Efficient Training with Denoised Neural Weights

Yifan Gong[1,2]*, Zheng Zhan[2], Yanyu Li[1,2], Yerlan Idelbayev[1], Andrey Zharkov[1], Kfir Aberman[1], Sergey Tulyakov[1], Yanzhi Wang[2], and Jian Ren[1]

[1] Snap Inc. [2] Northeastern University
Project Page: https://yifanfanfanfan.github.io/denoised-weights/

Abstract. Good weight initialization serves as an effective measure to reduce the training cost of a deep neural network (DNN) model. The choice of how to initialize parameters is challenging and may require manual tuning, which can be time-consuming and prone to human error. To overcome such limitations, this work takes a novel step towards building a *weight generator* to synthesize the neural weights for initialization. We use the image-to-image translation task with generative adversarial networks (GANs) as an example due to the ease of collecting model weights spanning a wide range. Specifically, we first collect a dataset with various image editing concepts and their corresponding trained weights, which are later used for the training of the weight generator. To address the different characteristics among layers and the substantial number of weights to be predicted, we divide the weights into equal-sized blocks and assign each block an index. Subsequently, a diffusion model is trained with such a dataset using both text conditions of the concept and the block indexes. By initializing the image translation model with the denoised weights predicted by our diffusion model, the training requires only 43.3 seconds. Compared to training from scratch (*i.e.*, Pix2pix), we achieve a 15× training time acceleration for a new concept while obtaining even better image generation quality.

1 Introduction

Efficient training for deep neural networks (DNN) not only accelerates the model development process but also reduces the requirements for computational resources and costs. Many prior works have investigated efficient training strategies, such as sparse training [2, 9, 12, 21, 22, 35, 38, 40, 42] and low-bit training [34, 37, 41]. However, achieving efficient training is often hindered by challenges in initializing model weights effectively. While some efforts have been conducted in the domain of weight initialization [1, 6, 7, 10, 17, 43], determining the appropriate schemes to use across different tasks remains challenging. Tuning parameters for weight initialization can be time-consuming and prone to human error, leading to sub-optimal performance and increased training time.

---

* Work done during internship at Snap Inc.

To tackle such challenges, inspired by recent advances in designing Hyper-Networks [11, 30, 39], for the first time, we investigate the feasibility of building a *weight generator* to provide better weight initialization across different tasks, thus reducing the training time and resource consumption for obtaining a well-trained DNN model. We use image-to-image translation tasks trained with GAN models [18, 24, 46] as an example to unfold our design for predicting neural weights, though our framework is a general design that is not restricted to generating GAN weights. The reason for the choice is the easy acquisition of a vast volume of different weights trained on different datasets.

More specifically, the weight generator can predict the initialized weight for unseen new concepts and styles. To reduce the number of weights to be predicted, we apply Low-Rank Adaptation (LoRA) [16] to the image generation model, resulting in many fewer model parameters while still maintaining high-quality image generation. As the GAN model is composed of different types of layers with different sizes and number of weights, we group the weights and divide them into equal-sized blocks. A diffusion process [28, 31–33] is leveraged to model the space of well-trained weights from GAN models. Hence, the weights estimation by training diffusion model, namely, the weight generator, becomes possible. To improve the performance of the weight generator, we further incorporate the block index as a further conditioning mechanism in the weight generator, by adopting a sinusoidal positional encoding scheme and computing the embeddings for block indexes. The embedding provides the weight generator with information about the position of each weight block within all the model weights. After obtaining the weight generator, to train a GAN-based image translation model, we conduct a fast inference of the weight generator through a single-step denoising process and use the predicted weights to initialize the GAN model. The GAN model only requires a subsequent efficient fine-tuning process to obtain high-quality image generation results, significantly reducing the time consumption for obtaining the model for the newly unseen concept. We summarize our contributions as follows:

- We propose a framework to generate weight initializations across different concepts/styles to efficiently train the GAN models for the image translation.
- We collect a vast ground-truth dataset of LoRA weights for different concepts/styles with the help of diffusion models (*i.e.*, preparing paired image datasets), which serves as the foundation for weight generator training.
- We introduce an efficient design for a weight generator by utilizing a diffusion process, which incorporates both textual concept information and block indexes as inputs. To deal with different layer types and weight shapes, we organize weights into equal-sized one-dimensional blocks, significantly reducing computational overhead. These block indexes are seamlessly integrated into the weight generator design by combining them with time step embeddings. Therefore, the weight generator has the information about the position of each weight block within all the model weights.
- Our proposed framework can predict the initialized neural weights of a GAN model with a single denoising step, taking only 1.19 seconds. By initializing

with the predicted weights, a fast fine-tuning process can convey the target style in 42.1 seconds. Compared to training from scratch (*i.e.*, Pix2pix [18]), we reduce the total training time by 15×, while maintaining even *better* image generation quality. Compared to other efficient training method [14], we can save the training time by 4.6×.

