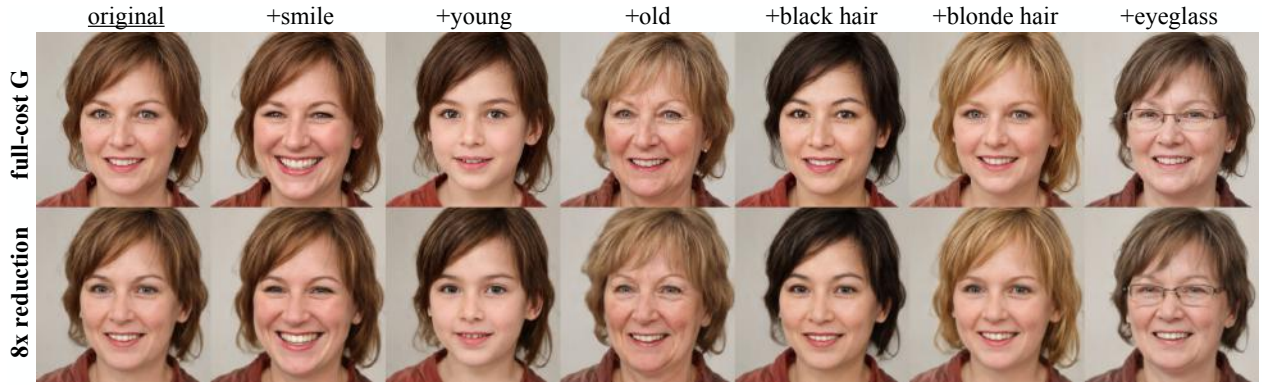


Figure 8: Anycost GAN maintains consistency after image editing, providing a quick preview at 8× computation reduction.

Table 6: Consistency-aware encoder gives accurate projection results for both the full generator and sub-generators. It largely reduces the gap between them. Here $\mathbf{w} = E(x)$.

	$\ell(G(\mathbf{w}), \mathbf{x})$		$\ell(G'(\mathbf{w}), \mathbf{x})$		$\ell(G'(\mathbf{w}), G(\mathbf{w}))$	
	LPIPS	MSE	LPIPS	MSE	LPIPS	MSE
ALAE [63]	0.32	0.15	-	-	-	-
IDInvert [81]	0.22	0.06	-	-	-	-
pSp [66]	0.19	0.04	-	-	-	-
Ours (fullG-only)	**0.13**	**0.03**	0.17	0.04	0.07	0.012
Ours (full+subG)	**0.13**	**0.03**	**0.14**	**0.03**	**0.02**	**0.003**

4.4. Anycost Image Projection and Editing

Encoder training. We compare the encoder only optimized for the full model and the one optimized for both the full network and the sub-networks. The results on CelebA test set are in Table 6. For generality, we used a simple encoder architecture: we used the ResNet-50 [27] backbone architecture and a linear layer to regress the latent code in $\mathcal{W}+$ space. We train the encoder both on real images (FFHQ + CelebA train set following [66]) and generated images for 200 epochs. We apply random horizontal flip, random color jittering, and random grayscale as augmentation [26]. To compare with existing literature, we measure the LPIPS loss using AlexNet backbone instead of VGG. Apart from the reconstruction loss for the full generator, we also measure the average reconstruction performance for all the sub-generator architectures found by our evolutionary algorithm. Our generator has better reconstruction performance compared to an advanced encoder design [66] that uses a Feature Pyramid Network [53] structure.

Optimization-based projection. For optimization-based projection, we used L-BFGS solver [54] with 100 iterations. We find that L-BFGS converges faster than Adam [46]. We use the encoder's prediction as the starting point to represent a more realistic use case, which also results in better convergence compared to starting from average latent $\mathbf{w}$ [45]. Interestingly, we find that a lower optimization loss results in a better reconstruction, but the latent code may not be suitable for latent space-based editing (see Section D.2). Therefore, we do not benchmark the quantitative results here. The qualitative results are shown in Figure 1.

Image editing with anycost generators. We show that anycost generators remain consistent after image editing. We compute several editing directions on FFHQ dataset in the $\mathcal{W}$ space following [68]. We compare the outputs of the full generators and 8×-smaller computation sub-generators after editing smiling, age, eye-glasses, and hair color in the latent space. The results are shown in Figure 8 and Figure 1 . Anycost generator gives consistent outputs under various attribute editing. We show more projection and editing examples in the supplementary materials.

5. Discussion

In this paper, we have proposed Anycost GAN, a scalable training method for learning unconditional generators that can adapt to diverse hardware and user latency requirements. We have demonstrated its application in image projection and editing. Several limitations still exist for our method. First, the control over the channel numbers might be difficult for users who are new to neural networks. In the future, we plan to provide more intuitive controls such as synthesizing similar color, texture, illumination, or shape as the original model does. Second, our current model aims to approximate every single output pixel equally well. In practice, certain objects (e.g., face) might be more important than others (e.g., background). We would like to learn models that can support spatially-varying trade-off between fidelity and latency.

Acknowledgment. We thank Taesung Park and Zhixin Shu for the helpful discussion. Part of the work is supported under NSF CAREER Award #1943349. We thank MIT-IBM Watson AI Lab for the support.

