E^{2}GAN: Efficient Training of Efficient GANs for Image-to-Image Translation

Yifan Gong[*1,2], Zheng Zhan[*2], Qing Jin[1], Yanyu Li[1,2], Yerlan Idelbayev[1], Xian Liu[1], Andrey Zharkov[1], Kfir Aberman[1], Sergey Tulyakov[1], Yanzhi Wang[2], Jian Ren[1]
[1] Snap Inc. [2] Northeastern University
gong.yifa@northeastern.edu, jren@snapchat.com

Abstract

One highly promising direction for enabling flexible real-time on-device *image editing is utilizing data distillation by leveraging large-scale text-to-image diffusion models, such as Stable Diffusion, to generate paired datasets used for training generative adversarial networks (GANs). This approach notably alleviates the stringent requirements typically imposed by high-end commercial GPUs for performing image editing with diffusion models. However, unlike text-to-image diffusion models, each distilled GAN is specialized for a specific image editing task, necessitating costly training efforts to obtain models for various concepts. In this work, we introduce and address a novel research direction:* can the process of distilling GANs from diffusion models be made significantly more efficient? *To achieve this goal, we propose a series of innovative techniques. First, we construct a base GAN model with generalized features, adaptable to different concepts through fine-tuning, eliminating the need for training from scratch. Second, we identify crucial layers within the base GAN model and employ Low-Rank Adaptation (LoRA) with a simple yet effective rank search process, rather than fine-tuning the entire base model. Third, we investigate the minimal amount of data necessary for fine-tuning, further reducing the overall training time. Extensive experiments show that we can efficiently empower GANs with the ability to perform real-time high-quality image editing on mobile devices with remarkable reduced training cost and storage for each concept.* [1]

1. Introduction

Recent development of diffusion-based image editing models has witnessed remarkable progress in synthesizing contents containing photo-realistic details with full of imagination [35, 36, 38, 39]. Albeit being creative and powerful, these generative models typically require a huge amount of computation even for inference and storage for saving weights. For example, Stable Diffusion [38] has more than one billion parameters and takes 30 seconds to conduct iterative denoising process to get one image on T4 GPU. Such low-efficiency issue prohibits their real-time application on mobile devices [24].

Exiting works try to tackle the problem through two main directions. One is accelerating the diffusion models by designing efficient model architecture or reducing the number of denoising steps [16, 23, 26, 40]. However, these efforts still struggle to obtain models that can run in real-time on mobile devices [24]. Another area focuses on data distillation, where diffusion models are leveraged to create datasets to train other mobile-friendly models, such as generative adversarial networks (GANs) for image-to-image translation [33, 54]. Nevertheless, although GAN is efficient for on-device deployment, each new concept still asks for the *costly training* of a GAN model from *scratch*.

In this work, we propose and aim to address a new research direction: *can the GAN models be trained efficiently under the data distillation pipeline to perform real-time on-device image editing?*

To tackle the challenge, we introduce **E^{2}GAN**, powered with the following techniques for the **E**fficient training and **E**fficient inference of **GAN** models with the help of diffusion models:

- First, we construct a base GAN model trained from various prompts and the corresponded edited images obtained from diffusion models. It enables efficient transfer learning for different new concepts by fine-tuning, rather than training models from scratch, to reduce the training cost. Meanwhile, the base GAN model achieves fast inference with fewer parameters on mobile devices (as in Fig. 1 *Right*), and maintaining high-performance.
- Second, we identify that only partial layers are necessary to be fine-tuned for new concepts. LoRA is applied on these layers with a simple yet effective rank search process, eliminating the need to fine-tune the entire base model (as in Fig. 1 *Left*). It brings two advantages – both

[1]Project website: https://yifanfanfanfan.github.io/e2gan/

the training cost and saving storage for each new concept are significantly reduced.
- Third, we investigate the amount of data for fine-tuning the base model for various concepts. The reducing the amount of training data helps reduce the training cost and time for adapting the base model to new concepts.

We show extensive experiments results to demonstrate that by using our approach, we can efficiently distill the image editing capability from a large-scale text-to-image diffusion model into GAN models via data distillation (examples in Fig. 5). The distilled GAN model showcases real-time image editing capabilities on mobile devices. We hope our work can shed light on how to democratize the diffusion models into efficient on-device computing.

2. Related Works

Generative Models learn the joint data distribution to generate new samples, such as VAEs [18, 37], GANs [7, 30, 55], auto-regressive models [27, 41, 46, 46, 50], and diffusion models [3, 10, 29, 43–45]. Among these, diffusion models demonstrate strong capability of generating images with high-fidelity [36, 38], at the cost of bulky model size and numerous sampling steps during inference. Several studies try to accelerate the image generation process of the diffusion models [23, 26, 40]. However, they still struggle to achieve real-time on-device generation. On the contrary, GANs are more efficient in terms of model size and inference speed for image editing [14, 22, 47]. To this end, we leverage the approach of data distillation to transfer knowledge from diffusion models to lightweight GANs that are compatible with real-time inference on mobile devices.

Efficient GANs. Exiting works actively explore the reduction of the inference runtime for GANs by using various model compression techniques, such as efficient architecture design [14, 22], network pruning and quantization [47, 48], and neural architecture search [5, 47]. On the other hand, the research about training cost savings for GANs is quite limited, as most works typically train all the parameters of a GAN model from scratch for the image-to-image translation task, involving large computing efforts. This work aims to fine-tune a very small portion, *i.e.* 1%, of the pre-trained models with the partial training data to reduce the training cost. Thus, the training of GAN can be tiny in terms of both parameters and data. There are many efforts on efficient training [12, 19], in particular the sparse training [4, 21, 52]. However, these methods rely on the mask of trainable parameters, which in turn is determined during training with a huge bunch of data. In contrast, our method adopts pre-defined learnable components, and only fine-tunes on a small fragment of data to make the transfer learning progress efficient and effective.

Table 1. **Comparison of model size, FLOPs, and latency** for different works [13, 24, 33]. Co-Mod-GAN [54] is trained following the pipeline in Pix2pix-zero [33]. Reported latency is averaged over 100 runs on iPhone 14 Pro. The training time of pix2pix and Co-Mod-GAN is measured on a single NVIDIA H100 GPU.

Model	Param num	FLOPs	Latency	Train time
SnapFusion	861M	>1T	1956 ms	7680 hours
Pix2pix with 9 RB	11.4M	56.9G	21.0 ms	16 min
Co-Mod-GAN	79.2M	98.2G	not supported	110 min

3. Motivation

Though diffusion models possess the advantages of generating high-fidelity images, the huge model size, high computation cost, and numerous sampling steps pose significant challenges for implementation on widely adopted mobile platforms with limited capacities. Even recent attempts at accelerating diffusion models, such as SnapFusion [24], still require nearly 2 seconds to generate a single image on an iPhone 14 Pro, as shown in Tab. 1. This efficiency issue strictly hinders their real-time application, *e.g.*, image editing with 30 frames per second (FPS).

In contrast, various efficient and mobile-friendly GAN designs exist. For instance, the pix2pix model with 9 ResNet Blocks (RBs) takes only 21 ms to generate an edited image on an iPhone 14 Pro. Recognizing the inefficiency in directly accelerating diffusion models and the lightweight nature of certain GANs, researchers have explored data distillation as an alternative research direction. This approach involves transferring the knowledge of diffusion models to GANs. Latest work pix2pix-zero [33] creates training data to train Co-Mod-GAN for model acceleration, yet it is not supported on mobile devices. Furthermore, the training time to obtain the Co-Mod-GAN for a new concept is still costly, which takes 110 min as shown in Tab. 1.

To overcome the above-mentioned limitations, the objective of this work is to achieve **efficient distillation** of diffusion models to **mobile-friendly real-time** GANs. Specifically, efficient distillation refers to minimizing the training efforts needed to obtain the GAN model for a new concept. Furthermore, when deployed on a mobile device after efficient distillation, the mobile-friendly real-time GANs should exhibit low latency (<33.3 ms) and demand minimal storage for a new concept.