2   Related Work

2.1   Efficient Training

Efficient training of DNNs has been a central point in machine learning research, aiming to reduce computational costs and memory requirements during model training while maintaining model performance. Sparse training methods [2, 9, 12, 21, 22, 35, 38, 40, 42] explore faster DNN training by applying sparse masks to the model. Static sparse training [22,35,38] executes traditional training after first pruning the model with a fixed sparse mask, which typically results in lower accuracy and higher computation and memory consumption for the pruning stage. On the contrary, dynamic mask methods [2, 9, 12] start with a sparse model structure from an untrained dense model and then combine sparse topology exploration with the sparse model training, which adjusts the sparsity topology during training while maintaining a low memory footprint. Besides applying sparse masks on weights and gradients, some recent works [21, 40, 42] also investigate incorporating data efficiency with different data selection approaches for better training accelerations. Meanwhile, another direction of the research explores low-bit training of DNNs to pursue model training efficiency [34, 37, 41]. However, using lower precision typically leads to an accuracy drop. A good weight initialization is essential to stabilize training, enable a higher learning rate, accelerate convergence, and improve generalization. Existing works explore rescaling paradigms [1,7,17,43] or leverage the relationship between layers [6,10]. However, determining schemes to use is still a challenging task and prone to human errors for different tasks. Meanwhile, LoRA methods [8, 16] aim to exploit the inherent low-rank structure present in DNN weight matrices to reduce computational complexity and memory requirements for fine-tuning from a pre-trained model on a specific, smaller dataset to specialize its performance on a particular task or domain. By keeping the original model unchanged and adds small, changeable parts to each layer of the model, LoRA methods can significantly reduce the number of parameters and operations required for forward and backward passes, which serves as an effective complement in the efficient training direction.

2.2   HyperNetwork

HyperNetworks have emerged as a promising approach in the field of generative AI by generating model parameters. HyperDreamBooth [30] introduces a HyperNetwork capable of generating personalized weights from a single image. By leveraging these weights within the diffusion model, HyperDreamBooth enables

the generation of personalized faces with high subject details in diverse styles and contexts followed by a fast fine-tuning process. HyperDiffusion [11] operates directly on MLP weight instead of directly applying generative modeling on implicit neural fields. It first collects a dataset of neural field multilayer perceptrons (MLPs) overfitting on 3D or 4D shapes. The dataset is then used to train the HyperNetwork, which is an unconditional generative model, to predict the MLP weights for 3D or 4D shape generation. Neural Network Diffusion [39] works on generating model weights for image classification tasks. However, the weight generation is based on first collecting multiple trained model weights for specific model architecture on the target dataset. Furthermore, it only works on generating the weights for two normalization layers.

3   Motivations and Challenges

Effective weight initialization is crucial for stabilizing training, facilitating a faster learning rate, expediting convergence, and enhancing generalization ability. However, identifying good weight initializations across different tasks remains challenging. Inspired by recent advances in HyperNetwork, we hope to investigate whether we can build a *weight generator* to obtain good weight initialization, thus reducing training time and resource consumption. Unlike the popular image/video generation, little research effort has been paid to explore weight generation. Building such a weight generator is promising yet challenging. The first significant challenge comes from the different layer types within DNN architectures. The weights in each layer exhibit diverse sizes and shapes, necessitating a weight generation approach capable of accommodating this heterogeneity. Second, the weight generator must possess the capacity to generate a substantial number of parameters efficiently, ensuring comprehensive coverage across the network. Third, the inference of the weight generator should be fast and efficient to save time in obtaining the weights for a new task. Addressing these challenges holds promise for building better DNN training paradigms with higher efficiency and effectiveness of deep learning systems. Thus, in this work, we study the construction of the weight generator for better weight initializations. We aim to show the generation ability not only restricted to the weight initialization for a single model architecture on a certain dataset, such as ResNet-18 on CIFAR-10 as in [39], but across the models for different tasks. To achieve this, we take the generation of initialization weights for GANs for image-to-image translation tasks as an example to show our methods due to the ease of collecting diverse datasets for the GAN models. Our method is not restricted to the GAN architecture or the image-to-image translation task.

4   Method

Our objective is to train a weight generator to predict the weight initializations for different tasks. We take GANs for image-to-image translation tasks as an example to demonstrate the effectiveness of our method. When there is a new
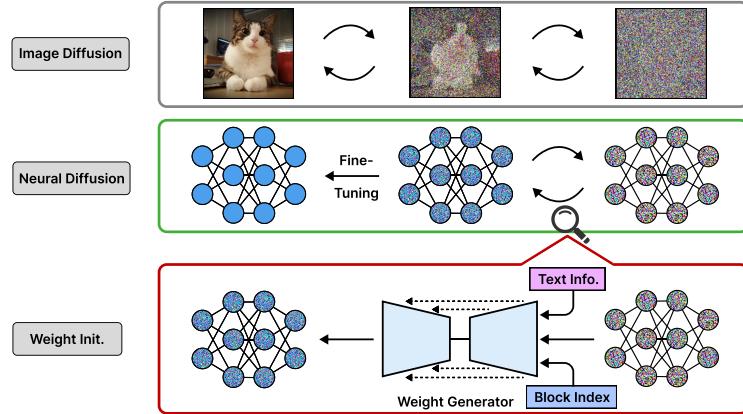
Fig. 1: The **framework overview** of our weight generator design. The standard diffusion process turns an image into noise in the forward pass and reverses a clean image from pure noise in the reverse process. Our weight generator is designed to turn a noise to weight initializations for efficient training purposes. Given the text information and block index, the weight generator provides the corresponding weight values.

concept/style, we can query the weight generator to provide the weight values for the initialization. The weight generator is modeled with a diffusion process, as illustrated in Fig. 1. Different from image diffusion models that reverse a clean image from a purse noise, our framework targets turning the noise into weight values used for the initialization. By plugging in the predicted weight values, a fast fine-tuning process is conducted to achieve the efficient training of the GAN models for the target style. The core of our framework is the design of the weight generator. To build this weight generator, we elaborate on how to create the training dataset for the weight generator in Sec. 4.1, the data format for the training and inference of the weight generator in Sec. 4.2, the architecture and training objective of the weight generator in Sec. 4.3, and the fine-tuning process after the weight prediction is Sec. 4.4.