References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 3, 5

[2] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan++: How to edit the embedded images? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3, 5

[3] Angeline Aguinaldo, Ping-Yeh Chiang, Alex Gain, Ameya Patil, Kolten Pearson, and Soheil Feizi. Compressing gans using knowledge distillation. *arXiv preprint arXiv:1902.00159*, 2019. 2, 3, 6, 7

[4] Rushil Anirudh, Jayaraman J Thiagarajan, Bhavya Kailkhura, and Peer-Timo Bremer. Mimicgan: Robust projection onto image manifolds with corruption mimicking. *International Journal of Computer Vision*, pages 1–19, 2020. 3

[5] David Bau, Hendrik Strobelt, William Peebles, Jonas Wulff, Bolei Zhou, Jun-Yan Zhu, and Antonio Torralba. Semantic photo manipulation with a generative image prior. *ACM SIGGRAPH*, 38(4):1–11, 2019. 3

[6] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations (ICLR)*, 2019. 1, 2

[7] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2017. 3

[8] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once for All: Train One Network and Specialize it for Efficient Deployment. In *International Conference on Learning Representations (ICLR)*, 2020. 5

[9] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations (ICLR)*, 2019. 2

[10] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 2

[11] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[12] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, pages 1251–1258, 2017. 2

[13] Michael F Cohen, Shenchang Eric Chen, John R Wallace, and Donald P Greenberg. A progressive refinement approach to fast radiosity image generation. In *ACM SIGGRAPH*, 1988. 2

[14] Yonggan Fu, Wuyang Chen, Haotao Wang, Haoran Li, Yingyan Lin, and Zhangyang Wang. Autogan-distiller: Searching to compress generative adversarial networks. *arXiv preprint arXiv:2006.08198*, 2020. 2

[15] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(1):2096–2030, 2016. 2

[16] Lore Goetschalckx, Alex Andonian, Aude Oliva, and Phillip Isola. Ganalyze: Toward visual definitions of cognitive image properties. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 3

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014. 1, 2

[18] Cindy M Goral, Kenneth E Torrance, Donald P Greenberg, and Bennett Battaile. Modeling the interaction of light between diffuse surfaces. *ACM SIGGRAPH*, 18(3):213–222, 1984. 2

[19] Jinjin Gu, Yujun Shen, and Bolei Zhou. Image processing using multi-code gan prior. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[20] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, pages 544–560. Springer, 2020. 2, 5

[21] Hu Han, Anil K Jain, Fang Wang, Shiguang Shan, and Xilin Chen. Heterogeneous face attribute estimation: A deep multi-task learning approach. *PAMI*, 2017. 8

[22] Song Han, Han Cai, Ligeng Zhu, Ji Lin, Kuan Wang, Zhijian Liu, and Yujun Lin. Design automation for efficient deep learning computing. *arXiv preprint arXiv:1904.10616*, 2019. 2

[23] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations (ICLR)*, 2015. 2

[24] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 2

[25] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. In *Advances in Neural Information Processing Systems*, 2020. 1, 3, 5

[26] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pages 9729–9738, 2020. 9

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 9

[28] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *European Conference on Computer Vision (ECCV)*, 2018. 2

[29] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 2, 7

[30] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems*, 2017. 4, 6, 13

[31] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NeurIPS Workshop*, 2015. 2

[32] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International Conference on Machine Learning (ICML)*, 2018. 2

[33] Liang Hou, Zehuan Yuan, Lei Huang, Huawei Shen, Xueqi Cheng, and Changhu Wang. Slimmable generative adversarial networks. 2021. 3

[34] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. *arXiv preprint arXiv:1905.02244*, 2019. 2

[35] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2

[36] Minyoung Huh, Richard Zhang, Jun-Yan Zhu, Sylvain Paris, and Aaron Hertzmann. Transforming and projecting images into class-conditional generative networks. In *European Conference on Computer Vision (ECCV)*, 2020. 3

[37] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 2

[38] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 3

[39] Ali Jahanian, Lucy Chai, and Phillip Isola. On the"steerability" of generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2020. 1, 3

[40] James T Kajiya. The rendering equation. In *ACM SIGGRAPH*, pages 143–150, 1986. 2

[41] Animesh Karnewar and Oliver Wang. Msg-gan: Multi-scale gradients for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 4, 6

[42] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations (ICLR)*, 2018. 1, 2, 3, 15

[43] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *NIPS*, 33, 2020. 13

[44] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 3, 5, 7, 8

[45] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 3, 4, 5, 7, 9, 13, 15

[46] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 9

[47] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[48] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Kumar Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representation. In *European Conference on Computer Vision (ECCV)*, 2018. 3

[49] Muyang Li, Ji Lin, Yaoyao Ding, Zhijian Liu, Jun-Yan Zhu, and Song Han. Gan compression: Efficient architectures for interactive conditional gans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[50] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *Advances in Neural Information Processing Systems*, pages 385–395, 2017. 13

[51] Ji Lin, Wei-Ming Chen, Yujun Lin, Chuang Gan, and Song Han. Mcunet: Tiny deep learning on iot devices. *Advances in Neural Information Processing Systems*, 2020. 2

[52] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 2

[53] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125, 2017. 9

[54] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989. 9

[55] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations (ICLR)*, 2019. 2

[56] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, 2017. 3

[57] Ming-Yu Liu, Xun Huang, Jiahui Yu, Ting-Chun Wang, and Arun Mallya. Generative adversarial networks for image and video synthesis: Algorithms and applications. *arXiv preprint arXiv:2008.02793*, 2020. 3

