Generating 3D Brain MRIs by Parameter Efficient Finetuning of Diffusion Transformer (DiT)

Yingbo Li
yingboli@stanford.edu

Abstract

As synthetic 3D brain MRI data can provide a lot of values for scientific research, generative AI models are used to create realistic synthetic brain images. Diffusion models has become one of the state-of-art methods for image generation, and has been shown successes for brain image generation. In this project, my goal is to finetune a transformer based diffusion model, DiT, to generate 3D brain MRI images. Because both the variational autoencoder and the patched transformers in DiT are pretrained on 2D images, to enable the finetuning, in the data pre-processing stage, I convert the 3D training samples to 2D, by tiling the 2D slices together into larger-sized images. Then I use these pro-processed larger images to finetune the DiT model. My experiments show that the generated synthetic samples can generate good quality images, preserving anatomy- consistent 3D structures.

1. Introduction

3D Brain MRI data are usually expensive to collect. Since brain research often requires a decent size of brain image data, scientists find that synthetic brain MRIs can provide a lot of values. Thus, a natural choice is to use generative AI models to create realistic synthetic brain images.

Diffusion models has become one of the state-of-art methods for image generation, and has been used for brain image generation [7]. Recently, thanks to OpenAI's SORA, the transformer based diffusion model, DiT [6], becomes extremely popular. In this project, I aim to finetune the DiT model to generate 3D brain MRI images.

The input to my algorithm are 3D brain MRI images. I first pre-process them into 2D images, and then use them as the training data to finetune the DiT algorithm. Last, after applying the finetuned DiT to to generate 2D images, and I post-process them back to 3D MRI images, i.e, the outputs.

2. Related Work

Diffusion models have been becoming the state-of-art method for image/video generation [10] [3]. For example, [7] has demonstrated that the diffusion model based approaches outperform Generative Adversarial Networks (GAN) based approaches for MRI image generation. [7] trains a novel conditional diffusion model to generate slices of 2D brain MRI images that conditional on other 2D images slices, so that the produced 3D MRIs are anatomy-consistent and of high quality.

While the diffusion model has gained a lot of successes, it can be slow to train and to sample with high resolutions images directly. Therefore, [8] proposed the latent diffusion model (LDM), which runs the diffusion process on a latent space after dimension reduction, rather than on the original pixel space. The dimension reduction step is conducted via the variational autoencoder (VAE) models [4]. In training time, the VAE encoder is first applied to the training images, then a diffusion model is trained on the compressed images. In generation time, after the reverse diffusion process is applied to obtain image in the latent space, it will then go though the VAE decoder to be turned into images on the original pixel scale. Thus, the diffusion model is trained and scored on a much smaller space and thus much more efficient.

For image generation diffusion models, the most common deep learning models used on reverse process has been the U-net [9], which consists of many convolutional layers, max pooling layers to reduce dimension, and up convolutional layers to increase the dimension back to the original one. Recently, the Diffusion Transformers (DiT) paper [6] replaces the U-net with a transformer, i.e., multi-headed self-attention architecture, which achieves the optimal image generation performance and is viewed as state-of-the-art.

To make use of the pretained DiT model parameters, finetuning algorithms has been proposed with successful results. For example, a recent parameter efficient finetuing paper [11] proposed an efficient strategy called DiffFit. Because it achieves great training speed-up and can generate high quality images, in this paper I mainly follow their fine-tuning strategy.

3. Methods

Since my plan is to finetune diffusion transformer (DiT), I will briefly introduce the method of diffusion models in general, and then DiT, followed by DiffFit, the parameter efficient finetuning algorithm.

3.1. Diffusion models

Diffusion models consist of two Markov chains of diffusion steps, a forward diffusion process and a reverse diffusion process. In the forward process, random independent Gaussian noises are gradually added to the observed data $\mathbf{x}_0$ (e.g., a real image). At step $t = 1, 2, \ldots, T$,

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathrm{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}),$$

where $\beta_t \in (0, 1)$ are hyperparameters that usually follow a linear or cosine based schedule. After a lot of steps (e.g., $T = 250$ in DiT), the image will become something no different to pure white noise.

