DiffFit: Unlocking Transferability of Large Diffusion Models via Simple Parameter-Efficient Fine-Tuning

Enze Xie[1], Lewei Yao[1], Han Shi[1], Zhili Liu[1], Daquan Zhou[2], Zhaoqiang Liu[1], Jiawei Li[1], Zhenguo Li[1]
[1]Huawei Noah's Ark Lab [2]National University of Singapore

Figure 1: Selected samples to show parameter-efficient fine-tuned DiT-XL/2 model using DiffFit. DiffFit only needs to fine-tune 0.12% parameters. *Top row*: 512×512 image generation on ImageNet with **3.02 FID**. *Bottom rows*: 256×256 image generation on several downstream datasets across diverse domains: Food, Fungi, Scene, Art, Bird, Flower.

Abstract

Diffusion models have proven to be highly effective in generating high-quality images. However, adapting large pre-trained diffusion models to new domains remains an open challenge, which is critical for real-world applications. This paper proposes DiffFit, a parameter-efficient strategy to fine-tune large pre-trained diffusion models that enable fast adaptation to new domains. DiffFit is embarrassingly simple that only fine-tunes the bias term and newly-added scaling factors in specific layers, yet resulting in significant training speed-up and reduced model storage costs. Compared with full fine-tuning, DiffFit achieves 2× training speed-up and only needs to store approximately 0.12% of the total model parameters. Intuitive theoretical analysis has been provided to justify the efficacy of scaling factors on fast adaptation. On 8 downstream datasets, DiffFit achieves superior or competitive performances compared to the full fine-tuning while being more efficient. Remarkably, we show that DiffFit can adapt a pre-trained low-resolution generative model to a high-resolution one by adding minimal cost. Among diffusion-based methods, DiffFit sets a new state-of-the-art FID of 3.02 on ImageNet 512×512 benchmark by fine-tuning only 25 epochs from a public pre-trained ImageNet 256×256 checkpoint while being 30× more training efficient than the closest competitor.

Correspondence to {xie.enze, li.zhenguo}@huawei.com

1. Introduction

Denoising diffusion probabilistic models (DDPMs) [20, 51, 49] have recently emerged as a formidable technique for generative modeling and have demonstrated impressive results in image synthesis [43, 12, 45], video generation [21, 19, 65] and 3D editing [40]. However, the current state-of-the-art DDPMs suffer from significant computational expenses due to their large parameter sizes and numerous inference steps per image. For example, the recent *DALL· E 2* [45] comprises 4 separate diffusion models and requires 5.5B parameters. In practice, not all users are able to afford the necessary computational and storage resources. As such, there is a pressing need to explore methods for adapting publicly available, large, pre-trained diffusion models to suit specific tasks effectively. In light of this, a central challenge arises: *Can we devise an inexpensive method to fine-tune large pre-trained diffusion models efficiently?*

Take the recent popular Diffusion Transformer (DiT) as an example, the DiT-XL/2 model, which is the largest model in the DiT family and yields state-of-the-art generative performance on the ImageNet class-conditional generation benchmark. In detail, DiT-XL/2 comprises 640M parameters and involves computationally demanding training procedures. Our estimation indicates that the training process for DiT-XL/2 on 256×256 images necessitates 950 V100 GPU days (7M iterations), whereas the training on 512×512 images requires 1733 V100 GPU days (3M iterations). The high computational cost makes training DiT from scratch unaffordable for most users. Furthermore, extensive fine-tuning of the DiT on diverse downstream datasets requires storing multiple copies of the whole model, which results in linear storage expenditures.

In this paper, we propose DiffFit, a simple and parameter-efficient fine-tuning strategy for large diffusion models, building on the DiT as the base model. The motivation can be found in Figure 2. Recent work in natural language processing (BitFit [61]) has demonstrated that fine-tuning only the bias term in a pre-trained model performs sufficiently well on downstream tasks. We, therefore, seek to extend these efficient fine-tuning techniques to image generative tasks. We start with directly applying BitFit [61] and empirically observe that simply using the BitFit technique is a good baseline for adaptation. We then introduce learnable scaling factors γ to specific layers of the model, initialized to 1.0, and made dataset-specific to accommodate enhancement of feature scaling and results in better adaptation to new domains. Interestingly, the empirical findings show that incorporating γ at specific locations of the model is important to reaching a better FID score.

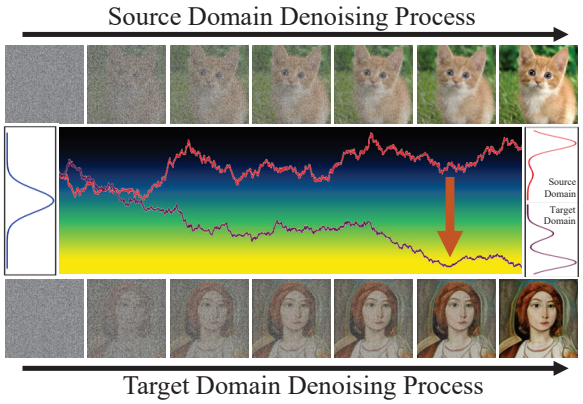

Figure 2: The denoising process of a diffusion model typically involves iteratively generating images from random noise. In DiffFit, the pre-trained large diffusion model in the source domain can be fine-tuned to adapt to a target domain with only a few specific parameter adjustments.