4.1 Dataset Collection

In order to effectively train a weight generator for generating weight initializations of GAN models across various concepts, we need to collect a large-scale ground-truth weight value dataset for different concepts. To obtain the ground-truth weight value dataset, a large-scale prompt dataset becomes crucial. By using the concepts/styles in the prompt dataset, we can achieve image collection with diffusion models to obtain a substantial collection of images representative of each target concept. The images for each concept/style are further leveraged to train the GANs for the obtaining of the ground-truth GAN weights.

As the foundation of data preparation for weight generator training, the prompt dataset should include diverse visual concepts/styles to enable the weight generator to learn comprehensive representations for initializing GANs tailored

to specific tasks. However, the process of collecting such a dataset poses great challenges. Ensuring diversity and representativeness across different concepts/styles demands considerable data. Moreover, the collected prompts are further used to generate images in the target concept/style with diffusion models.

To construct our prompt dataset for training a reliable weight generator for GAN weight initialization, we adopt a systematic approach that integrates both large language models (LLMs) for style **generation** and **augmentation** to ensure richness and diversity in conceptual representation. We begin by sketching out three broad categories: 1) art concepts, 2) characteristic concepts, and 3) facial modification concepts. Within each category, we leverage a large language model (ChatGPT-3.5 [4]) to ask it to generate a spectrum of textual descriptions encompassing various concepts. By filtering out redundant concepts/styles, we further conduct an augmentation method by querying a large language model (Vicuna [5]) to provide concepts/styles with similar meanings but different representations. To further enrich the prompt dataset, we permute and combine concepts/styles across different categories. Through the process, we are able to curate a large-scale prompt dataset that not only spans diverse conceptual domains but also captures intricate stylistic differences, providing the foundation for the training of the weight generator for better weight initialization.

After the prompt dataset collection, we use the diffusion models to edit real images to obtain the edited images for each concept/style in the prompt dataset, forming pairs of data for GAN training. Here, we adopt a generator with a hybrid of ResNet blocks and transformer blocks as in paper [14] due to the effectiveness of the model and the hybrid architecture design to show the generation ability of our method on different types of layers. Following the GAN training process, we build a dataset of weights from GAN checkpoints for different concepts/styles. To further augment the weight value dataset, we save K checkpoints through the training process for each concept/style after the FID performance converges.

4.2   Data Format Design for Weight Generator

To train a weight generator capable of efficiently producing weight initializations for GAN models across diverse concepts, it is important to design the weight format for both training and inference. The objective is whenever a new concept is provided as the input to the weight generator, it can generate the weight initialization of all layers for the concept. Given there exist multiple different types of layers within the model such as fully connected (FC), convolutional (CONV), and batch normalization (BN) layers, and the varying sizes and dimensions across the layers, designing the appropriate data format becomes crucial and challenging. Furthermore, the scale of weights in a GAN model is typically on the scale of millions, posing more challenges to the data format design.

A larger amount of weights to be predicted leads to more difficulties for the weight generator. To alleviate this, we apply Low-Rank Adaptation (LoRA) [16] to different layers to greatly reduce the number of weights to be predicted. For instance, for a CONV layer i with weights $\mathbf{w}_i \in \mathbb{R}^{c \times f \times k_h \times k_w}$, we apply two low-rank matrices with rank r_i, $i.e.$ $\mathbf{w}_i^A \in \mathbb{R}^{c \times r_i \times k_h \times k_w}$ as LoRA down layer,

and $\mathbf{w}_i^B \in \mathbb{R}^{r_i \times f \times 1 \times 1}$ as LoRA up layer, to approximate the weight change. By doing so, the total amount of weights to be predicted is reduced from 7.06M to 0.22M. We show that finetuning LoRA weights are sufficient to transfer the generative domain of the GAN model. Though greatly reducing the weight number, directly predicting all 0.22M weights simultaneously through inference of the weight generator once is still challenging. It requires a large weight generator with huge computation and memory burdens.

To tackle this, we partition the weights into groups to mitigate the computational complexity and enhance the feasibility of fitting the weight generator into memory during both training and inference. As different layers have different statistical characteristics, we group the LoRA down and up layers for each layer i, with the associated BN layers if applicable, into one group. Still, each group has a different number of weights and shapes. Thus, we further flatten the weights into 1-dimensional vectors and divide the weights into N equal-sized blocks, each with b weights. Thus, the data format is denoted as $< n, \mathbf{w}_n, T >$, where n is the block index, $\mathbf{w}_n \in \mathbb{R}^b$ is the flattened 1-dimensional weight vector for the n-th weight block, and T denotes the text prompt of current concept/style. The advantages of using such a data format include: 1) works for different types and shapes of layers; 2) reduces the computation complexity and difficulty for prediction; and 3) makes the weight generator easier to fit into memory.

4.3 Weight Generator Training

Using our dataset of weight values, we train a generative model that learns to provide the weight initializations for other concepts/styles. We model the weight initialization space of GANs through a diffusion process. The generator is a UNet weight information creator $\hat{\epsilon}_\theta$ parameterized by θ for 1-dimensional vectors, which is demonstrated in Fig. 2. We diffuse the weight block $\mathbf{w}_n$ from a real weight distribution $p(\mathbf{w}_n)$ into a noisy version and train the denoising UNet to gradually reverse this process, generating weights from Gaussian noise. The training can be formulated as the following noise prediction problem:
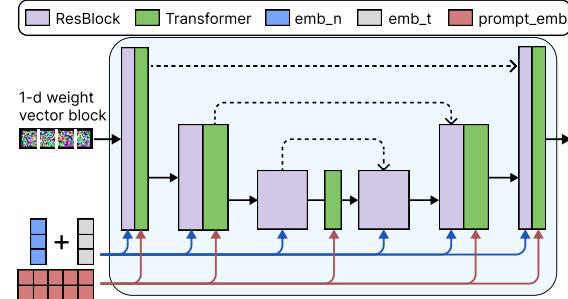