Then we use the reverse process to gradually remove the noise to get the original image. Here, each reverse step is assumed to be Gaussian with unknown mean $\mu(\cdot)$ and covariance $\Sigma(\cdot)$:

$$p(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathrm{N}(\boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)).$$

In order to estimate the unknown mean $\mu(\cdot)$ and covariance $\Sigma(\cdot)$, usually they are assumed to follow some deep learning models (U-net or transformer), and the parameter θ are obtained by optimizing the variational lower bound (ELBO),

$$L_{VLB} = \mathbb{E}_{q(\mathbf{x}_{0:T})} \frac{\log q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}{\log p_\theta(\mathbf{x}_{0:T})}$$

For further simplification, the mean $\boldsymbol{\mu}(\cdot)$ can be re-parameterized into a noise prediction network $\boldsymbol{\epsilon}(\cdot)$, which can then be estimated with L2 loss $L_{simple} = \|\boldsymbol{\epsilon}_\theta(\mathbf{x}_t) - \boldsymbol{\epsilon}_t\|^2$.

3.2. Diffusion Transformer (DiT)

The diffusion transformers paper [6] uses transformers, i.e., multi-headed self-attention architecture, as the noise prediction $\boldsymbol{\epsilon}_\theta(\cdot)$. Within the LDM framework, DiT first applies a pre-trained VAE model from [8] to compress the image height and width by 8 times. For example, it converts the image x from it original dimensions $256 \times 256 \times 3$ to a much smaller latent space of dimensions $32 \times 32 \times 4$. In additional to the 256×256 size image, another version of DiT is trained on larger images of size $512 \times 512 \times 3$, and the corresponding latent space size becomes $64 \times 64 \times 4$.

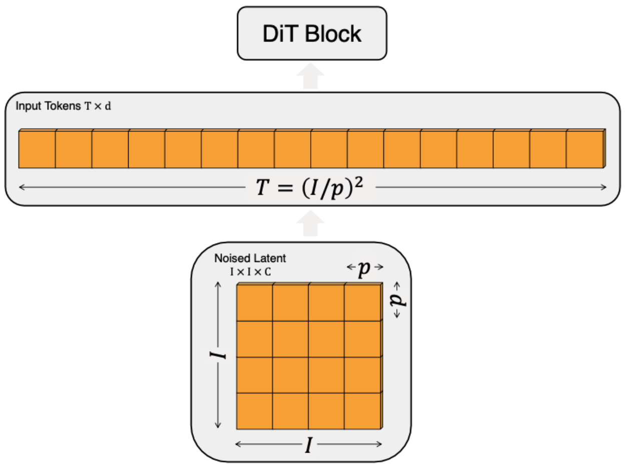


Figure 1. DiT patching. Screenshot from the DiT paper [6]

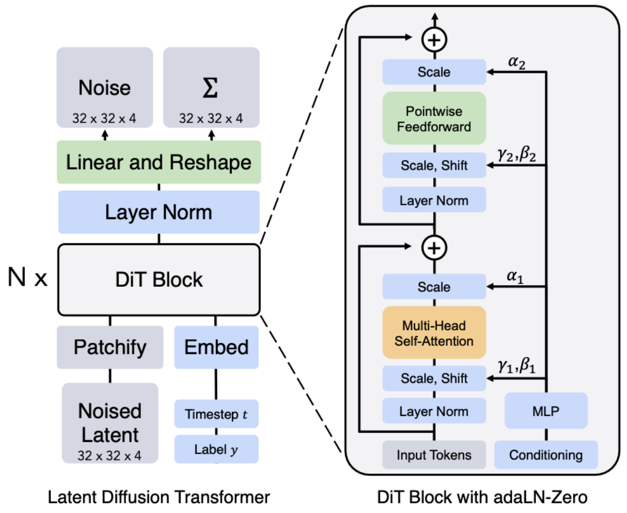


Figure 2. DiT architecture. Screenshot from the DiT paper [6]