4. Methods

Here we present our approach that efficiently transfers the image editing capability from large-scale text-to-image diffusion models to on-device real-time GANs. In the following sections, we first give an overview of our knowledge transfer pipeline (Sec. 4.1). Then, we study efficient training strategies to get a series of on-device models with *fewer* training costs and *less* storage, while maintaining

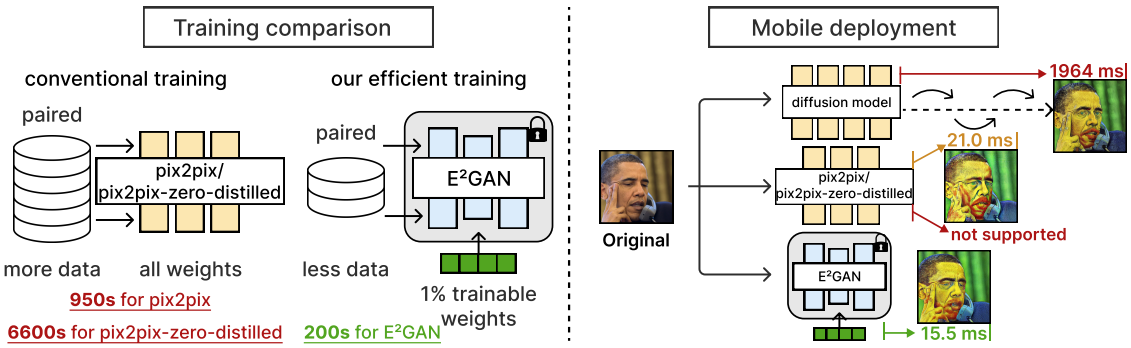

Figure 1. **Overview of E^{2}GAN.** ***Left****: Training Comparison.* Conventional GAN training, such as pix2pix [13] and pix2pix-zero-distilled that distills Co-Mod-GAN [54] using data from pix2pix-zero [33], requires all the weights trained from scratch, while our efficient training significant reduces the training cost by only fine-tuning 1% weights with only *portion* of training data. ***Right****: Mobile Inference Comparison.* Our efficient on-device model can achieve real-time (30FPS, iPhone 14) runtime and is faster than pix2pix and diffusion model, while the pix2pix-zero-distilled model (Co-Mod-GAN) is not supported on device.

high-quality image generation (Sec. 4.2).

4.1. Overview of Knowledge Transfer Pipeline

Benefits of Knowledge Transfer. Despite the powerful image generating capability of text-to-image diffusion models, they suffer from large model size and inefficient sampling [24]. For example, the denoising UNet of Stable Diffusion [38] has 860 millions of parameters. Even the latest efficient diffusion model, SnapFusion [24], costs around 2 seconds to generate a single image on iPhone 14 Pro, limiting the real-time applications for diffusion models. On the contrary, the image-to-image GANs are typically lightweight with much fewer parameters that can synthesize one image with only one forward pass. For instance, a pix2pix model with ResNet-based network has a latency of around 21ms on iPhone 14 Pro. Thus, these efficient architectures are more compatible with the mobile devices to achieve real-time image generation. To leverage the benefits of both models, we employ GANs as the image editing model and transfer knowledge of diffusion models to GAN by training the GANs with data created by the diffusion model. Such a data distillation pipeline serves as the foundation for our whole framework.

Pipeline for Dataset Creation. To enable the data distillation, we use the diffusion models to edit real images to obtain the edited images, forming pairs of data along with the used text prompts to create the training datasets, which then be utilized to train the image-to-image GAN model. The real images come from FFHQ [15] and Flickr-Scenery [2], covering diverse content and are challenging for content editing. For diffusion models, we choose the recent works for image editing, such as Stable Diffusion [38], Instruct-Pix2Pix (IP2P) [1], Null-text Inversion (NI) [28], ControlNet [53], and InstructDiffusion [6].

Training Objectives. With paired images and the associated prompts, we train the efficient GANs for image translation by using the conventional adversarial loss. Specifically, given the original image $\mathbf{x}$ and the editing prompt $\mathbf{c}$, the image generator $\mathcal{G}$ and discriminator $\mathcal{D}$ are jointly optimized as follows:

$$\min_{\theta_g}\max_{\theta_d} \underbrace{\lambda\, \mathbb{E}_{\mathbf{x},\tilde{\mathbf{x}}^c,\mathbf{z},\mathbf{c}}\left[\|\tilde{\mathbf{x}}^c - \mathcal{G}(\mathbf{x},\mathbf{z},\mathbf{c};\theta_g)\|_1\right]}_{\ell_1 \text{ loss}} + \underbrace{\mathbb{E}_{\mathbf{x},\tilde{\mathbf{x}}^c}\left[\log \mathcal{D}(\mathbf{x},\tilde{\mathbf{x}}^c;\theta_d)\right] + \mathbb{E}_{\mathbf{x},\mathbf{z},\mathbf{c}}\left[\log(1-\mathcal{D}(\mathbf{x},\mathcal{G}(\mathbf{x},\mathbf{z},\mathbf{c};\theta_g);\theta_d))\right]}_{\text{conditional GAN loss}}, \tag{1}$$

where $\tilde{\mathbf{x}}^c$ denotes the image generated by the diffusion model ϵ conditioned on the concept $\mathbf{c}$ of the target style, *i.e.*, $\tilde{\mathbf{x}}^c = \epsilon(\mathbf{x},\mathbf{c})$, $\mathcal{G}$ and $\mathcal{D}$ denote the generator and discriminator function parameterized by θ_g and θ_d, respectively, $\mathbf{z}$ is a random noise introduced to increase the stochasticity of output, and λ can be used to adjust the relative importance between ℓ_1 loss and the conditional GAN loss.

4.2. Efficient Training of GAN Models

Diffusion-based generative models can perform image editing on-the-fly, while existing lightweight GAN-based networks require training to be adapted to the new concept. The training of GAN models for various image translation tasks require substantial computation costs. Additionally, there is high storage demand for saving the trained weights. To mitigate such training and storage costs, we introduce three main techniques to reduce the number of trainable parameters and the demanded data for model fine-tuning: *First*, we establish a *base GAN model* equipped with generalized features and representations, ready to be leveraged for new concepts (Sec. 4.2.1). *Second*, starting from the base model, we identify key parameters to optimize during fine-tuning for a new concept, bolstered by the application of Low-Rank Adaptation (LoRA) [11] to further reduce the number of parameters (Sec. 4.2.2). *Third*, we explore the possibility of tiny fine-tuning where the training data are first clustered and only those near the cluster centers are

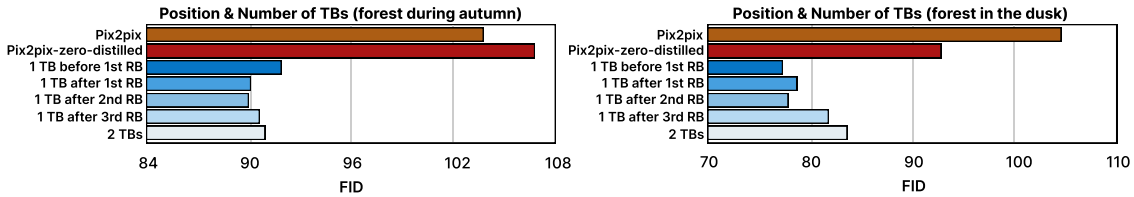

Figure 2. **FID comparison** of applying TBs in image generators trained on two datasets (*Left:* `forest during autumn`, *Right:* `forest in the dusk`). Vertical axis shows the position to inserting TBs. Pix2pix-zero-distilled uses pix2pix-zero for creating datasets to train Co-Mod-GAN [35].

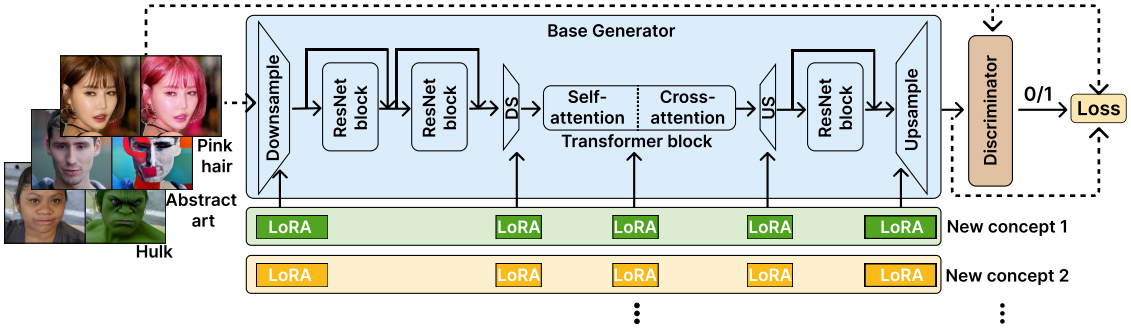

Figure 3. **Overview of E^{2}GAN model architecture.** The generator is composed of down/up-sampling layers, 3 RBs, and 1 TB. The base generator is trained on multiple representative concepts. New concepts are achieved by fine-tuning LoRA parameters on crucial layers.