Fig. 2: The UNet Weight Generator. The weight generator is composed of 1-d ResBlocks and 1-d Transformer blocks. The block embedding emb_n is combined with the time step embedding emd_t to be leveraged in each ResBlock.

$$\min_\theta \mathbb{E}[\|\hat{\epsilon}_\theta(\mathbf{w}_n^t, t, n, \tau(T)) - \epsilon\|_2^2], \tag{1}$$

where t refers to the time step; ϵ is the ground-truth noise; $\mathbf{w}_n^t = \alpha_t \mathbf{w}_n + \sigma_t \epsilon$ is the noisy weight for block n; α_t and σ_t are the strengths of signal and noise,

respectively, decided by a noise scheduler; τ is a frozen text encoder such as CLIP [27]. To incorporate the block index as a further conditioning mechanism in our weight generator, we adopt a strategy inspired by the sinusoidal positional encoding commonly used in sequence-to-sequence models [36]. We compute a sinusoidal block index encoding, which serves to provide the weight generator with information about the position of each weight block within all the model weights. Specifically, let N denote the total number of weight blocks and d denote the dimensionality of the encoding. The sinusoidal block index encoding SinEnc(n, d) for block index n is computed as follows:

$$\text{SinEnc}(n, 2i) = \sin\left(\frac{n}{10000^{2i/d}}\right), \text{SinEnc}(n, 2i+1) = \cos\left(\frac{n}{10000^{2i/d}}\right), \quad (2)$$

where i ranges from 0 to $\lfloor\frac{d-1}{2}\rfloor$. The sinusoidal encoding is then fed into embedding layers to obtain the block index embedding emb_n. Finally, the block index embedding emb_n is combined with the time step embedding emd_t, represented as $emb = emb\_n + emb\_t$, to be leveraged in each residual block in the generator. Thus, the weight generator has access to the block index throughout the denoising process. From the results, we observe that the block index n can model the weights from different blocks effectively without the necessity to condition on prior predicted weights, while greatly reducing the computations.

4.4   Fast Fine-Tuning with Generated Weight Initializations

When a new concept/style T arises, the weight initializations can be obtained by conducting inference for the trained weight generator $\hat{\epsilon}_\theta$ for each weight block n. To achieve fast acquisition of weight initializations, we employ a direct reconstruction method to avoid the iterative denoising process. More specifically, at the selected time step t that leans to the noise side, we forward the denoising diffusion model to predict the noise $\hat{\epsilon}_\theta(\mathbf{w}_n^t, t, n, \tau(T))$, and we conduct a direct recovery to obtain the real weight $\mathbf{w}_n = \mathbf{w}_n^0$:

$$\mathbf{w}_n^0 = \frac{1}{\alpha_t}\mathbf{w}_n^t - \sigma_t\hat{\epsilon}_\theta(\mathbf{w}_n^t, t, n, \tau(T)). \tag{3}$$

After conducting inference for all of the N weight blocks, we can obtain the weight initialization $\{\mathbf{w}_n\}_{n=1}^N$ for the concept/style T. To capture the details of the new concept/style better, a further fine-tuning process for the GAN weights is leveraged with the conditional GAN loss as follows

$$\min_{\mathbf{w}_{lora}} \max_{\mathbf{w}_d} \lambda \underbrace{\mathbb{E}_{\mathbf{x}, \tilde{\mathbf{x}}^T, \mathbf{z}, T}\left[\|\tilde{\mathbf{x}}^T - \mathcal{G}(\mathbf{x}, \mathbf{z}, T; \mathbf{w}_g, \mathbf{w}_{lora})\|_1\right]}_{\ell_1 \text{ loss}} +$$
$$\underbrace{\mathbb{E}_{\mathbf{x}, \tilde{\mathbf{x}}^T}\left[\log \mathcal{D}(\mathbf{x}, \tilde{\mathbf{x}}^T; \mathbf{w}_d)\right] + \mathbb{E}_{\mathbf{x}, \mathbf{z}, T}\left[\log(1 - \mathcal{D}(\mathbf{x}, \mathcal{G}(\mathbf{x}, \mathbf{z}, T; \mathbf{w}_g); \mathbf{w}_d))\right]}_{\text{conditional GAN loss}}, \tag{4}$$

where $\tilde{\mathbf{x}}^T$ denotes images generated by the diffusion model conditioned on the concept T of the target style, $\mathcal{G}$ is the generator with original weights $\mathbf{w}_g$ and

the LoRA weights $\mathbf{w}_{lora}$, $\mathcal{D}$ denotes the discriminator function parameterized by $\mathbf{w}_d$, 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 two loss terms. During the fine-tuning process, the generator only optimizes the LoRA weights $\mathbf{w}_{lora}$ which are initialized with the predictions $\{\mathbf{w}_n\}_{n=1}^{N}$. By initializing the GAN weights from predictions, we are able to use much fewer training epochs to reach the same or even better FID performance. Besides fine-tuning after prediction, we also consider incorporating the GAN training loss in Eq. (4) to the weight prediction loss in Eq. (1). However, through experiments, we find out that combining these two loss terms is not able to provide better performance, but leads to more computation costs for training the weight generator.