In other words, the FID score does not improve linearly with the number of γ included in the model. In addition, we conducted a theoretical analysis of the mechanism underlying the proposed DiffFit for fine-tuning large diffusion models. We provided intuitive theoretical analysis to help understand the effect of the newly-added scaling factors in the shift of distributions.

We employed several parameter-efficient fine-tuning techniques, including BitFit [61], AdaptFormer [7], LoRA [23], and VPT [24], and evaluated their performance on 8 downstream datasets. Our results demonstrate that DiffFit outperforms these methods regarding Frechet Inception Distance (FID) [38] trade-off and the number of trainable parameters. Furthermore, we surprisingly discovered that by treating high-resolution images as a special domain from low-resolution ones, our DiffFit approach could be seamlessly applied to fine-tune a low-resolution diffusion model, enabling it to adapt to high-resolution image generation at a minimal cost. For example, starting from a pre-trained ImageNet 256×256 checkpoint, by fine-tuning DIT for only 25 epochs (≈0.1M iterations), DiffFit surpassed the previous state-of-the-art diffusion models on the ImageNet 512×512 setting. Even though DiffFit has only about 0.9 million trainable parameters, it outperforms the original DiT-XL/2-512 model (which has 640M trainable parameters and 3M iterations) in terms of FID (3.02 *vs.* 3.04), while reducing 30× training time. In conclusion, DiffFit aims to establish a simple and strong baseline for parameter-efficient fine-tuning in image generation and shed light on the efficient fine-tuning of larger diffusion models.

Our contributions can be summarized as follows:

1. We propose a simple parameter-efficient fine-tuning approach for diffusion image generation named Diff-

Fit. It achieves superior results compared to full fine-tuning while leveraging only 0.12% trainable parameters. Quantitative evaluations across 8 downstream datasets demonstrate that DiffFit outperforms existing well-designed fine-tuning strategies (as shown in Figure 3 and Table 1).

2. We conduct an intuitive theoretical analysis and design detailed ablation studies to provide a deeper understanding of why this simple parameter-efficient fine-tuning strategy can fast adapt to new distributions.

3. We show that by treating high-resolution image generation as a downstream task of the low-resolution pre-trained generative model, DiffFit can be seamlessly extended to achieve superior generation results with FID 3.02 on ImageNet and reducing training time by 30 times, thereby demonstrating its scalability.

2. Related Works

2.1. Transformers in Vision

Transformer architecture was first introduced in language model [55] and became dominant because of its scalability, powerful performance and emerging ability [41, 42, 3]. Then, Vision Transformer (ViT) [13] and its variants achieved colossal success and gradually replaced ConvNets in various visual recognition tasks, *e.g.* image classification [54, 64, 60, 16], object detection [34, 56, 57, 4], semantic segmentation [63, 58, 52] and so on [53, 32, 62, 35, 17, 31]. Transformers are also widely adopted in GAN-based generative models [14, 25] and the conditional part of text-to-image diffusion models [45, 46, 43, 1]. Recently, DiT [39] proposed a plain Transformer architecture for the denoising portion of diffusion networks and verified its scaling properties. Our paper adopts DiT as a strong baseline and studies parameter-efficient fine-tuning.

2.2. Diffusion Models

Diffusion models [20] (aka. score-based models [50]) have shown great success in generative tasks, including density estimation [26], image synthesis [12], text-to-image generation [45, 1, 47] and so on. Different from previous generative models like GAN [8], VAE [27] and Flow [44], diffusion models [20] transform a data distribution to a Gaussian distribution by progressively adding noise, and then, reversing the process via denoising to retrieve the original distribution. The progressive step-by-step transformation between the two distributions makes the training process of diffusion models more stable compared to other models. However, the multiple time-step generations makes the diffusion process time-consuming and expensive.

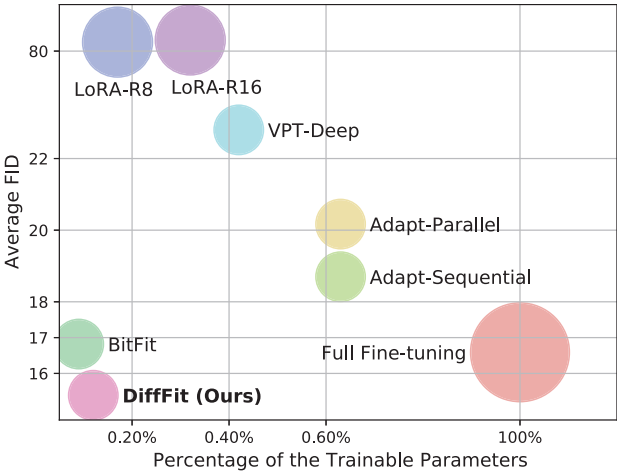

Figure 3: Average FID score of fine-tuned DiT across 8 downstream datasets. The radius of each bubble reflects the training time (smaller is better). We observe that DiffFit performs remarkably well in terms of achieving the best FID while requiring fewer computations and parameters.