Table 2. The model size, FLOPs, and latency of E^{2}GAN. The reported latency is an average of 100 runs measured on the GPU of an iPhone 14 Pro.

Model	Param num	FLOPs	Latency
3RB+1TB	**7.1M**	**23.6G**	**15.5 ms**
3RB+2TB	10.1M	26.6G	21.0 ms

used (Sec. 4.2.3).

4.2.1 Base GAN Model Construction

To obtain model weights for a new target concept with as few training efforts as possible, we explore transfer learning from a pre-trained base GAN model, instead of training from scratch. The base model should possess the capability of a more general features and representations, which can be learned from multiple image translation tasks, allowing the new concept to leverage existing knowledge. Thus, we opt to train the base model on a mixed dataset comprising diverse concepts.

The construction of the image-to-image model $\mathcal{G}$ serves as the first step in obtaining such a base model. This model should fulfill three key criteria: (1) the ability to learn multiple concepts; (2) achievement of real-time inference on mobile devices; and (3) strong image generation capabilities. We start from the classic ResNet generator with 9 RBs that is widely adopted [13, 31, 55]. To incorporate the text information of the concept and facilitate a more holistic understanding of global shapes and structure, we introduce Transformer Blocks (TBs) with self-attention and cross-attention modules into the architecture. For expedited inference purpose, we reduce the number of RBs from 9 to 3. The subsequent steps involve determining the number and position of TBs.

Number of TBs. We train models with different architecture designs, *e.g.* different number of TBs, and evaluate both the efficiency (in terms of model size, FLOPs, and latency) and image generation capability (in terms of the FID [9] between the images generated by GANs and diffusion models). The results are presented in Tab. 2 and Fig. 2, respectively. Interestingly, we find that one TB is enough for generating high quality images. Introducing more TBs does not further improve the performance yet brings in more computation cost. Notice that to reduce the inference cost of the introduced TB, we apply a downampling operation to halve the feature map size before sending it into the TB, and use an upsampling layer to recover the feature map size for the following operations.

Position of TBs. Additionally, we find that the position of the TB is important for the final performance of the image generation. First, the TB should be placed between the last downsample layers and the first upsample layers to avoid high computations on mobile devices, due to the high resolution of features. Second, we apply attention to different positions of the network bottleneck. Particularly, the TB can be inserted between one of the following: (1) before the first RB; (2) after the first RB; (3) after the second RB; and (4) after the third RB. As evident in Fig. 2, all these options lead to a generator with better performance than the con-

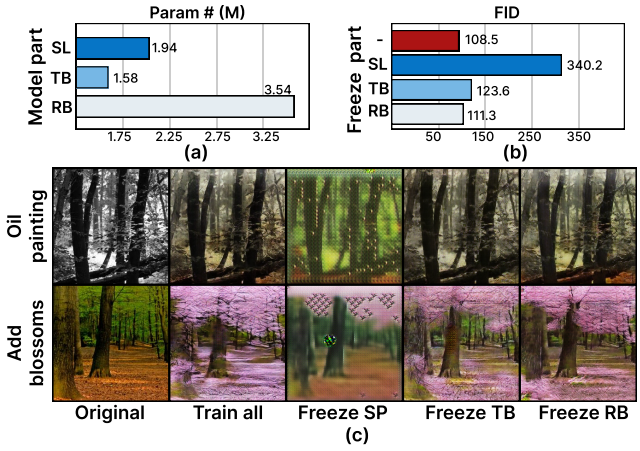

Figure 4. **Crucial weights analysis** via freezing partial weights in the base model. **(a)** Number of parameters for each part of the base model; **(b)** Averaged FID across 10 different concepts on the Flicker-Scenery dataset when freezing partial weights of base model. '-' indicates fine-tune all the weights; **(c)** The generated images when freezing each part of the base model.

ventional CONV-only networks used in pix2pix [13] and pix2pix-zero-distilled [33]. For our model, we place the TB after the second RB.

Therefore, our architecture is finalized with an overall architecture in Fig. 3. It achieves faster inference speeds, reduces the number of parameters, and lowers computational costs compared to existing image-to-image models, as shown in in Tab. 2 and Fig. 2. With the architecture determined, the base model is trained on a subset of concepts denoted as $\mathcal{C} = \{\mathbf{c}_1, \cdots, \mathbf{c}_K\}$, where each concept $\mathbf{c}_k$ is selected among different concepts by K-means clustering [25] based on the average of the CLIP image embedding [34] of uniformly sampled images.

4.2.2 Crucial Weights for Fine-Tuning

To save the training and storage cost, we reduce the number of trainable parameters during fine-tuning. Specifically, we pre-define trainable layers that occupy a small portion of weights from the base model. Then, we apply LoRA on top of the trainable layers. In this way, we only optimize 1.29% of the weights from the base model during the fine-tuning, greatly reducing the training cost and storage cost for a new concept.

Inspired by the recent work of customized diffusion [20], where a pre-trained diffusion model can be fine-tuned to a personalized version with updating only partial weights, we explore the feasibility to identify the minimal set of tunable weights for GAN. We aim to determine such a set that is sufficient for fine-tuning the base model to a new target concept. To this end, we analyze the components of the GAN model, which mainly consist of three parts: (1) sampling layers (SL) with downsampling and upsampling; (2) transformer block (TB); and (3) intermediate ResBlock (RB).

Identifying Crucial Layers. We systematically and empirically study the impact of each part in the image-to-image task by freezing each part individually, with results provided in Fig. 4. Combining Fig. 4(b) & (c), we see that SL plays a more crucial role in maintaining the quality of generated images, identified by the high FID score and low image quality. SL might be more crucial for constructing the desired output texture, yet intermediate RB might contain lower-level information that are common among styles. Meanwhile, compared to RB, TB has a fewer amount of parameters (1.58M *v.s.* 3.54M in Fig. 4(a)), while it is more important in keeping performance (123.6 *v.s.* 111.3 in Fig. 4(b)). Considering the situation with a limited training budget, RB has a lower priority to be optimized.

LoRA on Crucial Layers. From the perspective of maintaining image generating quality, it is better to include TB in training as self-attention modifies the image with a better holistic understanding and the cross-attention module takes the information from the given target concept. However, combing SL and TB leads to 3.42M parameters to be updated, taking account 47.90% of the entire model weights. To fine-tune the crucial layers with much fewer trainable parameters, we investigate the best way of incorporating Low-Rank Adaptation (LoRA) [11] into GAN training, which introduces two trainable low-rank weight matrices besides the original weight for each layer identified as crucial.

Rank for LoRA. With the leverage of LoRA, when fine-tuning to a new concept, the weights of base model are *frozen*, while only the two low-rank matrices with much fewer parameters for each crucial layer are updated to save computation and storage costs. For instance, for a CONV layer i with weights $\theta_i \in \mathbb{R}^{h \times w \times k_h \times k_w}$, we apply two low-rank matrices with rank r_i, *i.e.* $\theta_i^A \in \mathbb{R}^{h \times r_i \times k_h \times k_w}$ and $\theta_i^B \in \mathbb{R}^{r_i \times w \times 1 \times 1}$, to approximate the gradient update $\nabla\theta_i$. Given multiple crucial layers, determining the appropriate rank for *each of them* is important. Prior works mostly rely on manual setting [11] for deciding the rank value, due to a huge search space for the rank. However, in our task, the rank should be pre-fixed for different concepts to avoid the rank search process when a new concept comes. To tackle this challenge, we randomly sample K concepts and conduct a simple yet effective rank search process. For each concept, we start by assigning r_i as 1 for each crucial layer i, and upscale the rank for every e epochs by doubling the rank value, until r_i reaches the upper threshold τ_i for the layer i. The threshold τ_i is determined by the size of the weight. We evaluate the FID performance at the end of each e training epochs. If the performance saturates, the rank value from the best FID performance setting is returned as the rank for the concept. Typically, a larger rank can pro-

vide more model capability. Thus, the largest returned rank among the K selected concepts is viewed as r^* for the future use of a new concept.

4.2.3 Similarity Clustering to Reduce Training Data

Reducing the amount of training data can directly result in a reduction in the training time. Thus, we aim to investigate data efficiency as a means of decreasing the training cost in addition to the crucial weight update for E^{2}GAN. We find not all data are indispensable for reliable training, but only a small subset is necessary. We obtain this small subset in an unsupervised manner with a selection of the data crowding around the clustering center on the whole dataset.