[58] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[59] Ping Luo, Zhenyao Zhu, Ziwei Liu, Xiaogang Wang, and Xiaoou Tang. Face model compression by distilling knowledge from neurons. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2016. 2

[60] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018. 1

[61] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[62] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[63] Stanislav Pidhorskyi, Donald A Adjeroh, and Gianfranco Doretto. Adversarial latent autoencoders. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 9

[64] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2016. 1, 2, 3

[65] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI*, 2019. 5

[66] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. *arXiv preprint arXiv:2008.00951*, 2020. 9

[67] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018. 2

[68] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 3, 5, 9, 14

[69] Han Shu, Yunhe Wang, Xu Jia, Kai Han, Hanting Chen, Chunjing Xu, Qi Tian, and Chang Xu. Co-evolutionary compression for unpaired image translation. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 2

[70] Haotao Wang, Shupeng Gui, Haichuan Yang, Ji Liu, and Zhangyang Wang. Gan slimming: All-in-one gan compression by a unified optimization framework. In *European Conference on Computer Vision (ECCV)*. Springer, 2020. 2

[71] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *CVPR*, 2019. 2

[72] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[73] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 2

[74] Chong Yu and Jeff Pool. Self-supervised gan compression. *arXiv preprint arXiv:2007.01491*, 2020. 2

[75] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 5, 7, 13, 15

[76] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International Conference on Machine Learning (ICML)*, 2019. 2

[77] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[78] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 3

[79] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5, 7

[80] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. In *ICLR*, 2017. 2

[81] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *ECCV*, 2020. 9, 14

[82] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision (ECCV)*, 2016. 2, 3, 5

[83] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 2, 3

[84] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2017. 2

A. Interactive Demo

We provide an interactive demo in the code release to show the image editing use case with our anycost generator. Anycost generator provides a fast preview to enable interactive image editing. A screenshot of the demo interface is provided in Figure 9.

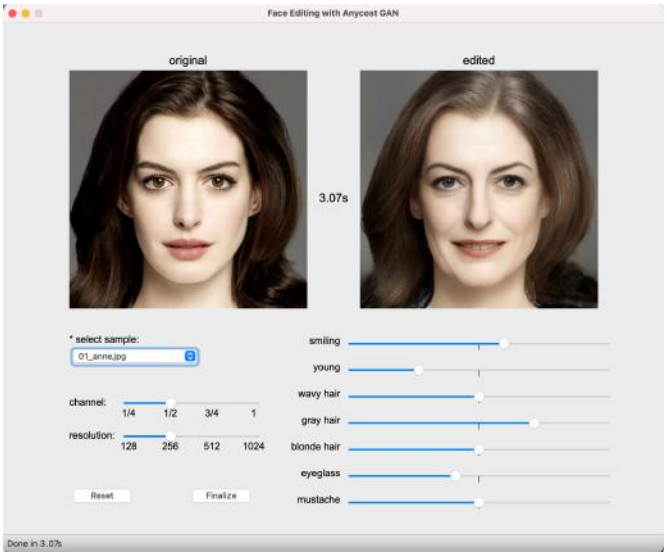


Figure 9: The UI interface of our face editing demo. Please refer to our demo video for more details.

B. Experimental Details

B.1. Training

We used similar training hyper-parameters as [45]. For the first stage, *i.e.*, training a vanilla StyleGAN2 model, we used exactly the same training configuration. For multi-resolution training and adaptive-channel training, since we can distill from the pre-trained generator, we do not need to add the path length regularization [45], which accelerates the training.

For multi-resolution training, at reach iteration, we randomly sampled two resolutions from the four possible resolutions for both generator and discriminator training, which improves the data efficiency and stabilizes training. During adaptive-channel training, we randomly sampled one channel configuration at each iteration. For the uniform channel setting, we randomly sampled one of the four ratios using a uniform probability. For the flexible channel setting, we followed [50] to use a "sandwich" rule for channel sampling (*i.e.*, sample the full generator for 25% of the time, sample the smallest generator for 25% of the time, and randomly sample a generator at the rest of the time). We empirically find that it makes the convergence more stable and improves the FID of the full generator.

B.2. Evolutionary search

We use evolutionary search to find the best sub-generator under a certain resource budget. We use a population size $P = 50$ and max iterations $\mathcal{T} = 20$. For each iteration, we only keep the top-10 best candidates in the current generation, and then generate a new generation consisting of 25 *crossover* candidates and 25 *mutation* candidates. For a crossover, we randomly sample two top candidates and cross them to generate a new one; for mutation, we keep a mutation probability of 0.1. For the new candidates, We directly evaluate the output difference w.r.t. the full generator or FIDs without further fine-tuning, as our anycost generator is well trained for any subset of weights.

B.3. Metrics

Fréchet Inception Distance (FID). For FID [30] evaluation on high-resolution images (1024 for FFHQ [45] and 512 for LSUN Car [75]), we follow [45] to compute the FID using 50k images (*i.e.*, FID-50k). For FID evaluation on low-resolution images (≤ 256 on FFHQ), we follow [43] to compute the FID using 70k images (*i.e.*, FID-70k), so that we can compare with the results in the paper.

Perceptual path length (PPL). We follow [45] to compute the PPL metric, using 100k random samples.