2.3. Parameter-efficient Fine-tuning

Witnessing the success of Transformers in language and vision, many large models based on Transformer architecture have been developed and pre-trained on massive upstream data. On the one hand, the industry continues to increase the model parameters to billion, even trillion scales [3, 15, 11] to probe up the upper bound of large models. On the other hand, fine-tuning and storage of large models are expensive. There are three typical ways for parameter-efficient fine-tuning as follows:

1. Adaptor [22, 23, 7]. Adaptor is a small module inserted between Transformer layers, consisting of a down-projection, a nonlinear activation function, and an up-projection. Specifically, LoRA [23] adds two low-rank matrices to the query and value results between the self-attention sub-layer. AdaptFormer [7], however, places the trainable low-rank matrices after the feed-forward sub-layer.

2. Prompt Tuning [30, 29, 36, 24, 66]. Usually, prefix tuning [30] appends some tunable tokens before the input tokens in the self-attention module at each layer. In contrast, prompt-tuning [29] only appends the tunable tokens in the first layer for simplification. VPT [24] focuses on the computer vision field and proposes deep and shallow prompt tuning variants.

3. Partial Parameter Tuning [61, 59, 33]. Compared with the above parameter-efficient methods, partial parameter tuning does not insert any other components and only fine-tunes the partial parameters of the original model. For example, BitFit [61] tunes the bias of each linear projection and Child-Tuning [59] evaluates the importance of parame-

ters and tunes only the important ones.

3. Methodology

3.1. Preliminaries

Diffusion Models. Denoising diffusion probabilistic models (DDPMs) [20] define generative models by adding Gaussian noise gradually to data and then reversing back. Given a real data sample $\mathbf{x}_0 \sim q_{data}(\mathbf{x})$, the forward process is controlled by a Markov chain as $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$, where β_t is a variance schedule between 0 and 1. By using the reparameterization trick, we have $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. For larger time step t, we have smaller $\bar{\alpha}_t$, and the sample gets noisier.

As for the reverse process, DDPM learns a denoise neural network $p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$. The corresponding objective function is the following variational lower bound of the negative log-likelihood:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_t \mathcal{D}_{\mathrm{KL}}\left(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)\middle|p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)\right) - p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1), \quad (1)$$

where $\mathcal{D}_{\mathrm{KL}}(p|q)$ represents the KL divergence measuring the distance between two distributions p and q. Furthermore, the objective function can be reduced to $\mathcal{L}_{vlb} = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon},t}\left[\frac{\beta_t^2}{2\alpha_t(1-\bar{\alpha}_t)\sigma_t^2}\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\|^2\right]$ and a simple variant loss function $L_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon},t}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\|^2\right]$. Following iDDPM [37], we use a hybrid loss function as $\mathcal{L}_{hybrid} = \mathcal{L}_{simple} + \lambda\mathcal{L}_{vlb}$, where λ is set to be 0.001 in our experiments.

Diffusion Transformers (DiT). Transformer [55] architecture has proved to be powerful in image recognition, and its design can be migrated to diffusion models for image generation. DiT [39] is a recent representative method that designs a diffusion model with Transformers. DiT follows the design of latent diffusion models (LDMs) [45], which have two parts given a training sample $\mathbf{x}$: (1) An autoencoder consisting of an encoder E and a decoder D, where the latent code $\mathbf{z} = E(\mathbf{x})$ and the reconstructed data $\hat{\mathbf{x}} = D(\mathbf{z})$; (2) A latent diffusion transformer with patchify, sequential DiT blocks, and depatchify operation. In each block B_i, we have $\mathbf{z}_i = B_i(\mathbf{x}, t, c)$, where t and c are time embedding and class embedding. Each block B_i contains a self-attention and a feed-forward module. The patchification/depatchification operations are used to encode/decode latent code $\mathbf{z}$ to/from a sequence of image tokens.

3.2. Parameter-efficient Fine-tuning

DiffFit Design. This section illustrates the integration of DiT with DiffFit. Note that DiffFit may be generalized to other diffusion models *e.g.* Stable Diffusion. Our approach, illustrated in Figure 4, involves freezing the majority of parameters in the latent diffusion model and training only the bias term, normalization, and class condition module. We moreover insert learnable scale factors $\boldsymbol{\gamma}$ into several blocks of the diffusion model, wherein the $\boldsymbol{\gamma}$ is initialized to 1.0 and multiplied on corresponding layers of each block. Each block typically contains multiple components such as multi-head self-attention, feed-forward networks, and layer normalization, and the block can be stacked N times. Please refer to Algorithm 1 for additional detailed information.