To identify the small subset of essential data, we conduct unsupervised learning to analyze the structure of the training data. We first extract an embedding for each image $\mathbf{x}$ with an extractor $\mathcal{E}$. Then, we apply clustering on the embeddings by the K-Means algorithm [25] to obtain $K < N$ clusters (N it the total number of images), each with center μ_k. The embeddings within the same cluster have a closer distance among each other, indicating a higher *similarity* of the data points. To reduce the data amount while maintaining data diversity for the good generalization ability of the model, one data point, which is the closet to the center μ_k, is selected for each of the K clusters.

With our data selection method using K clusters, we further reduce the number of training iterations by N/K times. In contrast to prior methods involving additional computations in the training process to shrink the dataset [49, 51], our Similarity Clustering (SC) data reduction is tailored for expediting the training of image editing tasks. It reduces the training data volume directly before the training process without incurring any additional costs during the training.

5. Experiments

In this section, we provide the detailed experimental settings and results of our proposed method. More details as well as some ablation studies can be found in the Appendix.

Paired Data Preparation. We verify our method on $1,000$ images from FFHQ dataset [15] and Flickr-Scenery dataset [2] with image resolution as 256×256. The images in the target domain are generated with several different text-to-image diffusion models, including Stable Diffusion [38], IP2P [1], NI [28], ControlNet [53], and InstructDiffusion [6]. The generated images with the best perceptual quality among diffusion models are selected to form with the real images into paired datasets. To perform training and evaluation of GAN models, we divide the image pairs from each target concept into training/validation/test subsets with the ratio as $80\%/10\%/10\%$.

Table 3. **FID comparison.** FID is calculated between the images generated by GAN-based approaches and diffusion models. Reported FID is averaged across different concepts (30 for FFHQ and 10 for Flicker Scenery).

Method \ Dataset	FFHQ	Landscape
Pix2pix	86.03	114.2
Pix2pix-zero-distilled	87.76	132.6
E^{2}GAN	**80.28**	**109.37**

Baselines. We compare E^{2}GAN with image-to-image translation methods like pix2pix [13] (image generator with 9 ResNet blocks) and pix2pix-zero-distilled that distills Co-Mod-GAN [54] using data generated by pix2pix-zero [33].

Training Setting. We follow the standard approach that alternatively updates the generator and discriminator [7]. The training is conducted from an initial learning rate $2e-4$ with mini-batch SGD using Adam solver [17]. The total training epochs is set to 100 for E^{2}GAN, and 200 for pix2pix [13] and pix2pix-zero-distilled [33] for them to converge well. For SC (Sec. 4.2.3), we choose the cluster number as 400 and use the feature extractor $\mathcal{E}$ as FaceNet [42] on FFHQ dataset and CLIP image encoder [34] on Flicker Scenery dataset. To train the base model, we use 20 prepared tasks/datasets from FFHQ dataset and 7 from Flicker Scenery dataset.

Evaluation Metric. We compare the images generated by GAN-based models and diffusion-based models by calculating Clean FID proposed by [32] on the test sets.

5.1. Experimental Results

Qualitative Results. The synthesized images in the target domain obtained by E^{2}GAN and other methods are shown in Fig. 5. The original images are listed at the leftmost column, and the synthesized images for the target concept obtained by diffusion models, pix2pix, pix2pix-zero-distilled, and E^{2}GAN are shown from top to bottom. The tasks span a wide range, such as changing the age, artistic styles, and editing the seasons. According to the results, E^{2}GAN is able to modify the original images to the target concept domain by updating only the LoRA parameters. For instance, for the `green lantern` prompt on FFHQ dataset, diffusion model fails to modify the image, pix2pix and pix2pix-zero-distilled add colors to wrong areas, while E^{2}GAN generates the image that fits the prompt well. As for the `add blossoms` prompt on the Flicker Scenery dataset, E^{2}GAN preserves the structure of the original image better than other models while editing the image as desired.

Quantitative Comparisons between E^{2}GAN and other baseline methods are provided in Tab. 3. Note that for each concept, pix2pix and pix2pix-zero-distilled are trained on the whole training dataset of 800 samples. E^{2}GAN begins with a base model and is fine-tuned with only 400 sam-

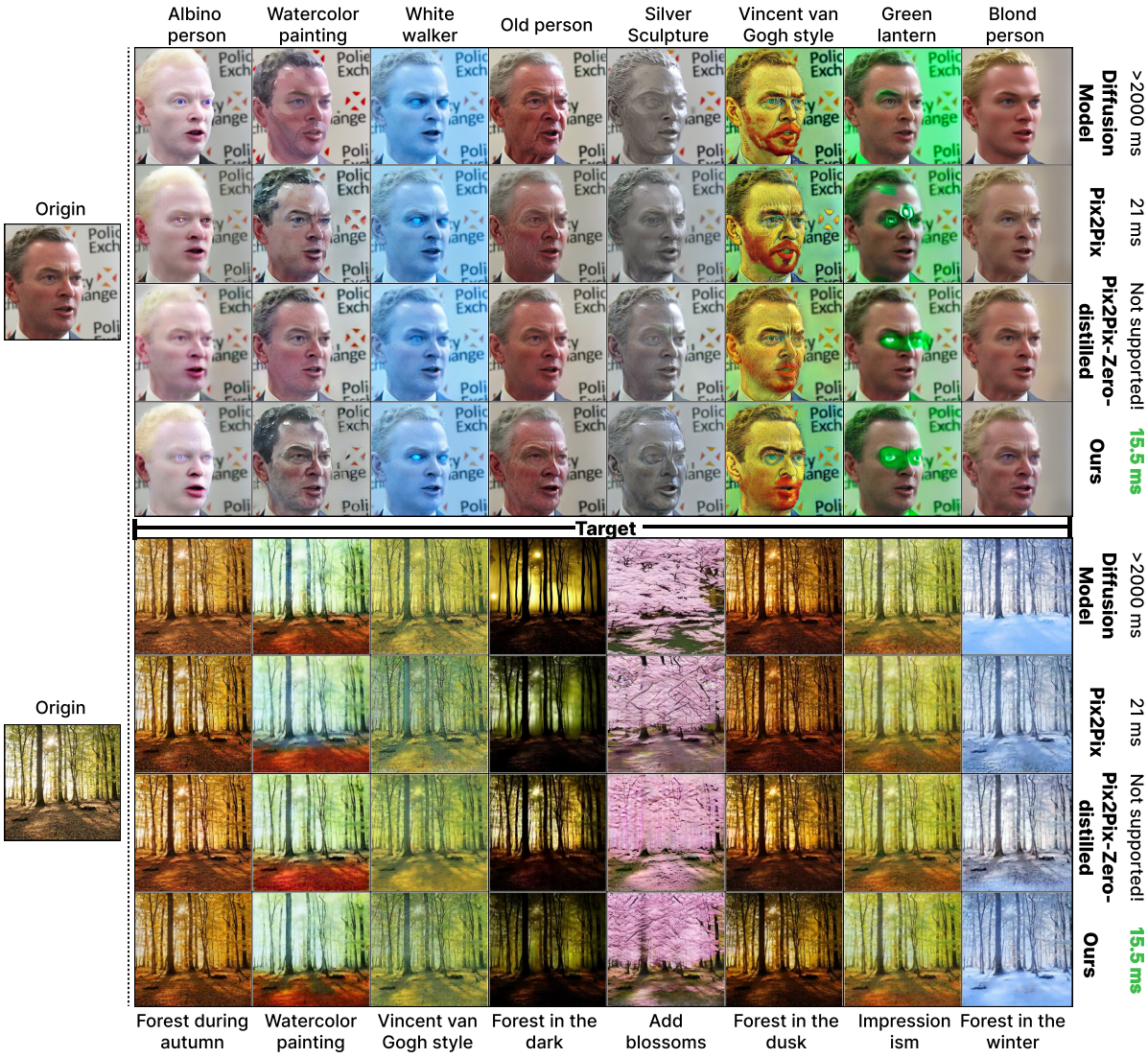

Figure 5. **Qualitative comparisons** on various tasks. The *leftmost* column shows two original images and the remaining columns present the corresponding synthesized images in the target concept domain, where target prompts are shown at the bottom row. We provide images generated by various models.