Attribute consistency. We trained our own attribute predictor to evaluate the attribute consistency (can be found in the code release). We randomly sampled 10k images with truncation $\psi = 0.5$ to compute the attribute consistency.

C. Training Time

We discuss the training time on FFHQ [45] dataset that consists of 70k samples. Our experiments are based on a PyTorch implementation of StyleGAN2 [45][†]. Given a normally trained StyleGAN2 model, we first train the model for 5 million images to support multi-resolution, and then train it for another 5 million images to support adaptive channel numbers. On 8 NVIDIA Titan RTX GPUs, the training takes 5.3 days in total, which is roughly 60% of the StyleGAN2 training time. Note that we just need to train the model once, and we can run it under various computational budgets by using a smaller sub-generator, while distillation or compression-based methods need to train each sub-generator architecture one by one.

The cost of evolutionary search is small compared to the training. We used 2048 samples to evaluate the difference between the full generator and sub-generators' outputs and find the one with minimal difference. Given a certain budget, the evolutionary search can be done in 4 hours on a single GPU. Therefore, the marginal cost to support extra devices is very small. We compare the GPU hours *vs*. #devices in

[†]https://github.com/rosinality/stylegan2-pytorch

Figure 10. We can save 3.8× GPU hours when deploying on 6 different devices. This is a practical case when we cover devices from smartphones, iPads, laptops, to cloud servers.

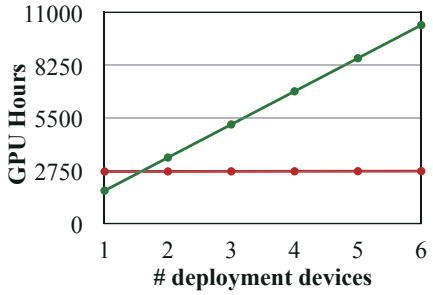


Figure 10: Anycost GAN has a small marginal cost for supporting extra devices.

D. Image Projection and Editing

D.1. Extracting Editing Directions

To extract the editing directions, We use the trained generator to randomly generate 100,000 $\{\mathbf{w}, G(\mathbf{w})\}$ pairs, and use a pre-trained attribute classifier to give the binary attribute predictions (*e.g.*, young or old) [68]. The samples are generated using truncation $\psi = 0.7$. We then compute the decision boundary between positive $\mathbf{w}$'s and negative $\mathbf{w}$'s. Please refer to [68] for more details. We empirically find that, finding the editing directions is less precise compared to image projection. Therefore, we only used the largest generator to compute the directions, which works pretty well for other sub-generators.

D.2. Discussion on Projection *vs*. Editing

We notice that sometimes a better projected image may not lead to an easily editable latent code. For example, in Figure 11, we compare two projected latent codes (row 1 and row 2). The second projected code has lower LPIPS loss, but after editing, there are clear artifacts on the image. We hypothesize that the projected latent code may not be in the training domain of the generator [81]. Therefore, the reconstruction loss alone cannot fully reflect the quality of the projected code in image editing scenario. We left the more in-depth discussion to the future work.

In practice, we empirically find that encoder-based initialization and 100 iterations of backward optimization can lead to both good visual similarity and editing ability at the same time.

D.3. Qualitative Comparison of Image Projection

We have shown the quantitative results for image projection in the main paper (Table 6). Here we compare the qualitative results of baseline image projection and consistency-aware projection to show the necessity of applying consistency constraints.

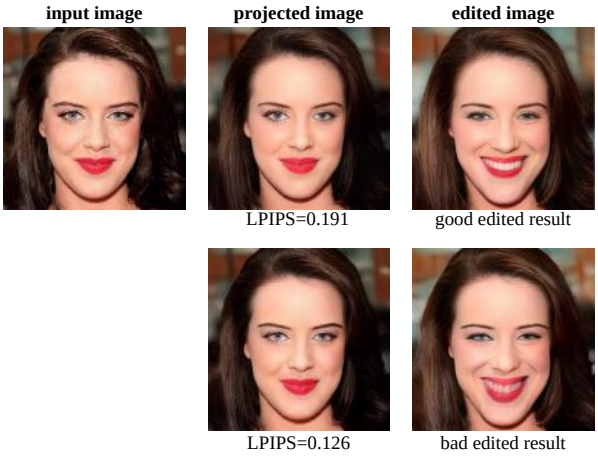


Figure 11: A better projected latent code (*i.e.*, with lower LPIPS loss) is not necessarily easily editable. The second projection (row 2) has lower LPIPS loss, but after editing, there are clear artifacts.

Figure 12: Our consistent-aware encoder achieves better consistency between the full and the sub-generator's outputs. Vanilla encoder trained only for the full generator leads to visible differences in clothing colors, skin tones, and more fine-grained details such as eye shapes. Both encoders have achieved the same reconstruction performance for the full generator.Results are hand-picked to reflect the difference.