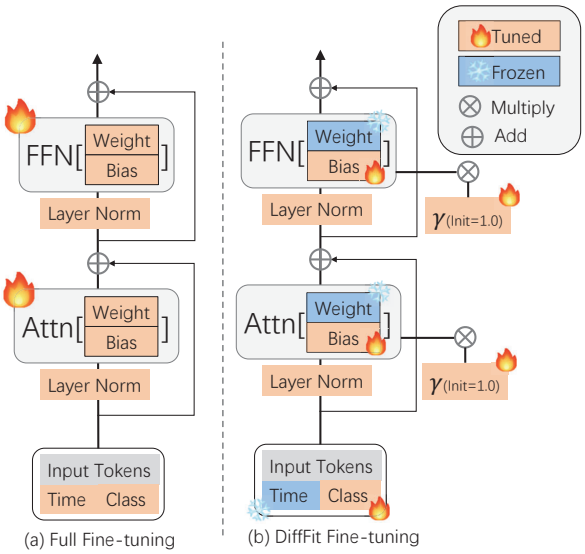

Figure 4: Architecture comparison between full fine-tuning and the proposed DiffFit. DiffFit is simple and effective, where most of the weights are frozen and only bias-term, scale factor $\boldsymbol{\gamma}$, LN, and class embedding are trained.

Fine-tuning. During fine-tuning, diffusion model parameters are initially frozen, after which only specific parameters related to bias, class embedding, normalization, and scale factor are selectively unfrozen. Our approach, outlined in Algorithm 2, enables fast fine-tuning while minimizing disruption to pre-trained weights. DiT-XL/2 requires updating only 0.12% of its parameters, leading to training times approximately 2× faster than full fine-tuning. Our approach avoids catastrophic forgetting while reinforcing the pre-trained model's knowledge and enabling adaptation to specific tasks.

Inference and Storage. After fine-tuning on K datasets, we only need to store one copy of the original model's full parameters and $K\times$ dataset-specific trainable parameters, typically less than 1M for the latter. Combining these weights for the diffusion model enables adaptation to multiple domains for class-conditional image generation.

3.3. Analysis

In this subsection, we provide intuitive theoretical justifications for the efficacy of scaling factors and reveal the

Algorithm 1 Add trainable scale factor γ in the model.

```
import torch
import torch.nn as nn

class Block():
  # An example of adding trainable scale factors
  def __init__():
    # Initilize gamma to 1.0
    self.gamma1 = nn.Parameter(torch.ones(dim))
    self.gamma2 = nn.Parameter(torch.ones(dim))

  def forward(x, c, t):
    # Apply gamma on self-attention and ffn
    x = x + self.gamma1 * self.attn(wrap(x, c, t))
    x = x + self.gamma2 * self.ffn(wrap(x, c, t))

  return x
```

Algorithm 2 Parameter-efficient fine-tuning strategy.

```
# List of trainable parameters
trainable_names = ["bias","norm","gamma","y_embed"]

def finetune():
  # Step 1: Freeze all params
  for name, param in model.named_parameters():
    param.requires_grad = False

  # Step 2: Unfreeze specific params
  for name, param in model.named_parameters():
    # unfreeze specific parameters with name
    if match(name, trainable_names):
      param.requires_grad = True

  # Step 3: Fine-tuning
  train(model, data, epochs)
```

Method \ Dataset	Food	SUN	DF-20M	Caltech	CUB-Bird	ArtBench	Oxford Flowers	Standard Cars	Average FID	Params. (M)	Train Time
Full Fine-tuning	10.46	**7.96**	**17.26**	35.25	5.68	25.31	21.05	**9.79**	16.59	673.8 (100%)	1×
Adapt-Parallel [7]	13.67	11.47	22.38	35.76	7.73	38.43	21.24	10.73	20.17	4.28 (0.63%)	0.47×
Adapt-Sequential	11.93	10.68	19.01	34.17	7.00	35.04	21.36	10.45	18.70	4.28 (0.63%)	0.43×
BitFit [61]	9.17	9.11	17.78	34.21	8.81	24.53	20.31	10.64	16.82	**0.61 (0.09%)**	0.45×
VPT-Deep [24]	18.47	14.54	32.89	42.78	17.29	40.74	25.59	22.12	26.80	2.81 (0.42%)	0.50×
LoRA-R8 [23]	33.75	32.53	120.25	86.05	56.03	80.99	164.13	76.24	81.25	1.15 (0.17%)	0.63×
LoRA-R16	34.34	32.15	121.51	86.51	58.25	80.72	161.68	75.35	81.31	2.18 (0.32%)	0.68×
DiffFit **(ours)**	**6.96**	8.55	17.35	**33.84**	**5.48**	**20.87**	**20.18**	9.90	**15.39**	0.83 (0.12%)	0.49×

Table 1: FID performance comparisons on 8 downstream datasets with DiT-XL-2 pre-trained on ImageNet 256×256.

principle behind their effectiveness. We note that these theoretical justifications are intended as a simple proof of concept rather than seeking to be comprehensive, as our contributions are primarily experimental.