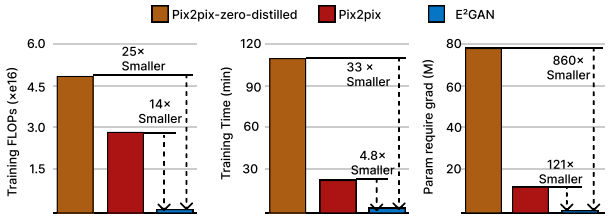

Figure 6. **Training cost comparison** of conventional GAN and E^{2}GAN. ***Left*****:** Training FLOPs. ***Middle*****:** Training time. ***Right*****:** Number of parameters that required gradient update, which also equals to the weights need to be saved for a concept.

ples to obtain models for different target concepts. The results demonstrate E^{2}GAN is able to reach a even better FID performance than the conventional GAN training techniques like pix2pix and pix2pix-zero-distilled, indicating high-fidelity of generated images.

Training Cost Analysis between E^{2}GAN and other approaches are provided in Fig. 6. Compared with pix2pix and pix2pix-zero-distilled, E^{2}GAN greatly saves the training FLOPs of $14\times$ and $25\times$, respectively, and accelerates the training time by $4.8\times$ and $33\times$, respectively. Moreover, E^{2}GAN only requires updating 0.092M parameters for a new concept, greatly saving the storage requirement when training models for various tasks/concepts, *i.e.* $869\times$ less than pix2pix-zero-distilled.

Notably, E^{2}GAN requires *much fewer* trainable parameters, training data, and training time than other GAN-based approaches to reach even *better* generation quality, *i.e.* E^{2}GAN has lower FID than pix2pix on FFHQ (80.28

Table 4. **Analysis (FID) of various base models** on FFHQ.

Concept \ Base model	Ours	20 random	200 art concepts	Single concept
White walker	**40.18**	53.92	40.32	51.99
Blond person	**48.01**	52.77	61.50	55.58
Sunglasses	**38.49**	40.54	41.37	44.12
Vangogh style	71.82	78.58	**68.21**	78.06

Table 5. **Analysis of searching LoRA rank** on the Flicker Scenery dataset. The reported FID values are averaged over 10 different target concepts.

Scheme	FID	# of Param
Our searched	**109.37**	0.092M
Upcale 1×	130.98	0.056M
Upcale 4×	111.42	0.164M
Random	129.87	0.100M

v.s. 86.03). Furthermore, E^{2}GAN enjoys a faster inference speed on mobile devices (Tab. 1). The good performance of E^{2}GAN is originated from our effective framework design, including the efficient model architecture and efficient training strategy (Sec. 4.2).

5.2. Ablation Analysis

We provide ablation analysis to understand the impact of each components. We first study the effectiveness of the base model determination. After that, we provide an analysis on the LoRA rank search. Finally, we discuss the effect of our data selection.

Analysis of Base Model Determination. We study the impacts of our base model determination method discussed in Sec. 4.2.1 by comparing our method with the following three settings: (1) train the base model on 20 *random* concepts; (2) train the base model on 200 artist concepts; (3) train the base model on a *single* concept `old person` from the FFHQ dataset. The results are demonstrated in Tab. 4. Note our method is obtained by training on 20 selected representative concepts. The results indicate our base model construction outperforms or matches the alternatives across the evaluated concepts. This underscores the efficacy of our base model in enhancing performance. In contrast, the single concept base model generally performs worse. Furthermore, simply increasing the amount of concepts does not necessarily leads to better performance as indicated by training the base model with 200 art concepts.

Analysis of LoRA on Crucial Layers. Tab. 5 presents an evaluation of the effectiveness of our LoRA rank search on the Flicker Scenery dataset. The table reports the FID averaged across 10 different target concepts, as well as the number of LoRA parameters for various schemes. We compare our method with the other three settings: (1) upscale the rank 1× for each crucial layer by doubling the rank from the initialization until the rank reaches the threshold; (2) upscale the rank 4× for each crucial layer from the initialization; and (3) random assign ranks for the crucial layers. Our searched scheme achieves the lowest FID value of 109.37 while maintaining a relatively low number of parameters as 0.092M. This demonstrates the effectiveness of our LoRA rank search approach.

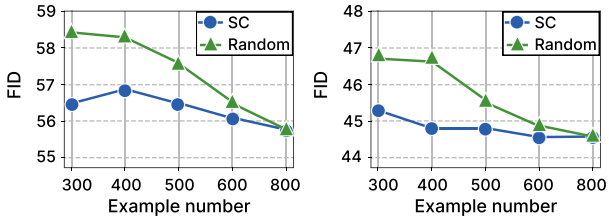

Figure 7. **Comparisons of Data Selection Rule.** Prompts for the *left* and *right* figures are `old person` and `put on a pair of sunglasses`, respectively.

Analysis of Cluster Number of Data Selection. To investigate our sampling rule SC for obtaining training samples (proposed in Sec. 4.2.3 to reduce the number of training data), we compare it with randomly sampling method. Random sampling is implemented as shuffling the training data randomly and only accessing the first K examples as training data. The comparisons are conducted with different number of training samples K. We show the results in Fig. 7 and can draw the following observations. First, SC provides better FID performance in all scenarios, indicating the effectiveness of our sampling method by enriching data diversity. Second, the cluster number, *i.e.* the number of target training samples, influences the SC performance to some extent. More training examples (clusters) do not necessarily lead to better performance. Furthermore, SC can work for a wide range of different number of training samples by providing models with good FID performance.

6. Conclusion

This paper addresses the growing demand for efficient on-device image editing by introducing a novel research direction, that is the efficient training of efficient GAN models via distilling the large-scale text-to-image diffusion models with data distillation. The proposed framework, E^{2}GAN, incorporates a hybrid training pipeline that can efficiently adapt a pre-trained text-conditioned GAN model, which has real-time inference speed on mobile devices, to different concepts, while significantly mitigating computational and storage demands. Extensive experimental results validate the effectiveness of E^{2}GAN, that enables real-time image editing capabilities on mobile devices and brings the power of diffusion models for efficient on-device computing.

Limitation and Discussion. Generating high-quality images using diffusion models can be challenging for diverse prompts, which in turn restricts the expansion of our training datasets. Moreover, utilizing diffusion models for data collection remains an expensive endeavor. Develop-

ing efficient techniques to rapidly construct well-paired and high-quality datasets from diffusion models would greatly enhance the training of E^{2}GAN. Furthermore, while our approach does succeed in reducing training costs, alternative methods such as hyper-networks [8] have the potential to eliminate the need for training. This could result in the seamless obtaining real-time on-device GAN models as needed without any fine-tuning.

References

[1] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. *arXiv preprint arXiv:2211.09800*, 2022. 3, 6, 1

[2] Yen-Chi Cheng, Chieh Hubert Lin, Hsin-Ying Lee, Jian Ren, Sergey Tulyakov, and Ming-Hsuan Yang. Inout: Diverse image outpainting via gan inversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11431–11440, 2022. 3, 6

[3] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021. 2

[4] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pages 2943–2952. PMLR, 2020. 2

[5] 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

[6] Zigang Geng, Binxin Yang, Tiankai Hang, Chen Li, Shuyang Gu, Ting Zhang, Jianmin Bao, Zheng Zhang, Han Hu, Dong Chen, et al. Instructdiffusion: A generalist modeling interface for vision tasks. *arXiv preprint arXiv:2309.03895*, 2023. 3, 6, 1

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. 2, 6

[8] David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations*, 2017. 9

[9] 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. *Advances in neural information processing systems*, 30, 2017. 4

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 2

[11] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 3, 5

[12] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32, 2019. 2

[13] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 2, 3, 4, 5, 6

[14] Qing Jin, Jian Ren, Oliver J Woodford, Jiazhuo Wang, Geng Yuan, Yanzhi Wang, and Sergey Tulyakov. Teachers do more than teach: Compressing image-to-image models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13600–13611, 2021. 2

[15] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019. 3, 6, 2

[16] Bo-Kyeong Kim, Hyoung-Kyu Song, Thibault Castells, and Shinkook Choi. On architectural compression of text-to-image diffusion models. *arXiv preprint arXiv:2305.15798*, 2023. 1

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[18] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[19] Urs Köster, Tristan Webb, Xin Wang, Marcel Nassar, Arjun K Bansal, William Constable, Oguz Elibol, Scott Gray, Stewart Hall, Luke Hornof, et al. Flexpoint: An adaptive numerical format for efficient training of deep neural networks. *Advances in neural information processing systems*, 30, 2017. 2

[20] Nupur Kumari, Bingliang Zhang, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. Multi-concept customization of text-to-image diffusion. *arXiv preprint arXiv:2212.04488*, 2022. 5