Then DiT patchifies the latent image z of size $I \times I \times C$ into patches of size p and converts it into sequence of patches with length $(I/p)^2$. For each patch, its dimension $p^2 \times C$ is then linearly converted into the hidden size d. In the network, a sin-cos based position embeddings are applied to the sequence (see Figure 1).

As to the transformer block design, in-context conditioning are enabled in two places, such that both the adaptive layer normalization layer and the scale layer right before the residual connections depend on both the diffusion step t and the class label c (see Figure 2).

The DiT paper explores different model sizes (S, B, L, XL), mixed with different size of patch $p = 2, 4, 8, 16$. The empirical results in the paper suggest that the largest model, XL with $p = 2$, which has 28 layers of transformer blocks, each of which has 16 heads and hidden size 1152, achieves the best performance. Therefore, my work in this project will be based on pre-trained parameters in this DiT-XL-2 model.

The DiT authors release two versions of this largest model, one pretrained on 256×256 resolution ImageNet data, and the other pretrained on 512×512 resolution ImageNet data. Since the image size only affects the length of the patches, it doesn't affect the totally number of parameters in the network. Assuming that larger images contain more information, in this paper, I conduct the finetuning work on the 512×512 model (`https://dl.fbaipublicfiles.com/DiT/models/DiT-XL-2-512x512.pt`).

After realizing that the size of the training image does not impact the network size and structure, I find that I can finetune the DiT model with larger than 512-resolution images. The only modification needed is to modify the position embedding, which is not a part of the model parameters to be learned from the data. Of course, the cost is that the length of the patched sequences becoming longer, and the computation complexity of diffusion model is on the squared order of the sequence length.

My work is to modify the PyTorch code base released from Facebook Research,`https://github.com/facebookresearch/DiT`. This code base contains a `models.py` script of the DiT model, a `train.py` script to train DiT, and a `sample.py` script to generate images from the model. I mainly modify these three scripts to enable the finetuning work.

3.3. Parameter efficient finetuning of DiT

Since DiT contains huge number of parameters, to finetune it, it may not be necessary to train each and every parameter. In fact, DiffFit [11] only finetuned a very small fraction of the parameters while freezing majority of them.

DiffFit first adds new scale parameters γ to the self-attention and feed-forward layers in each transformer block from layer 1 to 14. Its strategy is to only finetunes all biases, layer normal parameters, class embeddings, and the newly-added γ scale parameters. It also recommended to use a learning rate that is 10 times as large as the pretraining learning rate, so that the updating of parameters such as biases are more efficient. See Figure 3 for a visual illustration.

In this paper, due to time constraints, I did not explore the option of adding the γ blocks. Thus, I finetune all biases, layer normalization parameters, and class embeddings. In addition, the learning rate I use is 5 times as large as the original DiT learning rate.

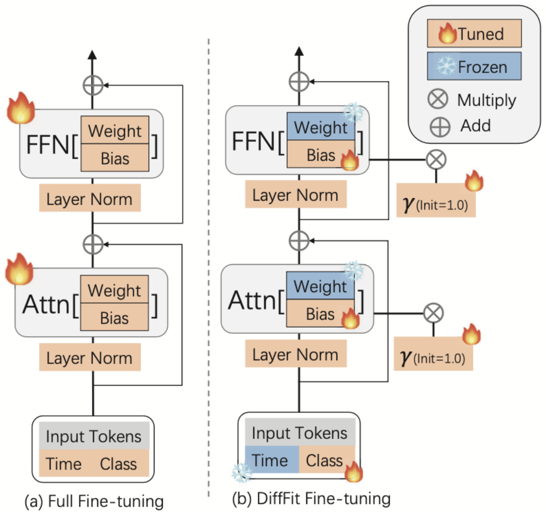


Figure 3. Screenshot from the DiffFit paper [11]. My implementation in this paper omits the γ blocks.