Specifically, recent theoretical works for diffusion models, such as [10, 9, 28, 6, 5], have shown that under suitable conditions on the data distribution and the assumption of approximately correct score matching, diffusion models can generate samples that approximately follow the data distribution, starting from a standard Gaussian distribution. Given a mapping $\boldsymbol{f}$ and a distribution P, we denote $\boldsymbol{f}\#P$ as a pushforward measure, i.e., for any measurable Ω, we have $(\boldsymbol{f}\#P)(\Omega) = P(\boldsymbol{f}^{-1}(\Omega))$. Note that our base model DiT is pre-trained on ImageNet with a resolution of 256×256 and is fine-tuned on downstream datasets with the same resolution but much fewer data points and classes. Motivated by this, if assuming that the data in the ImageNet 256×256 dataset follows a distribution Q_0, then we can assume that the data in the downstream dataset follows a distribution $P_0 = \boldsymbol{f}_{\boldsymbol{\gamma}^*}\#Q_0$, where $\boldsymbol{f}_{\boldsymbol{\gamma}^*}$ is a linear mapping dependent on some ground-truth scaling factors $\boldsymbol{\gamma}^*$.

With these assumptions in place, we can formulate an intuitive theorem that provides insight into the effectiveness of scaling factors. A formal version of this theorem is included in the supplementary material.

Theorem 1 (informal)**.** *Suppose that for a dataset generated from data distribution Q_0, we can train a neural network such that the diffusion model generates samples that approximately follow Q_0. Further assuming that the data distribution P_0 for a relatively small dataset can be written as $P_0 = \boldsymbol{f}_{\boldsymbol{\gamma}^*}\#Q_0$ with $\boldsymbol{f}_{\boldsymbol{\gamma}^*}$ being a linear mapping dependent on ground-truth scaling factors $\boldsymbol{\gamma}^*$. Then, if only retraining the neural network with the goal of optimizing scaling factors (and all other parameters remain unchanged), under suitable conditions, a simple gradient descent algorithm seeks an estimate $\hat{\boldsymbol{\gamma}}$ that is close to $\boldsymbol{\gamma}^*$ with high probability. Furthermore, with the fine-tuned neural network corresponding to $\hat{\boldsymbol{\gamma}}$, the denoising process produces samples following a distribution that is close to P_0.*

In summary, Theorem 1 essentially states that when distributions Q_0 and P_0 satisfy the condition that $P_0 = \boldsymbol{f}_{\boldsymbol{\gamma}^*}\#Q_0$ with $\boldsymbol{f}_{\boldsymbol{\gamma}^*}$ being dependent on ground-truth scaling factors $\boldsymbol{\gamma}^*$, diffusion models can transfer from distribution Q_0 to P_0 in the denoising process with fine-tuning the scaling factors in the training process.

4. Experiments

4.1. Implementation Details

Our base model is the DiT[1], which is pre-trained on ImageNet 256×256 with 7 million iterations, achieving an FID score of 2.27[2]. However, since the original DiT reposi-

[1] https://github.com/facebookresearch/DiT

[2] https://dl.fbaipublicfiles.com/DiT/models/DiT-XL-2-256x256.pt

Figure 5: Fine-tune DiT-XL/2-512 from the checkpoint of DiT-XL/2-256 using DiffFit with the positional encoding trick.

tory does not provide training code, we re-implemented this and achieved reasonable results. Following DiT, we set the constant lr=1e-4 for full fine-tuning and set the classifier-free guidance to 1.5 for evaluation and 4.0 for visualization. In addition, we re-implemented several parameter-efficient fine-tuning methods such as Adaptor, BitFit, Visual Prompt Tuning (VPT), and LoRA. We found VPT to be sensitive to depth and token number while training was exceptionally unstable. As such, we sought for a more stable configuration with depth=5, token=1, and kept the final layers unfrozen for all tasks. We enlarge lr $\times 10$ for parameter-efficient fine-tuning settings to obtain better results following previous works [23, 7].

4.2. Transfer to Downstream Datasets

Setup. For fine-tuning downstream small datasets with 256×256 resolutions, we use 8 V100 GPUs with a total batch size of 256 and train 24K iterations. We choose 8 commonly used fine-grained datasets: Food101, SUN397, DF-20M mini, Caltech101, CUB-200-2011, ArtBench-10, Oxford Flowers and Stanford Cars. We report FID using 50 sampling steps for all the tasks. Most of these datasets are selected from CLIP downstream tasks except ArtBench-10 since it has distinct distribution from ImageNet, which enables a more comprehensive evaluation of the out-of-distribtion generalization performance of our DiffFit.