[21] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. In *ICLR*, 2019. 2

[22] Muyang Li, Ji Lin, Yaoyao Ding, Zhijian Liu, Jun-Yan Zhu, and Song Han. Gan compression: Efficient architectures for interactive conditional gans. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5284–5294, 2020. 2

[23] Muyang Li, Ji Lin, Chenlin Meng, Stefano Ermon, Song Han, and Jun-Yan Zhu. Efficient spatially sparse inference for conditional gans and diffusion models. *arXiv preprint arXiv:2211.02048*, 2022. 1, 2

[24] Yanyu Li, Huan Wang, Qing Jin, Ju Hu, Pavlo Chemerys, Yun Fu, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Snapfusion: Text-to-image diffusion model on mobile devices within two seconds. *arXiv preprint arXiv:2306.00980*, 2023. 1, 2, 3

[25] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. 5, 6

[26] Chenlin Meng, Ruiqi Gao, Diederik P Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of

guided diffusion models. *arXiv preprint arXiv:2210.03142*, 2022. 1, 2

[27] Jacob Menick and Nal Kalchbrenner. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. *arXiv preprint arXiv:1812.01608*, 2018. 2

[28] 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, 6, 1

[29] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021. 2

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

[31] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*, pages 319–345. Springer, 2020. 4

[32] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11410–11420, 2022. 6

[33] Gaurav Parmar, Krishna Kumar Singh, Richard Zhang, Yijun Li, Jingwan Lu, and Jun-Yan Zhu. Zero-shot image-to-image translation. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–11, 2023. 1, 2, 3, 5, 6

[34] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 5, 6

[35] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021. 1, 4

[36] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 1, 2

[37] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014. 2

[38] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. 1, 2, 3, 6

[39] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022. 1

[40] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022. 1, 2

[41] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017. 2

[42] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015. 6

[43] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015. 2

[44] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[45] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 2

[46] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR, 2016. 2

[47] Haotao Wang, Shupeng Gui, Haichuan Yang, Ji Liu, and Zhangyang Wang. Gan slimming: All-in-one gan compression by a unified optimization framework. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*, pages 54–73. Springer, 2020. 2

[48] Peiqi Wang, Dongsheng Wang, Yu Ji, Xinfeng Xie, Haoxuan Song, XuXin Liu, Yongqiang Lyu, and Yuan Xie. Qgan: Quantized generative adversarial networks. *arXiv preprint arXiv:1901.08263*, 2019. 2

[49] Zifeng Wang, Zheng Zhan, Yifan Gong, Geng Yuan, Wei Niu, Tong Jian, Bin Ren, Stratis Ioannidis, Yanzhi Wang, and Jennifer Dy. Sparcl: Sparse continual learning on the edge. *arXiv preprint arXiv:2209.09476*, 2022. 6

[50] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022. 2

[51] Geng Yuan, Xiaolong Ma, Wei Niu, Zhengang Li, Zhenglun Kong, Ning Liu, Yifan Gong, Zheng Zhan, Chaoyang He, Qing Jin, et al. Mest: Accurate and fast memory-economic sparse training framework on the edge. *Advances in Neural Information Processing Systems*, 34:20838–20850, 2021. 6

[52] Geng Yuan, Yanyu Li, Sheng Li, Zhenglun Kong, Sergey Tulyakov, Xulong Tang, Yanzhi Wang, and Jian Ren. Layer

freezing & data sieving: Missing pieces of a generic framework for sparse training. *Advances in Neural Information Processing Systems*, 35:19061–19074, 2022. 2

[53] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023. 3, 6, 1

[54] Shengyu Zhao, Jonathan Cui, Yilun Sheng, Yue Dong, Xiao Liang, Eric I Chang, and Yan Xu. Large scale image completion via co-modulated generative adversarial networks. *arXiv preprint arXiv:2103.10428*, 2021. 1, 2, 3, 6

[55] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 2, 4

E^{2}GAN: Efficient Training of Efficient GANs for Image-to-Image Translation

Supplementary Material

Table of Contents

A. More Implementation Details

A.1. Details for Diffusion Model

We apply most recent diffusion-based image editing models to create paired datasets, which include Stable Diffuison (SD) [38], Instruct-Pix2Pix (IP2P) [1], Null-text inversion (NI) [28], ControlNet [53], and Instruct Diffusion [6]. For all these models, we use the checkpoints or pre-trained weights reported from their official websites[2].

More specifically, for SD, the strength, guidance scale, and denoising steps are set to 0.68, 7.5, and 50, respectively. For IP2P, the images are generated with 100 denoising steps using a text guidance of 7.5 and an image guidance of 1.5. For NI, each image is generated with 50 denoising steps and the guidance scale is 7.5. The fraction of steps to replace the self-attention maps is set in the range from 0.5 to 0.8 while the fraction to replace the cross-attention maps is 0.8. The amplification value for words is 2 or 5, depending on the quality of the generation. For ControlNet, the control strength, normal background threshold, denoising steps, and guidance scale are 1, 0.4, 20, and 9, respectively. For Instruct Diffusion, the denoising steps, text guidance, and image guidance are set as 100, 5.0, and 1.25, respectively.

[2]SD v1.5: https://huggingface.co/runwayml/stable-diffusion-v1-5, IP2P: http://instruct-pix2pix.eecs.berkeley.edu/instruct-pix2pix-00-22000.ckp, NI: https://huggingface.co/CompVis/stable-diffusion-v1-4, ControlNet: https://huggingface.co/lllyasviel/ControlNet/blob/main/models/control_sd15_normal.pth, InstructDiffusion: https://github.com/cientgu/InstructDiffusion.

A.2. Hyperparameters in LoRA Rank Search

During the process of searching LoRA rank, the rank r_i for each crucial layer i is upscaled once for every e epochs until r_i reaches the upper threshold τ_i for the layer i. In the experiments, e is set as 10. The rank threshold τ_i is determined by the size of the layer. More specifically, the crucial layers include: (1) four CONV-based upsampling layers with the shape as $[3, 64, 7, 7]$, $[64, 128, 3, 3]$, $[128, 256, 3, 3]$, and $[256, 256, 3, 3]$; (2) four corresponding downsampling layers by transpose CONV with the same set of weight shape as upsampling; and (3) transformer blocks with projection matrices q,k,v with shape as $[256, 256]$, and multilayer perceptron (MLP) module with shape as $[2048, 256]$ and $[256, 1024]$. Based on the weight size, the rank threshold τ is set as 1, 4, 16, and 32 for the four upsampling/downsampling layers, respectively, and 1 for the layers in the transformer block. After the search process, the suitable rank is determined as 1, 4, 8, 8 for the four upsampling/downsampling layers.

Table 6. FID comparison between E^{2}GAN model architecture (3RB+1TB) and pix2pix (9RB) under the setting of training-from-scratch.

Concept	E^{2}GAN (3RB+1TB)	Pix2pix (9RB)
Angry person	**49.56**	55.16
Pale person	**42.65**	49.14
Tan person	**42.47**	51.37
Young person	**51.27**	56.10

B. More Analysis for the Efficient Image-to-Image Model

B.1. Effectiveness of Model Architecture

Here we further show the effectiveness of our efficient model architecture design in complementary to the results in Sec. 4.2.1. We compare our 3RB+1TB design against the 9RB design used in pix2pix for several concept settings. The results are shown in Tab. 6 with both models trained on the entire training set of 800 samples. From which we can see that the 3RB+1TB design can reach higher FID with

fewer parameters and FLOPs (as in Tab. 1). For instance, a 3RB+1TB model in the target concept domain of `pale person` has a FID as 42.65, decreasing the FID value by 6.49 compared to the 9RB model of pix2pix.

B.2. Sampling Operations for Transformer Block

As mentioned in Sec. 4.2.1, we apply a downsampling operation with a CONV layer to halve the feature map size before sending it into the transformer block, and use an upsampling layer implemented by transpose CONV operation to recover the feature map size for the following operations to reduce the amount of computations. We conduct another set of experiments on the Flicker Scenery dataset to see if these sampling operations can be replaced by pooling and unpooling operations, such that a smaller model size can be reached. We first train these two models on the selected prompts to get the base model. Then, we fine-tune the entire model with all the training data for a new concept. The comparision results are shown in Tab. 7. From the results, we can observe that though applying pooling operations can reduce the number of parameters from base model by 1.2M, the FID performance becomes much worse. Thus, we use CONV operation instead of pooling to tackle the feature map reduction and recovery for the transformer block.