4. Dataset and Features

The data I use are from the public dataset ADNI (`https://adni.loni.usc.edu/`). The training data consists of a very small random sample of 63 examples, and separately, the test data contains another 32 examples. Each data point is a black-and-white 3D brain MRI image. Its original dimension is $144 \times 176 \times 144$.

For pre-processing, I first apply linear interpolation to convert the size of the 3D image to $128 \times 128 \times 128$. An example of the real image is shown in Figure 4. Then, along the third dimension, I decompose each 3D image into 128 2D slices of size 128×128. To concatenate all these 128 images into a single 2D image, I make use of the three color channels to put three neighboring slices into a single $128 \times 128 \times 3$ color image, such that I have 43 color images in total. Then, I tile these images into a 7×7 block of a bigger squared image. Thus, the size of the pre-processed image (see Figure 5 for an example) is $7 \times 128 = 896$.

When input the training images into the DiT model, I put them in the same class. This means that, when finetuning the model, only class embedding of this class will be updated.

5. Experiments and Results

In my finetuning process, I followed the original DiT training parameters whenever appropriate.

- I start from `DiT-XL-2-512x512.pt` as the pre-trained model.
- I use the AdamW optimizer (the same as what DiT uses) to finetune all biases, layer normalization param-

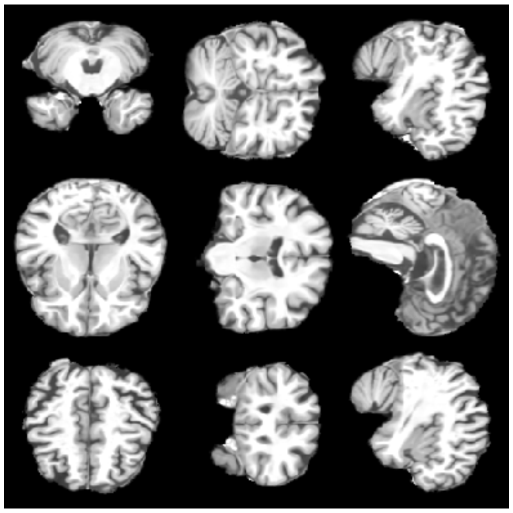


Figure 4. 2D slices of a real 3D brain MRI image example. Each column is from each of the three dimensions, and the rows are at locations 32, 64, 96, respectively.

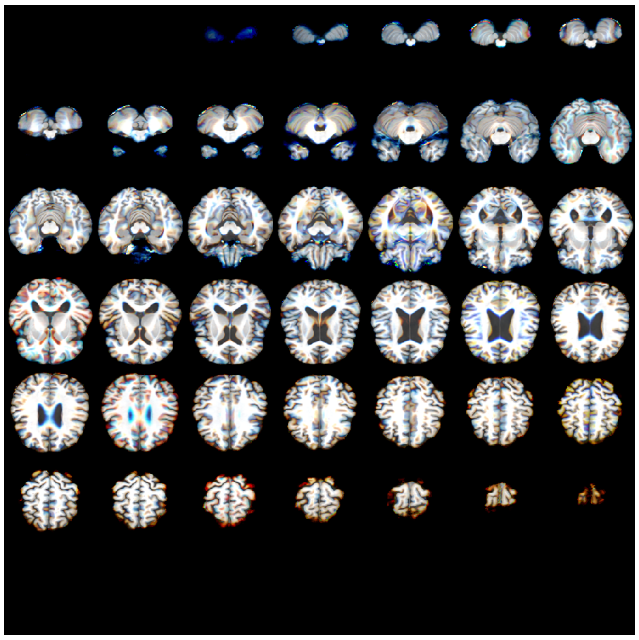


Figure 5. A real example of pre-processed 896×896 color image, that contains all information from the original $128 \times 128 \times 128$ 3D MRI image. This is the training data input to DiT.

eters, and class embeddings, while freezing all other parameters.