Encoder-based projection. We compare the baseline encoder optimized only for the full generator, and our consistency-aware encoder optimized for both full generator and sub-generators. Both encoders achieve the same

level of reconstruction performance for the full generator (measured by LPIPS+MSE loss). Given the encoder's predicted $\mathbf{w}$ code, we compare the outputs of the full generator and the $8\times$ faster sub-generator. The results are shown in Figure 12. We can see that our consistency-aware encoder results in a smaller gap between the full generator and sub-generators' outputs. Vanilla encoder trained only for the full generator leads to a poor reconstruction performance for the $8\times$ faster generator, and a noticeable difference between the two outputs: there are differences in both macro properties like skin tones, clothing colors, and fine-grained details such as eye shapes. The comparison demonstrates the advantage for our consistency-aware encoder training. *Note that the images are picked to clearly demonstrate the difference.*

Optimization-based projection. Given the encoder's prediction, we further optimize $\mathbf{w}$ to improve the projection quality. We provide a visual comparison of vanilla optimization and consistency-aware optimization in Figure 13. Again, we compare the outputs of the full generator and $8\times$ faster sub-generator. First, despite that the encoder's predicted $\mathbf{w}$ can reconstruct the input image reasonably well (the second column), additional optimization can help refine details (*e.g.*, the gaze in the first example, the earring in the second example), which is critical for editing high-resolution real images. Second, the consistent-aware optimization provides better consistency between the full and sub-generators' outputs (*e.g.*, the clothing in the first example, and the eye shape in the second examples, the skin tone and cap color in the third example). The improved consistency allows us to provide a more accurate, fast preview for the final output. *Note that the images are picked to clearly demonstrate the difference*.

E. Additional Synthesis Results

Anycost GAN (uniform) on FFHQ. We provide more results of Anycost GAN (uniform) on FFHQ dataset [45] in Figure 14.

Anycost GAN (flexible) on FFHQ. We provide more results of Anycost GAN (uniform) on FFHQ dataset [45] in Figure 15.

Anycost GAN (uniform) on LSUN Car. We provide more results of Anycost GAN (uniform) on LSUN Car dataset [75] in Figure 16.

F. Additional Projection Results

We provide additional projection results on FFHQ in Figure 17. The target images are taken from the test set of CelebA-HQ dataset [42] dataset. Anycost generators produce consistent outputs at various computational budgets.

G. Additional Editing Results

We provide more editing results of FFHQ generator. Similarly, we compare the edited images from the full generator and the $8\times$ computation-reduced sub-generator. The results are shown in Figure 18.

H. Failure Cases

The major failure cases we observed are that, some fine details are missing or changed in smaller generators' outputs, like earrings and rim. We show some example images in Figure 19. It could be a problem when we try to edit the details of images. The failure case could be mitigated by using a slightly larger sub-generator to provide better consistency (*e.g.*, $0.5\times$ instead of $0.25\times$ in this example).

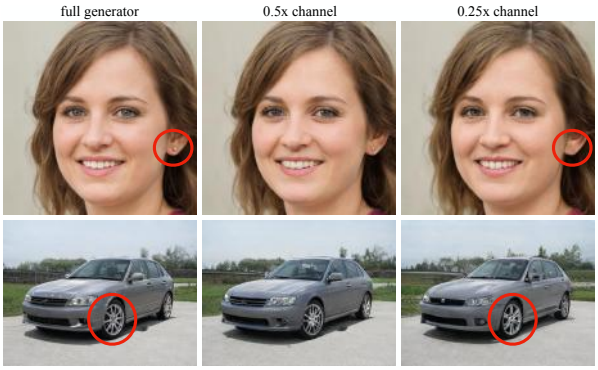


Figure 19: Some fine details are changed when using a sub-generator for synthesis.

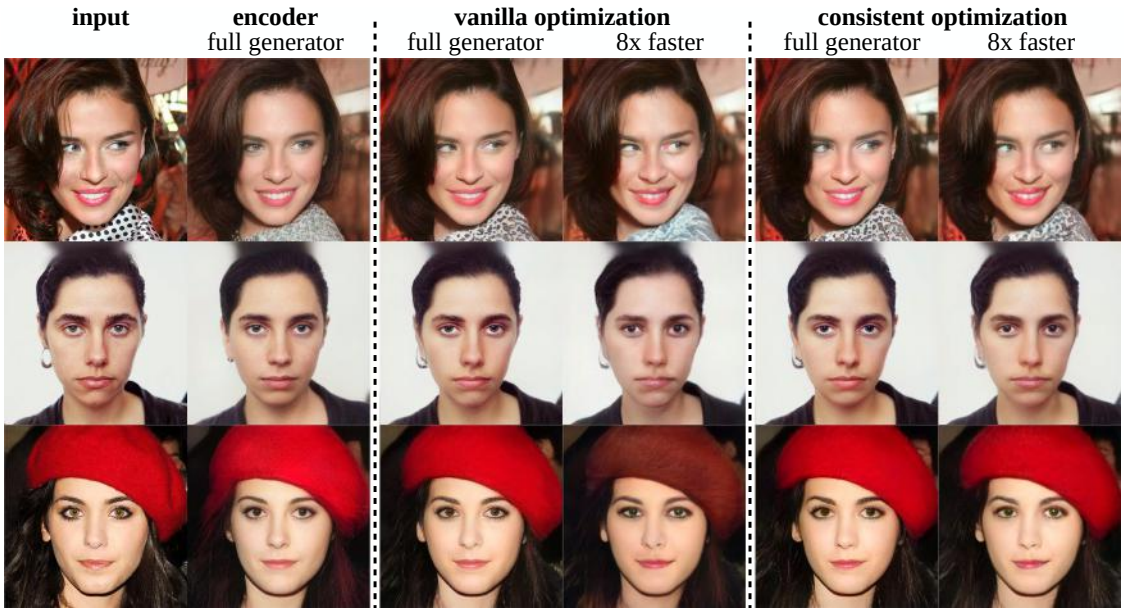


Figure 13: Consistent-aware optimization leads to better image projection for sub-generators, reducing the output difference between the full generator and sub-generators. Results are hand-picked to reflect the difference.