Table 7. FID performance of replacing the downsampling and upsampling layers for the transformer block with Max Pool and Max Unpool operations.

Operation	CONV + transpose CONV	Max Pool + Max Unpool
Concept \ Model Size	7.1M	5.9M
`Forest in the dark`	**121.60**	190.05
`Impressionism painting`	**88.52**	135.96
`Forest in the autumn`	**88.82**	141.29

C. More Ablation Analysis for the Base Model

C.1. Pre-train with Multiple Concepts for Conventional GAN Training

We investigate if conventional GAN training such as pix2pix can benefit from fine-tuning a pre-trained base model, as leveraged in E^{2}GAN. To verify this, we follow the same step as E^{2}GAN to pre-train pix2pix with the selected 7 prompts/datasets on the Flicker Scenery dataset. Then, the base model is fine-tuned to adapt to other concepts. The results in Tab. 8 show that pix2pix does not gain much benefits from pre-training. Moreover, the performance becomes even worse, such as for the concept `Vangogh style` (FID degrades from 138.77 to 151.20 with a pre-trained base model). The results indicate that with our efficient architecture design, our base model possess the capability of a more general features and representations when trained on multiple concepts. The transformer block with self-attention modifies the image with a better holistic understanding and the cross-attention module takes the information from the given target concept. Thus, our method allows the new concept to better leverage existing knowledge, which is not possessed by prior methods.

Table 8. FID performance of fine-tuning from a pre-trained base model for pix2pix on Flicker Scenery dataset.

Method	Pixpix		E^{2}GAN
Concept \ Pretrain	✓	✗	✓
`Vangogh style`	151.2	138.77	**117.41**
`Add blossoms`	157.76	150.96	**146.42**
`Forest in the winter`	119.31	122.35	**119.15**

C.2. Autoencoder as Pre-trained Base Model

In E^{2}GAN, we first train the GAN model with multiple diverse concepts to get a pre-trained base model, and then fine-tune it to other concepts. We have shown multiple base model settings in Sec. 5.2. One may wonder if the pre-trained base model can be chosen as an auto-encoder, *e.g.* the base model encodes the input data into a lower-dimensional representation and then decodes it back into the original data, instead of being trained on other concepts. To verify this, we conduct experiments by first training an auto-encoder on the original images in the subset of FFHQ [15] with only the ℓ_1 loss in Eq. 1, then fine-tune the auto-encoder following the same method as fine-tuning a pre-trained GAN. The results are compared in Tab. 9. We find that auto-encoder is not comparable as fine-tuning a GAN trained on a single concept as `old person`, not to mention our base model that is pre-trained on multiple concepts. For instance, for the target style `angry person`, tuning from a base model pre-trained to generate `old person` can give an FID as low as 54.48, yet tuning from the auto-encoder results in a much worse FID of 110.35. This might due to the simplicity of the auto-encoder, which only needs to generate the original image and does not necessarily include other semantic information, either coarse-grained global features, or fine-grained local details. In constrast, the GAN models include more information like texture or color, during training. From this observation, in E^{2}GAN, we adopt a model pre-trained on several concepts instead of using auto-encoder as base model.

D. Ablation on the Influence of Longer Training Time

E^{2}GAN greatly saves the training time compared to conventional GAN training while maintaining good image synthesize ability. To see if training longer can lead to better performance, we add further experiments to increase the

Table 9. The FID performance of using autoencoder as the pre-trained base model.

Base model \ Concept	Angry person	White walker
Auto-encoder	110.35	80.43
Old person	54.48	51.99
Ours	**54.27**	**40.18**

Table 10. The FID comparison between training E^2GAN for 100 epochs and 200 epochs.

Concept	Train 100 epochs	Train 200 epochs
Forest in the dark	115.32	114.17
Oil painting	**110.87**	111.93
Forest in the spring	**122.77**	124.91

training time by doubling the training epochs. The results can be found in Tab. 10. The reported FID is evaluated on the model weights obtained at the end of training. The results show that training longer will not bring obvious performance improvements for E^2GAN, but leads to more computation cost. The results indicate that our efficient E^2GAN is able to reach good performance with fewer epochs compared to conventional GAN training.

E. Additional Qualitative Results

We provide more example images generated by our approach and other baseline methods in Fig. 8, 9, 10, 11, and 12.

Figure 8. **Qualitative comparisons** on various tasks. The *leftmost* column shows two original images and the remaining columns present the corresponding synthesized images in the target concept domain, where target prompts are shown at the bottom row. We provide images generated by various models.

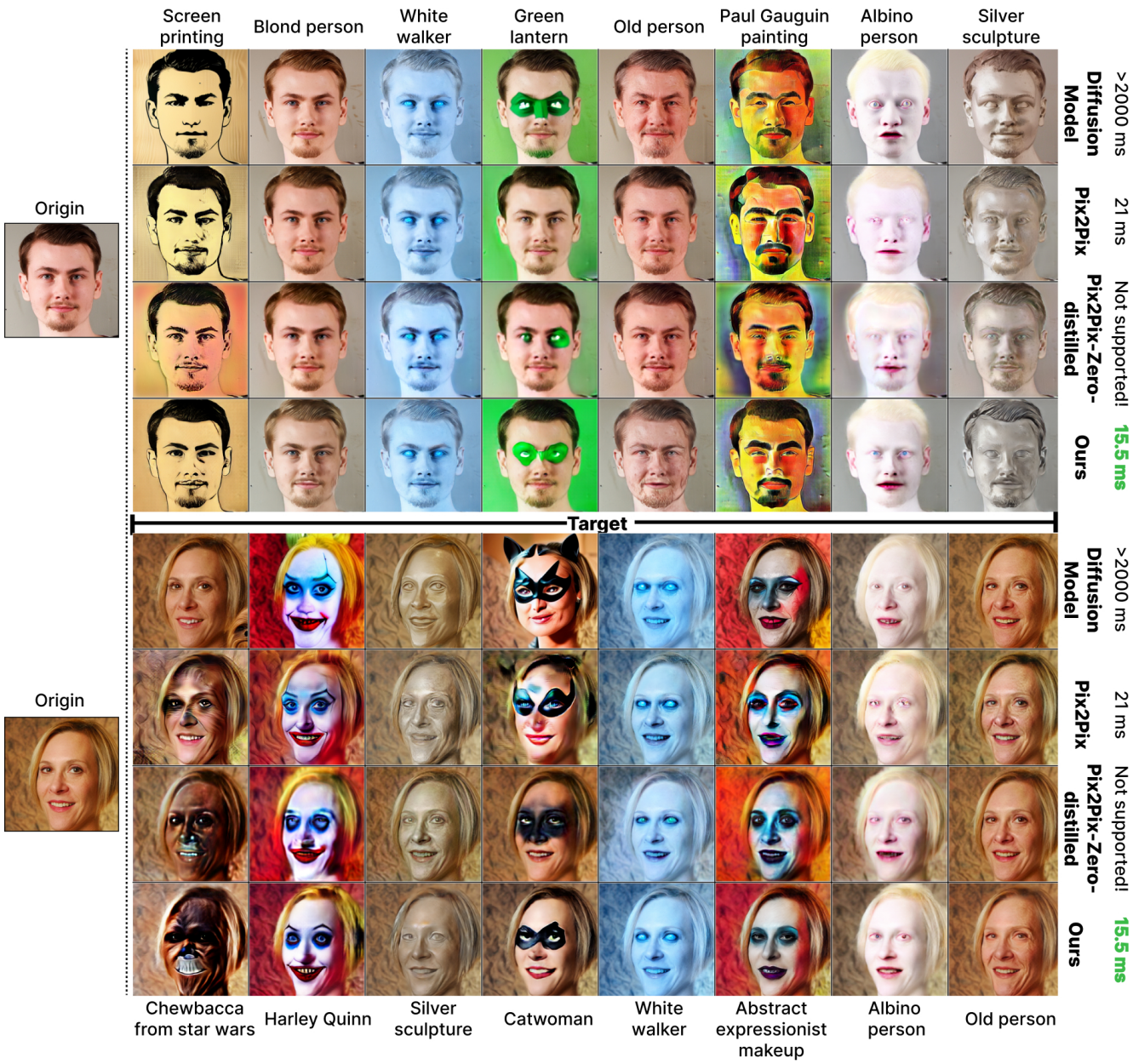

Figure 9. **Qualitative comparisons** on various tasks. The *leftmost* column shows two original images and the remaining columns present the corresponding synthesized images in the target concept domain, where target prompts are shown at the bottom row. We provide images generated by various models.