5   Experiments

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

5.1   Experiment Settings

Baselines. We compare our method with image-to-image translation methods like pix2pix [18] (image generator with 9 ResNet blocks), pix2pix-zero-distilled that distills Co-Mod-GAN [45] from pix2pix-zero [25], and efficient GAN training methods $\mathrm{E}^2\mathrm{GAN}$ [14].

Prompt Dataset Preparation. We first use ChatGPT-3.5 [4] to collect prompts for the three categories: 1) art, 2) characteristic, and 3) facial modification concepts as discussed in Sec. 4.1. After filtering out repeated and unmeaning prompts, we get 226 art concepts, 441 character concepts, and 26 facial modification concepts. We reserve 20 art concepts, 20 character concepts, and 5 facial modification concepts for test use, never used during the weight generator training. By combining the concepts across different categories that are not reserved as test concepts and filtering, the prompt dataset is enriched with another 84477 concepts. We further augment the obtained concepts with Vicuna [5] for concepts with the same meaning but different expressions and filter meaningless ones, which leads to an additional 4126 augmented art concepts, 8070 augmented characteristic concepts, and 245 augmented facial modification concepts.

Paired Image Preparation. After the prompt dataset is collected, we generate images for GAN training for each concept. We verify our method on $1,000$ images from the FFHQ dataset [19] 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 [29], Instruct-Pix2Pix [3], Null-Text
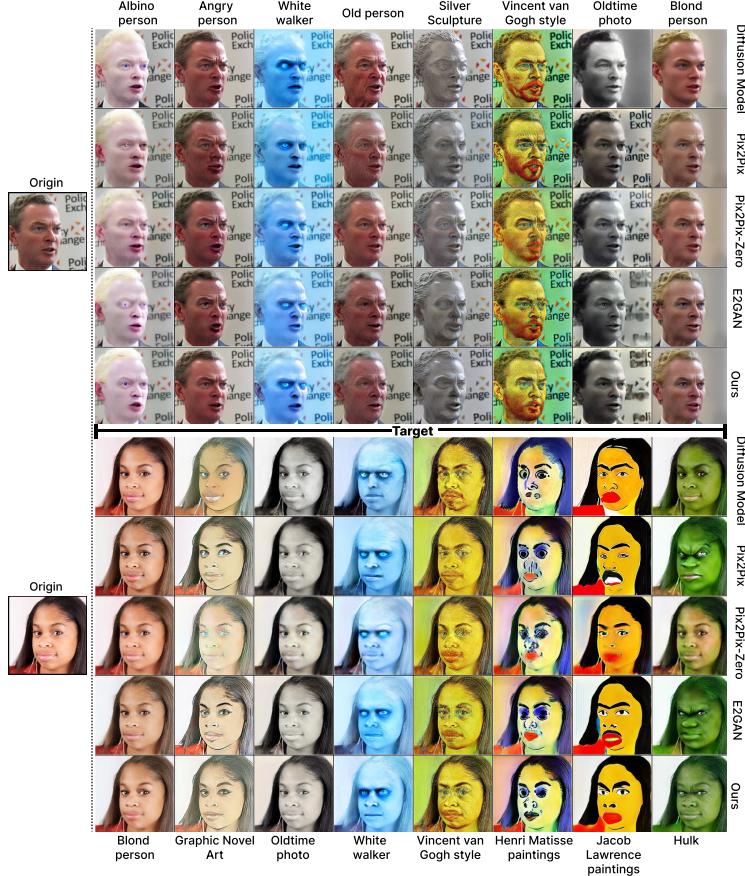
Fig. 3: Qualitative comparisons across different concept domains. The *leftmost* column shows two original images and the remaining columns present the corresponding synthesized images in the target concept domain, where target prompts are shown at the top/bottom row. We provide images generated by various models.

Inversion [23], ControlNet [44], and InstructDiffusion [13]. The generated images with the best perceptual quality among diffusion models are selected to form 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%.

Ground-truth GAN Weights Preparation. With the paired images, we collect the ground-truth GAN weights to train the weight generator. We apply LoRA to each layer of the generator, leading to 0.2256M weights for each concept to learn. We follow the standard approach that alternatively updates the generator and discriminator [15]. The training is conducted from an initial learning rate of 2e-4 with mini-batch SGD using Adam solver [20]. The total training

epochs are set to 100. The obtained weights are grouped following Sec. 4.2 and divided with a block size of 256, resulting in a total of 854 weight blocks. When dividing the weights into equal-sized blocks, zeros are padded when necessary.

Training Settings. The weight generator is trained with AdamW optimizer [20]. The initial learning rate is set as 1e-5, the weight decay is set as 0.01, and the training batch size is set as 512. The weight generator training is conducted with 4 or 8 nodes, each with 8 NVIDIA A100 GPUs with 40GB or 80GB memory. The block size for weight division is set as 256. For the following fine-tuning process, to show a fair comparison with the efficient GAN training approach E^2GAN, we adopt the same cluster size as 400 for each concept. The initial learning rate is set as 0.0015 and the fine-tuning epochs are set as 20.

Evaluation Metric. We compare the performance of our efficient weight generalization by comparing images generated by models obtained via our approach and baseline methods. The evaluation is achieved by calculating Clean FID proposed by [26] on the test sets of the paired images.

5.2 Experimental Results

Qualitative Results. The synthesized images in the target domain obtained by our method and other methods are shown in Fig. 3. The original images are listed in the leftmost column, and the synthesized images for the target concept obtained by diffusion models, pix2pix, pix2pix-zero-distilled, E^2GAN, and ours are shown from top to bottom. The tasks span a wide range, such as changing the age, artistic styles, and characteristic styles. According to the results, the models obtained by ours can modify the original images to the target concept domain by fast fine-tuning with the weight initializations from the weight generator. For instance, for the `Jacob Lawrence paintings` prompt on the FFHQ dataset, our model generates more meaningful images compared to all baseline methods. As for the `albino person` prompt, our method edits the image as desired while having fewer artifacts. We provide more qualitative results in the Appendix.