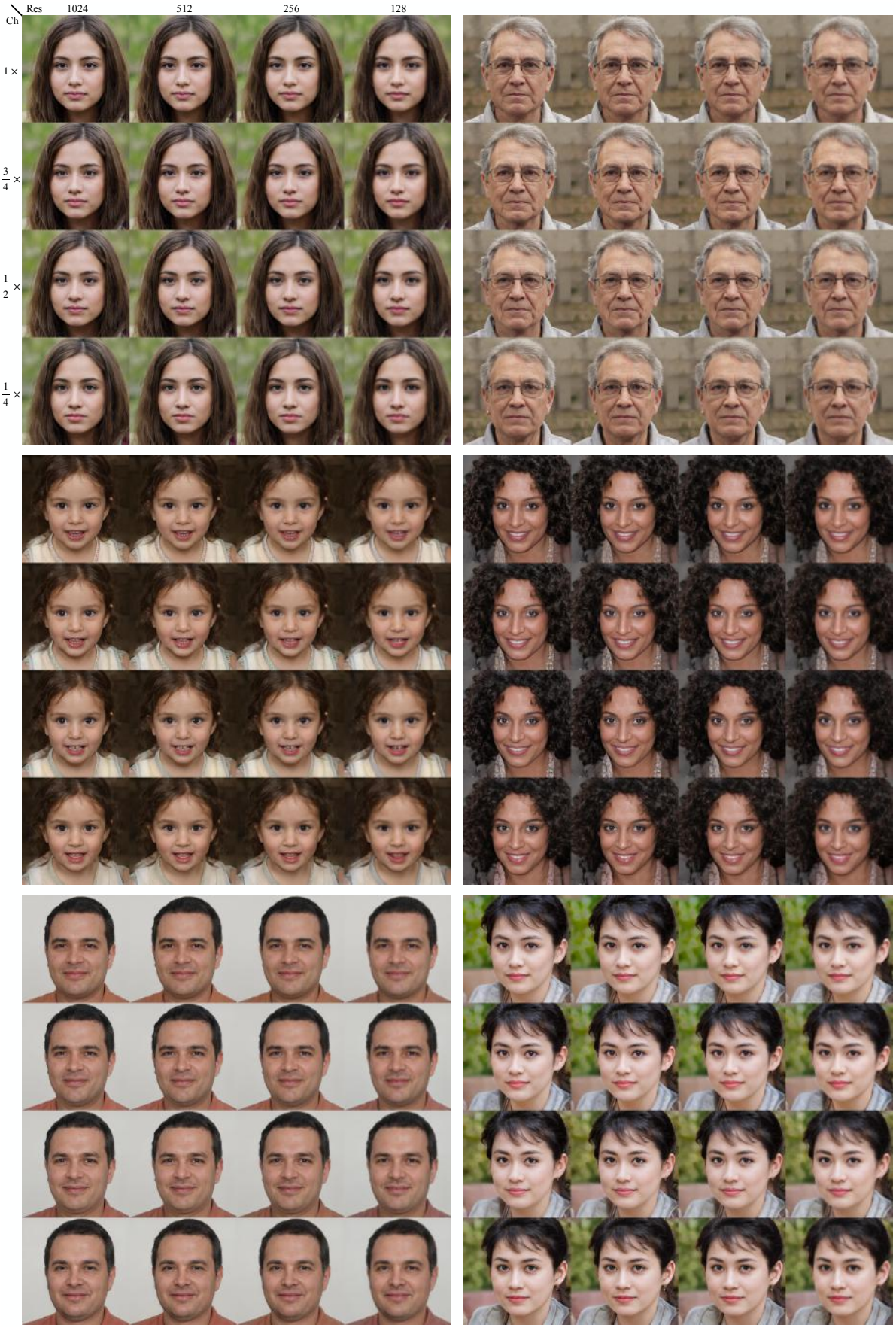


Figure 14: Additional synthesis results for anycost generator (uniform) on FFHQ.

Figure 15: Additional synthesis results for anycost generator (flexible) on FFHQ.