Results. We list the performance of different parameter-efficient fine-tuning methods in Table 1. As can be seen, by tuning only 0.12% parameters, our DiffFit achieves the lowest FID on average over 8 downstream tasks. While full fine-tuning is a strong baseline and has slightly better results on 3/8 datasets, it is necessary to fine-tune 100% parameters. Among all baselines, the performance of LoRA is surprisingly poor. As discussed in [18], LoRA performs worse on image classification tasks than other parameter-efficient fine-tuning methods. As the image generation task is conceptually more challenging than the image classification task, it is reasonable that the performance gap between LoRA and other approaches becomes larger here.

Method	FID ↓	Training Cost (GPU days) ↓
BigGAN-Deep [2]	8.43	256-512
StyleGAN-XL [48]	2.41	400
ADM-G, AMD-U [12]	3.85	1914
DiT-XL/2 [39]	3.04	1733
DiffFit **(ours)**	**3.02**	**51** (+950†)

Table 2: **Class-conditional image generation on ImageNet 512×512.** The training cost of the original DiT model and our method is measured on V100 GPU devices, and other methods are quoted from original papers. †: 950 GPU days indicates the pre-training time of DiT-XL/2 model on ImageNet 256×256 with 7M steps.

Method	Pre-trained Checkpoint	FID ↓
Full Fine-tune	IN256(7M)→Food256	23.08
DiffFit	IN512 (3M)	19.25
DiffFit	IN256 (7M)→Food256	19.50
+PosEnc Trick	IN256 (7M)→Food256	**19.10**

Table 3: **Class-conditional image generation on Food-101 512×512.** We use two pre-trained models from: (1) ImageNet 512×512, and (2) ImageNet 256×256 and firstly fine-tuned on Food-101 256×256.

4.3. From Low Resolution to High Resolution

Setup. Considering the generating images with different resolutions as a special type of distribution shift, our proposed method can effortlessly adapt a pre-trained low-resolution diffusion model to generate high-resolution images. To demonstrate the effectiveness of DiffFit, we load a pre-trained ImageNet 256×256 DiT-XL/2 checkpoint and fine-tune the model on the ImageNet 512×512. We employ a positional encoding trick to speed up fine-tuning. We fine-tune DiT-XL/2 on ImageNet 512×512 using 32 V100 GPUs with 1024 batch size and 30K iterations. We report FID using 250 sampling steps. Note that we do not need to fine-tune the label embedding here since the label does not change.

Positional Encoding Trick. DiT [39] adopts a static sinusoidal 2D positional encoding scheme. To better utilize the positional information encoded in the pre-trained model, we develop a sinusoidal interpolation that aligns the positional encoding of 512×512 resolution with that of 256×256 resolution. This is implemented by replacing each pixel coordinate (i, j) in the positional encoding formula with its half value $(i/2, j/2)$, which is simple and have no extra costs.

Results. As demonstrated in Table 2, DiffFit achieves 3.02 FID on the ImageNet 512×512 benchmark, surpassing ADM's 3.84 and the official DiT's 3.04. DiffFit sets a new state-of-the-art among diffusion-based methods. Importantly, our method is significantly more efficient than the previous methods, as the fine-tuning process only requires an overhead of 51 GPU days. Even when accounting for the 950 GPU days of pre-training, our method remains superior to the 1500+ GPU days required by DiT and ADM. We observe faster training convergence using the positional encoding trick, as shown in Figure 5. For more visualizations please see Figure 1 and appendix.

In Table 3, we conducted fine-tuning experiments on the Food101 dataset with a resolution of 512×512 using the DiffFit method. Our results reveal that fine-tuning a pre-trained Food101 256×256 checkpoint with DiffFit yields a FID of 4 improvements over full fine-tuning. Interestingly, we found that utilizing a pre-trained ImageNet 512×512 checkpoint leads to a FID performance similar to that achieved by the pre-trained Food101 with 256×256 resolution. Moreover, we observed a slight improvement in FID performance by incorporating the proposed positional encoding trick into the fine-tuning process.

4.4. Fine-tuning Convergence Analysis

To facilitate the analysis of converging speed, we present the FID scores for several methods every 15,000 iterations in the Food-101, ArtBench-10, Flowers-102 and CUB-200 datasets, as shown in Figure 6. Our observations demonstrate that full fine-tuning, BitFit, and our proposed DiffFit exhibit similar convergence rates, which surpass the initial performance of AdaptFormer and VPT. While AdaptFormer initially presents inferior performance, it shows a rapid improvement in the middle of training. In contrast, VPT exhibits the slowest rate of convergence. Compared to full fine-tuning, DiffFit freezes most of the parameters and thus maximally preserves the information learned during the pre-training and thus achieves a better fine-tuned performance. Compared to BifFit, DiffFit adjusts the feature using scale factor, resulting in faster convergence and better results.

The above observation and analysis verify that our proposed DiffFit demonstrates fast adaptation abilities to target domains and an excellent image generation capability.