Quantitative Results. We compare the quantitive results and training time consumption between our method and other baseline methods, and the results are provided in Tab. 1. Note that for each concept, pix2pix-zero-distilled and pix2pix are trained on the whole training dataset of 800 samples

Table 1: **FID and time consumption comparison.** FID is calculated between the images generated by GAN-based approaches and diffusion models. Reported FID is averaged across different concepts in the test prompt dataset.

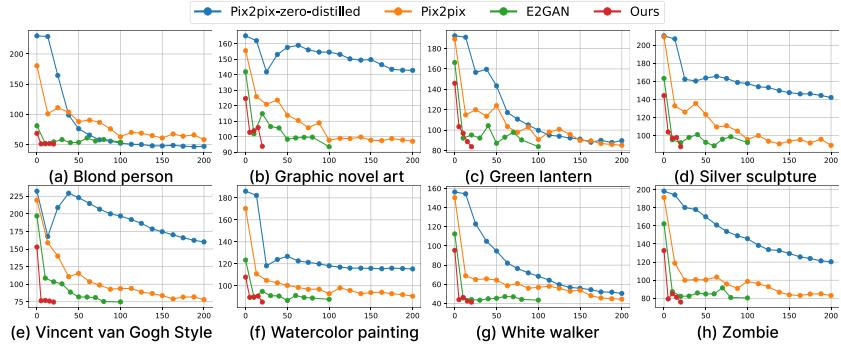| Method	FID	Time Consumption
Pix2pix-zero-distilled	144.81	112 min
Pix2pix	99.20	659.8 secs
E²GAN	93.86	198.5 secs
Ours	**89.93**	**43.3 secs**

Fig. 4: The FID performance comparison between our method and baseline methods along with the training process on the test dataset for different concepts/styles.

with 200 epochs. E^{2}GAN begins with a base model and is fine-tuned with 400 samples for 100 epochs. Our method initializes the GAN weights with the prediction from the weight generator and is fine-tuned with 400 samples for 20 epochs. The reported FID values are computed with an average for all the concepts in the test prompt dataset. The time consumption is measured on one NVIDIA H100 GPU. The results demonstrate that our method can reach an even better FID performance than the conventional GAN training techniques like pix2pix and pix2pix-zero-distilled, and efficient GAN training methods such as E^{2}GAN, indicating the high-fidelity of generated images. Furthermore, our method greatly reduces the time consumption for obtaining the model for a new concept/style. The time consumption of our method is mainly composed of two parts: 1) the inference of the weight generator to get the predicted weight initialization values for each block; and 2) the fine-tuning process with the initialized weight values. By using a batch size of 64, **the prediction process of the weight generator only takes 1.19 secs**. Taking advantage of the effective weight initializations, the fine-tuning process only requires 20 epochs to reach good performance, which takes 42.1 secs and results in 43.3 secs of training time in total. Specifically, our method reduces the training time by 152×, 15×, and 4.6× compared to Pix2pix-zero-distilled, Pix2pix, and E^{2}GAN, respectively. Furthermore, the results also indicate the effectiveness of only updating LoRA weights to transfer the generative domain of the GAN model as both our method and E^{2}GAN only update LoRA weights, but achieve good FID performance.

FID Curves. We show the FID curves along with the training process by different methods in Fig. 4. More FID curve results can be found in the Appendix. From the results, we can observe that our method achieves a faster convergence across different concepts/styles. The effectiveness of our weight generator for generating weight initializations is revealed in two aspects. First, our method provides a better starting point for the training, indicated by the better FID score before training (epoch 0). Second, training from the initialization can reach

a better FID score with much fewer training epochs. For instance, for the concept `white walker`, our weight generator provides a weight initialization with an FID of 95.46, while the baseline methods have an initial FID of 154.29, 150.32, and 112.50, respectively. After fine-tuning with 20 epochs, our method reaches an FID of 41.38, which improves the baseline methods by at least 2.1.

Ablation Studies. We conduct ablation studies on the **block size** and **weight grouping** rules. Due to the huge training cost of the weight generator on the entire training dataset, we conduct small-scale experiments for the ablation study. We overfit the weight generator solely on the weights corresponding to a particular concept of interest. The approach provides a precise assessment of how well a particular configuration captures a specific concept. Any discrepancies between the overfitted results and the ground-truth values can be attributed directly to the efficacy of the chosen configuration.

For the block size study, we investigate different block size settings including 128, 256, and 512, on two randomly selected concepts `grey hair` and `Batik`. The block size selection is based on the size of all layers in the GAN model. The results are shown in Tab. 2. The results show that the block size selection has an impact on the weight generation performance. Setting a larger block size leads to a faster weight generation process. However, the FID performance is the best when the block size is set as 256, while the generation time is slightly slower than a block size of 512. The results indicate that the appropriate selection of the block size to divide grouped weights is important for achieving good performance of the weight generator.

Table 2: Ablation study on block size for weight division in data preparation. The first row represents the performance of the ground-truth trained model while the remaining rows correspond to the performance of generated weights from the weight generator with different weight block sizes.