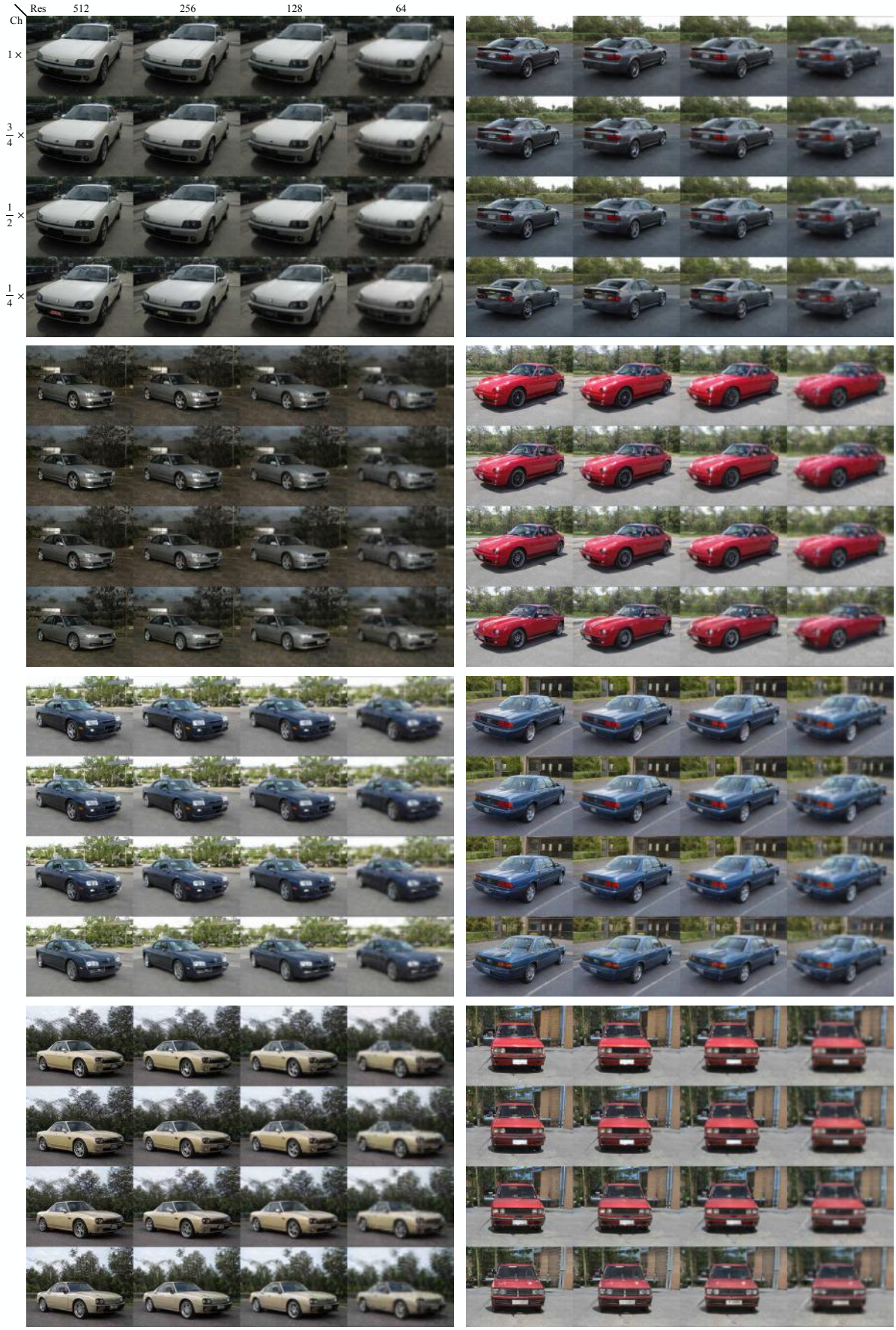


Figure 16: Additional synthesis results for anycost generator (uniform) on LSUN Car.