4.5. Ablation studies

Setup. We conduct ablation studies on Food101 dataset, which has 101 classes. Each class has 750/250 images in the train/test set. Other settings are the same as Section 4.2.

Scale factor γ in different layers. We investigate the effect of the scaling factor γ via 2 designs: gradually adding γ from deep to shallow (Table 4a) and from shallow to

Blocks	#params (M)	FID ↓
28→25	0.747	10.04
28→22	0.754	10.03
28→18	0.763	10.33
28→14	0.770	10.51
28→11	0.779	9.92
28→8	0.786	9.28
28→4	0.796	8.87
28→1	0.803	**8.19**

(a) **Scale: Deep→Shallow.**

Blocks	#params (M)	FID ↓
1→3	0.745	8.29
1→7	0.754	7.99
1→11	0.763	7.72
1→14	0.770	**7.61**
1→18	0.779	7.63
1→21	0.786	7.67
1→25	0.796	7.85
1→28	0.803	8.19

(b) **Scale: Shallow→Deep.**

#ID	Scale Factor	FID ↓
1	NA (BitFit)	9.17
2	+Blocks	8.19
3	+PatchEmb	9.05
4	+TimeEmb	8.46
5	+QKV-Linear	7.37
6	+Final Layer	7.49
7	+ID 1, 2, 5, 6	7.17
8	+(1→14 Layers)	**6.96**

(c) **Scale Location.**

LR Ratio	FID ↓
0.1×	25.85
0.2×	21.42
0.5×	17.16
1×	15.68
2×	13.84
5×	10.97
10×	**8.19**
20×	8.30

(d) **Learning Rate.**

Table 4: **Ablation experiments on Food101 dataset with DiT-XL/2.** Red means adding scale factor leads to negative results and green means positive. The best setting are marked in gray.

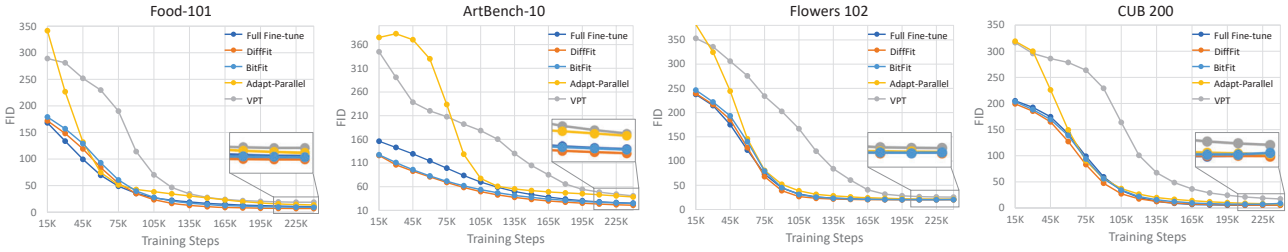

Figure 6: **FID of five methods every 15K iterations on four downstream datasets.** Our observations indicate that DiffFit can rapidly adapt to the target domain while maintaining a robust FID score.

deep (Table 4b). The results demonstrate that adding γ before the 14th layer of DiT-XL/2 gradually increases the performance from 8.29 to 7.61 FID while adding more γ in the deeper layers hurts the performance. We hypothesize that deeper layers are responsible for learning high-level features that capture abstract data patterns, which contribute to synthesizing the final output and are often complex and nonlinear. Adding γ in deeper layers poses a risk of disrupting the learned correlations between the high-level features and data, which might negatively impact the model.

Scale factor γ in different modules. We study the impact of scaling factor γ in various DiT modules, as illustrated in Table 4c. Based on BitFit [61]'s FID score of 9.17, incorporating the scaling factor γ in transformer blocks and QKV-linear layers of self-attention can significantly enhance the performance, resulting in a FID score of 7.37 (row 5). However, introducing scale factors in other modules, such as patch embedding (row 3) and time embedding (row 4), does not bring noticeable improvements. By integrating the effective designs, i.e., adding γ in blocks, QKV-linear and final layers, we improve FID score to 7.17 (row 7). We further improve the FID score of 6.96 by placing γ in the first 14 blocks (Table 4b). This optimal setting is adopted as the final setting for our approach.

Learning rate. Adjusting the learning rate is a crucial step in fine-tuning. Parameter-efficient fine-tuning typically requires a larger learning rate than the pre-training [23, 7] since pre-training has already initialized most of the model's parameters to a certain extent and a larger learning rate can help quickly adapt the remaining parameters to the new tasks. We perform a learning rate search on our method, as shown in Table 4d. We observe that using a learning rate 10× greater than pre-training yields the best result. Larger learning rates than 10× resulted in decreased performance and even unstable training.