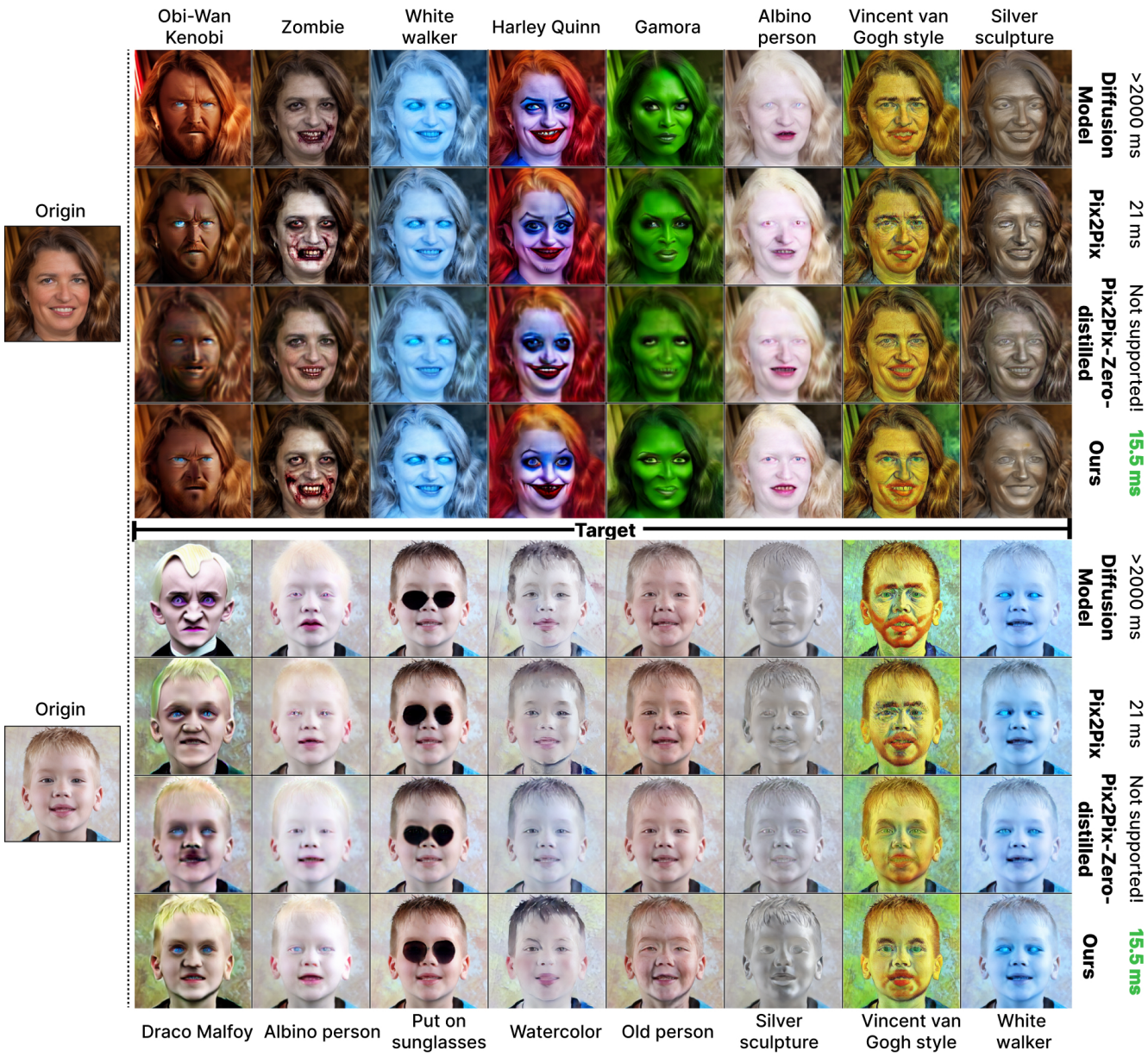

Figure 10. **Qualitative comparisons** on various tasks. The *leftmost* column shows two original images and the remaining columns present the corresponding synthesized images in the target concept domain, where target prompts are shown at the bottom row. We provide images generated by various models.

Figure 11. **Qualitative comparisons** on various tasks. The *leftmost* column shows two original images and the remaining columns present the corresponding synthesized images in the target concept domain, where target prompts are shown at the bottom row. We provide images generated by various models.

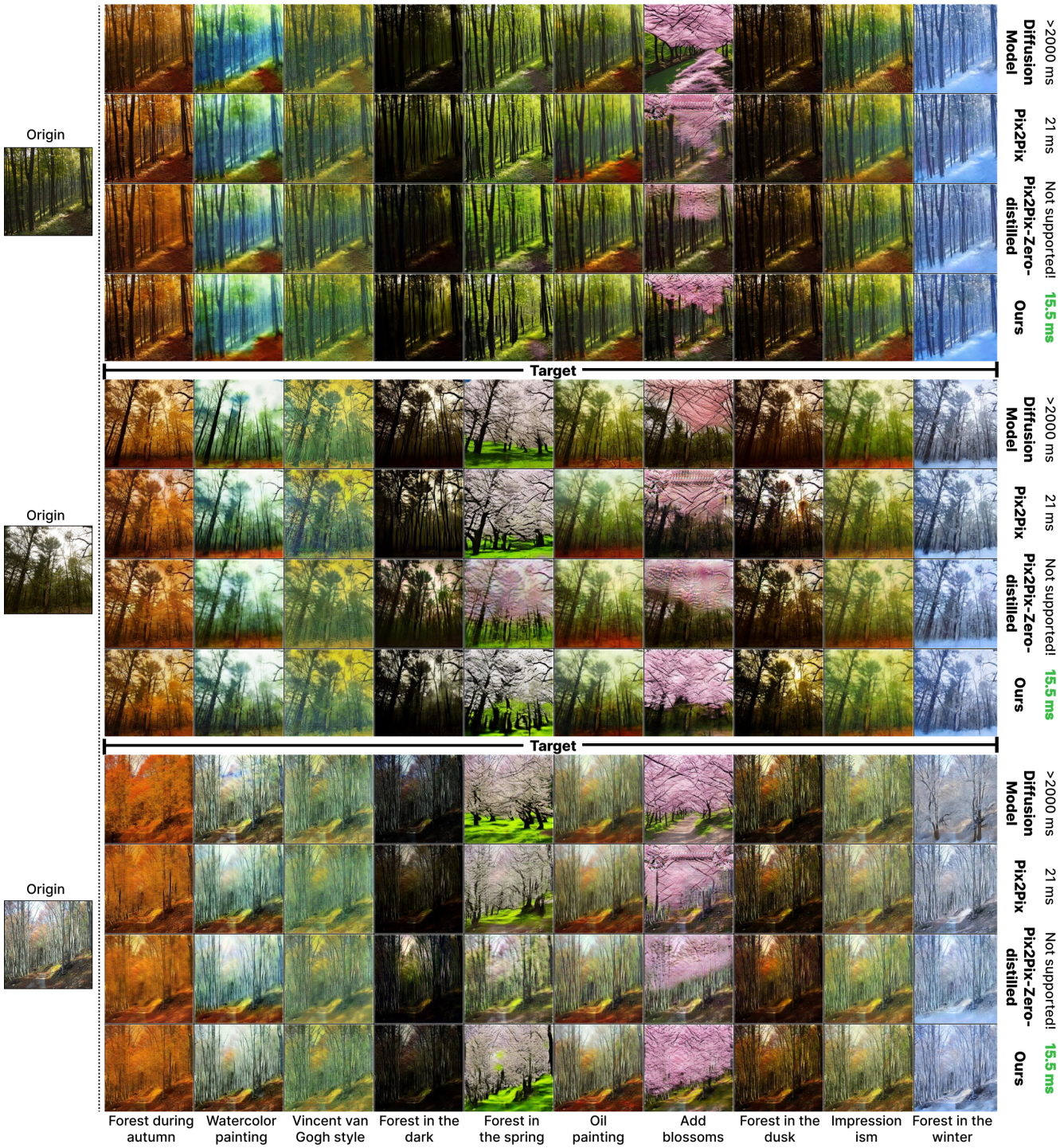

Figure 12. **Qualitative comparisons** on various tasks. The *leftmost* column shows two original images and the remaining columns present the corresponding synthesized images in the target concept domain, where target prompts are shown at the bottom row. We provide images generated by various models.