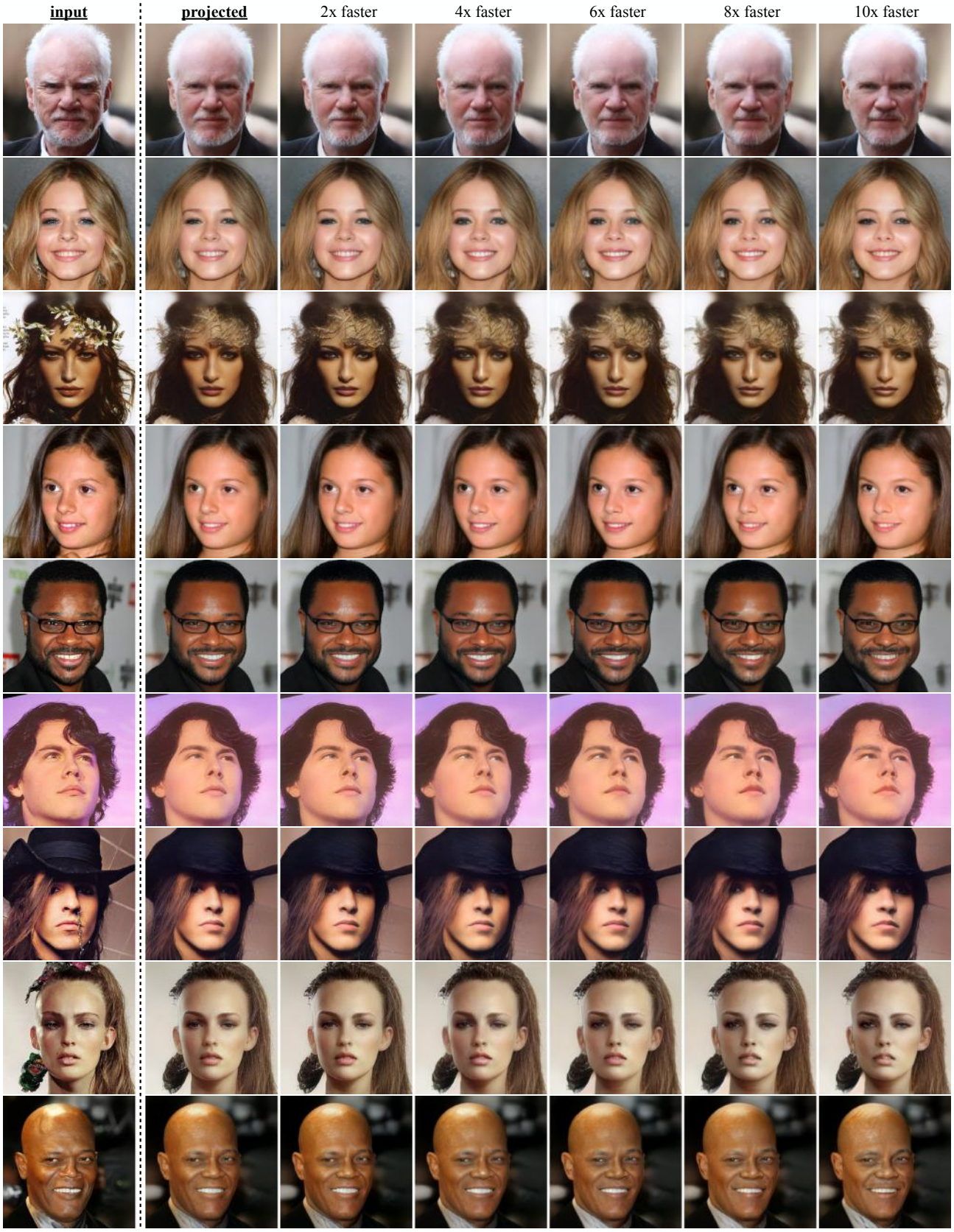


Figure 17: Additional projection results for FFHQ anycost generators. Zoom in for a better view.

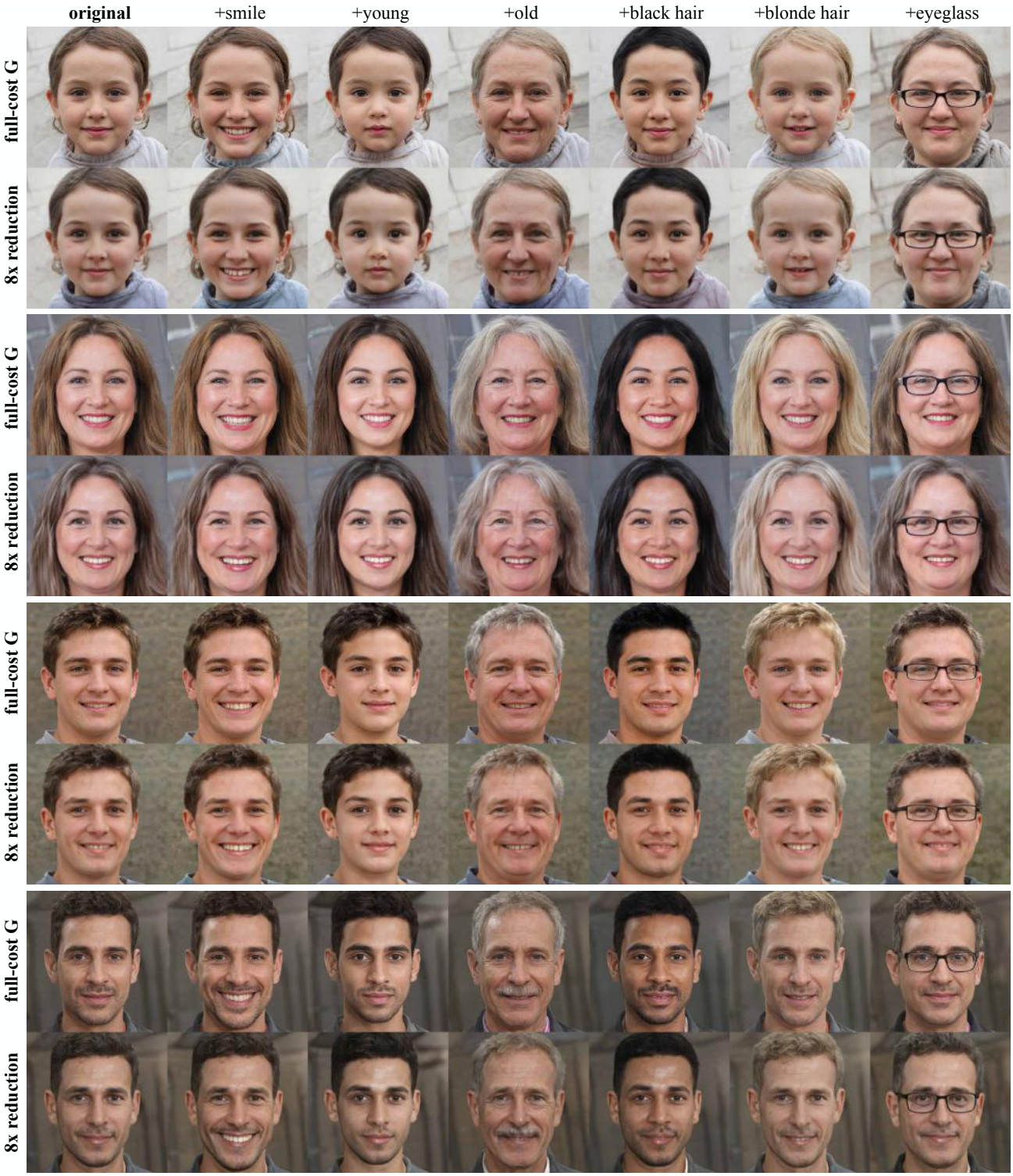


Figure 18: Editing results for FFHQ anycost generator.