5. Conclusions and Limitations

In this paper, we propose DiffFit, a straightforward yet effective fine-tuning approach that can quickly adapt a large pre-trained diffusion model to various downstream domains, including different datasets or varying resolutions. By only fine-tuning bias terms and scaling factors, DiffFit provides a cost-effective solution to reduce storage requirements and speed up fine-tuning without compromising performance. One limitation is that our experiments mainly focus on class-conditioned image generation. It is still unclear whether this strategy could perform equally well in more complex tasks such as text-to-image generation or video/3D generation. We leave these areas for future research.

Acknowledgement. We would like to express our gratitude to Junsong Chen, Chongjian Ge, and Jincheng Yu for their assistance with experiments on LLaMA and DreamBooth.

References

[1] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, et al. ediffi: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv*, 2022. 3

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv*, 2018. 7

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 2020. 3

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 3

[5] Minshuo Chen, Kaixuan Huang, Tuo Zhao, and Mengdi Wang. Score approximation, estimation and distribution recovery of diffusion models on low-dimensional data. *arxiv*, 2023. 5

[6] Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru R Zhang. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. In *ICLR*, 2023. 5

[7] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *arXiv*, 2022. 2, 3, 5, 6, 8

[8] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 2018. 3

[9] Valentin De Bortoli. Convergence of denoising diffusion models under the manifold hypothesis. *TMLR*, 2022. 5

[10] Valentin De Bortoli, James Thornton, Jeremy Heng, and Arnaud Doucet. Diffusion schrödinger bridge with applications to score-based generative modeling. In *NeurIPS*, 2021. 5

[11] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. *arXiv*, 2023. 3

[12] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *NeurIPS*, 2021. 2, 3, 7

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv*, 2020. 3

[14] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021. 3

[15] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *JMLR*, 2021. 3

[16] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *NeurIPS*, 2021. 3

[17] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 3

[18] Xuehai He, Chunyuan Li, Pengchuan Zhang, Jianwei Yang, and Xin Eric Wang. Parameter-efficient fine-tuning for vision transformers. *arXiv*, 2022. 6

[19] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv*, 2022. 2

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020. 2, 3, 4

[21] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv*, 2022. 2

[22] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *ICML*, 2019. 3

[23] 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*, 2021. 2, 3, 5, 6, 8

[24] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *ECCV*, 2022. 2, 3, 5

[25] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two pure transformers can make one strong gan, and that can scale up. *NeurIPS*, 2021. 3

[26] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *NeurIPS*, 2021. 3

[27] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 2019. 3

[28] Holden Lee, Jianfeng Lu, and Yixin Tan. Convergence for score-based generative modeling with polynomial complexity. In *NeurIPS*, 2022. 5

[29] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv*, 2021. 3

[30] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv*, 2021. 3

[31] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *ECCV*, 2022. 3

[32] Zhiqi Li, Wenhai Wang, Enze Xie, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, Ping Luo, and Tong Lu. Panoptic segformer: Delving deeper into panoptic segmentation with transformers. In *CVPR*, 2022. 3

[33] Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Scaling & shifting your features: A new baseline for efficient model tuning. *arXiv*, 2022. 3

[34] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 3

[35] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. In *CVPR*, 2022. 3

[36] Robert L Logan IV, Ivana Balažević, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. Cutting down on prompts and parameters: Simple few-shot learning with language models. *arXiv*, 2021. 3

[37] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, 2021. 4

[38] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *CVPR*, 2022. 2

[39] William Peebles and Saining Xie. Scalable diffusion models with transformers. *arXiv*, 2022. 3, 4, 7

[40] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022. 2

[41] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018. 3

[42] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019. 3

[43] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv*, 2022. 2, 3

[44] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, 2015. 3

[45] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 2, 3, 4

[46] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv*, 2022. 3

[47] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo-Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. In *NeurIPS*. 3

[48] Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. In *SIGGRAPH*, 2022. 7

[49] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *NeurIPS*, 2019. 2

[50] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*. 3

[51] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021. 2

[52] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *ICCV*, 2021. 3

[53] Peize Sun, Jinkun Cao, Yi Jiang, Rufeng Zhang, Enze Xie, Zehuan Yuan, Changhu Wang, and Ping Luo. Transtrack: Multiple object tracking with transformer. *arXiv*, 2020. 3

[54] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 3

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017. 3, 4

[56] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, 2021. 3

[57] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvt v2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 2022. 3

[58] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *NeurIPS*, 2021. 3

[59] Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. Raise a child in large language model: Towards effective and generalizable fine-tuning. *arXiv*, 2021. 3

[60] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *ICCV*, 2021. 3

[61] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv*, 2021. 2, 3, 5, 8

[62] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *ICCV*, 2021. 3

[63] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *CVPR*, 2021. 3

[64] Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. Deepvit: Towards deeper vision transformer. *arXiv*, 2021. 3

[65] Daquan Zhou, Weimin Wang, Hanshu Yan, Weiwei Lv, Yizhe Zhu, and Jiashi Feng. Magicvideo: Efficient video generation with latent diffusion models. *arXiv*, 2022. 2

[66] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *IJCV*, 2022. 3