Block Size	Weight Gene. Time	FID	
		Grey hair	Batik
-	-	89.04	90.43
128	1.71 secs	107.67	128.44
256	**1.19 secs**	**98.35**	**94.36**
512	1.04 secs	130.85	118.31

For the weight grouping before weight division, we study 3 different rules including 1) group the LoRA down layer, LoRA up layer, and the following BN layer if applicable for each layer i to one group, and append the reshaped 1-dimensional weight vectors one by one; 2) group the LoRA down layer, LoRA up layer, and the following BN layer if applicable for each layer i to one group, concatenate the weights through the channel dimension and reshape it to 1-dimensional vector; 3) view each layer as a single group and reshape to 1-dimensional vector. Besides the weight grouping rules, all the other settings are the same as the default setting for the main results. We show the comparisons of the 3 rules in Tab. 3. From the results, we can see

Table 3: Ablation study on weight grouping.

Group Rule	Grey hair	Batik
Rule 1)	**98.35**	**94.36**
Rule 2)	98.40	95.17
Rule 3)	122.71	126.50

that Rule 1) and Rule 2) both perform much better than Rule 3), which indicates the importance of grouping the layers belonging to the same layer i into one group. The rationale behind this might be related to the different statistics among different layers. Furthermore, Rule 1) and Rule 2) do not have obvious performance differences, which means after grouping weights for the same layer i together, it is not necessary to take different channels separately. Combining the ablation studies, we get the default setting used in the main results, which correspond to use rule 1) to group weights and set the block size as 256.

6   Conclusion

This paper studies to generate good weight initializations with a weight generator to reduce the training cost of a DNN. Leveraging the image-to-image translation task with GANs as a case study, we demonstrate the feasibility and effectiveness of our approach. Through the division of weights into equal-sized blocks and the incorporation of block indexes, we mitigate the complexity of varied layer characteristics and a large number of weights. By training a diffusion process with both textual concept conditions and block indexes, the weight generator produces weight initializations for new concepts/styles efficiently with a one-step direct recovery. We conduct extensive experiments on different concepts to demonstrate the effectiveness of our proposed framework. By leveraging the synthesized weight initializations, we can achieve better FID performance with much fewer training costs across various concepts/styles than baseline methods including conventional GAN training and efficient GAN training approaches. We reduce the time consumption for obtaining the model for a new concept by $4.6\times$ while improving the FID performance by 3.93 than efficient GAN training baseline and reduce the total training time by $15\times$ than training from scratch (*i.e.*, Pix2pix [18]) with better FID performance.

7   Discussion of limitations

The development of a weight generator to synthesize improved weight initializations can increase the efficiency and efficacy of model training. To prepare the training data for the weight generator, we leverage diffusion models to edit real images, thereby obtaining edited images that encompass a wide range of concepts. This approach allows us to create paired data spanning various concepts/styles, which provide the foundation for training diverse GAN weights for different generation domains. However, the quality of the generated images plays a pivotal role in influencing the performance of the trained GAN model, consequently impacting the performance of the weight generator. While diffusion models offer a powerful tool for image editing, the quality and fidelity of the generated images may not always meet the desired standards. Furthermore, utilizing diffusion models for data collection remains expensive. Developing efficient techniques to rapidly construct well-paired and high-quality images from diffusion models would greatly enhance the training of the weight generator.

References

1. Bachlechner, T., Majumder, B.P., Mao, H., Cottrell, G., McAuley, J.: Rezero is all you need: Fast convergence at large depth. In: Uncertainty in Artificial Intelligence. pp. 1352–1361. PMLR (2021) 1, 3
2. Bellec, G., Kappel, D., Maass, W., Legenstein, R.: Deep rewiring: Training very sparse deep networks. arXiv preprint arXiv:1711.05136 (2017) 1, 3
3. Brooks, T., Holynski, A., Efros, A.A.: Instructpix2pix: Learning to follow image editing instructions. arXiv preprint arXiv:2211.09800 (2022) 9
4. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020) 6, 9
5. Chiang, W.L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J.E., et al.: Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. See https://vicuna. lmsys. org (accessed 14 April 2023) (2023) 6, 9
6. Cordonnier, J.B., Loukas, A., Jaggi, M.: On the relationship between self-attention and convolutional layers. arXiv preprint arXiv:1911.03584 (2019) 1, 3
7. De, S., Smith, S.: Batch normalization biases residual blocks towards the identity function in deep networks. Advances in Neural Information Processing Systems **33**, 19964–19975 (2020) 1, 3
8. Dettmers, T., Pagnoni, A., Holtzman, A., Zettlemoyer, L.: Qlora: Efficient finetuning of quantized llms. Advances in Neural Information Processing Systems **36** (2024) 3
9. Dettmers, T., Zettlemoyer, L.: Sparse networks from scratch: Faster training without losing performance. arXiv preprint arXiv:1907.04840 (2019) 1, 3
10. d'Ascoli, S., Touvron, H., Leavitt, M.L., Morcos, A.S., Biroli, G., Sagun, L.: Convit: Improving vision transformers with soft convolutional inductive biases. In: International Conference on Machine Learning. pp. 2286–2296. PMLR (2021) 1, 3
11. Erkoç, Z., Ma, F., Shan, Q., Nießner, M., Dai, A.: Hyperdiffusion: Generating implicit neural fields with weight-space diffusion. arXiv preprint arXiv:2303.17015 (2023) 2, 4
12. Evci, U., Gale, T., Menick, J., Castro, P.S., Elsen, E.: Rigging the lottery: Making all tickets winners. In: International Conference on Machine Learning. pp. 2943–2952. PMLR (2020) 1, 3
13. Geng, Z., Yang, B., Hang, T., Li, C., Gu, S., Zhang, T., Bao, J., Zhang, Z., Hu, H., Chen, D., et al.: Instructdiffusion: A generalist modeling interface for vision tasks. arXiv preprint arXiv:2309.03895 (2023) 10
14. Gong, Y., Zhan, Z., Jin, Q., Li, Y., Idelbayev, Y., Liu, X., Zharkov, A., Aberman, K., Tulyakov, S., Wang, Y., et al.: E2gan: Efficient training of efficient gans for image-to-image translation. arXiv preprint arXiv:2401.06127 (2024) 3, 6, 9
15. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. Communications of the ACM **63**(11), 139–144 (2020) 10
16. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021) 2, 3, 6
17. Huang, X.S., Perez, F., Ba, J., Volkovs, M.: Improving transformer optimization through better initialization. In: International Conference on Machine Learning. pp. 4475–4483. PMLR (2020) 1, 3

18. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1125–1134 (2017) 2, 3, 9, 14
19. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4401–4410 (2019) 9
20. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 10, 11
21. Kong, Z., Ma, H., Yuan, G., Sun, M., Xie, Y., Dong, P., Meng, X., Shen, X., Tang, H., Qin, M., et al.: Peeling the onion: Hierarchical reduction of data redundancy for efficient vision transformer training. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 8360–8368 (2023) 1, 3
22. Lee, N., Ajanthan, T., Torr, P.H.: Snip: Single-shot network pruning based on connection sensitivity. arXiv preprint arXiv:1810.02340 (2018) 1, 3
23. Mokady, R., Hertz, A., Aberman, K., Pritch, Y., Cohen-Or, D.: Null-text inversion for editing real images using guided diffusion models. arXiv preprint arXiv:2211.09794 (2022) 10
24. Park, T., Efros, A.A., Zhang, R., Zhu, J.Y.: 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. pp. 319–345. Springer (2020) 2
25. Parmar, G., Kumar Singh, K., Zhang, R., Li, Y., Lu, J., Zhu, J.Y.: Zero-shot image-to-image translation. In: ACM SIGGRAPH 2023 Conference Proceedings. pp. 1–11 (2023) 9
26. Parmar, G., Zhang, R., Zhu, J.Y.: On aliased resizing and surprising subtleties in gan evaluation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11410–11420 (2022) 11
27. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PMLR (2021) 8
28. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10684–10695 (2022) 2
29. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10684–10695 (2022) 9
30. Ruiz, N., Li, Y., Jampani, V., Wei, W., Hou, T., Pritch, Y., Wadhwa, N., Rubinstein, M., Aberman, K.: Hyperdreambooth: Hypernetworks for fast personalization of text-to-image models. arXiv preprint arXiv:2307.06949 (2023) 2, 3
31. Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics. In: International conference on machine learning. pp. 2256–2265. PMLR (2015) 2
32. Song, Y., Ermon, S.: Generative modeling by estimating gradients of the data distribution. Advances in neural information processing systems **32** (2019) 2
33. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. arXiv preprint arXiv:2011.13456 (2020) 2

34. Sun, X., Wang, N., Chen, C.Y., Ni, J., Agrawal, A., Cui, X., Venkataramani, S., El Maghraoui, K., Srinivasan, V.V., Gopalakrishnan, K.: Ultra-low precision 4-bit training of deep neural networks. Advances in Neural Information Processing Systems **33**, 1796–1807 (2020) 1, 3

35. Tanaka, H., Kunin, D., Yamins, D.L., Ganguli, S.: Pruning neural networks without any data by iteratively conserving synaptic flow. Advances in neural information processing systems **33**, 6377–6389 (2020) 1, 3

36. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017) 8

37. Venkataramani, S., Srinivasan, V., Wang, W., Sen, S., Zhang, J., Agrawal, A., Kar, M., Jain, S., Mannari, A., Tran, H., et al.: Rapid: Ai accelerator for ultra-low precision training and inference. In: 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA). pp. 153–166. IEEE (2021) 1, 3

38. Wang, C., Zhang, G., Grosse, R.: Picking winning tickets before training by preserving gradient flow. arXiv preprint arXiv:2002.07376 (2020) 1, 3

39. Wang, K., Xu, Z., Zhou, Y., Zang, Z., Darrell, T., Liu, Z., You, Y.: Neural network diffusion (2024) 2, 4

40. Wang, Z., Zhan, Z., Gong, Y., Yuan, G., Niu, W., Jian, T., Ren, B., Ioannidis, S., Wang, Y., Dy, J.: Sparcl: Sparse continual learning on the edge. Advances in Neural Information Processing Systems **35**, 20366–20380 (2022) 1, 3

41. Wortsman, M., Dettmers, T., Zettlemoyer, L., Morcos, A., Farhadi, A., Schmidt, L.: Stable and low-precision training for large-scale vision-language models. Advances in Neural Information Processing Systems **36** (2024) 1, 3

42. Yuan, G., Ma, X., Niu, W., Li, Z., Kong, Z., Liu, N., Gong, Y., Zhan, Z., He, C., Jin, Q., et al.: Mest: Accurate and fast memory-economic sparse training framework on the edge. Advances in Neural Information Processing Systems **34**, 20838–20850 (2021) 1, 3

43. Zhang, H., Dauphin, Y.N., Ma, T.: Fixup initialization: Residual learning without normalization. arXiv preprint arXiv:1901.09321 (2019) 1, 3

44. Zhang, L., Agrawala, M.: Adding conditional control to text-to-image diffusion models (2023) 10

45. Zhao, S., Cui, J., Sheng, Y., Dong, Y., Liang, X., Chang, E.I., Xu, Y.: Large scale image completion via co-modulated generative adversarial networks. arXiv preprint arXiv:2103.10428 (2021) 9

46. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE international conference on computer vision. pp. 2223–2232 (2017) 2