- I use a learning rate 5×10^{-4}, that is 5 times as large as the original learning rate. My rational of using a bigger learning is that I hope it can be more efficient and speed up the training. Also, the DiffFit paper suggested to use a larger learning rate.

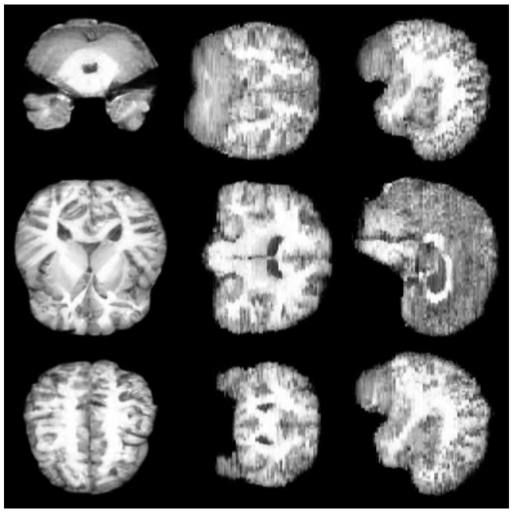


Figure 6. 2D slices of a synthetic 3D brain MRI image. Again, each column is from each of the three dimensions, and the rows are at locations 32, 64, 96, respectively.

- Since I only have one A100 GPU to training my model, the largest batch size that can fit the 40G GPU memory is merely 4.

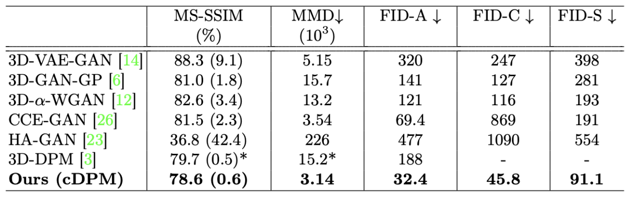
- I trained on this A100 GPU for about 36 hours and finished 66000 steps, i.e., about 4400 epochs.

To evaluate the finetuned model, I randomly generated 32 synthetic samples. Actually, due to the big 896×896 image size, the largest number of samples I can generate at a time is 8. I generated 4 times with different random seeds to obtain the 32 synthetic samples. Then I applied the reverse process as in the pre-processing, to convert the $896 \times 896 \times 3$ image back to 3D image of size $128 \times 128 \times 128$.

Figure 6 shows one generated sample, via three views of 2D slices, decomposed from each of the three dimensions. First, overall we can see that the generated image does look like brain MRI images. In particular, in the first column, where the 2D slices are decomposed along the third dimension, the quality of the generated images seems to be satisfactory, with clear details.

For the second and third columns in Figure 6, the image quality is still acceptable, as they give the correct overall shapes and rough details of brain MRI at different slice locations. But they look noisier than the first column. In particular, they lack horizontal smoothness to some extent. This is not surprising because for these type of slices, which decomposes the 3D image along the first or the second dimension, do not directly appear in the training data of pre-processed 7×7 tiled color images.

My quantitative validation metrics agrees with the above qualitative observations. Following the [7], I computed the same three metrics.



	MS-SSIM (%)	MMD$\downarrow$ (10^3)	FID-A $\downarrow$	FID-C $\downarrow$	FID-S $\downarrow$
3D-VAE-GAN [14]	88.3 (9.1)	5.15	320	247	398
3D-GAN-GP [6]	81.0 (1.8)	15.7	141	127	281
3D-α-WGAN [12]	82.6 (3.4)	13.2	121	116	193
CCE-GAN [26]	81.5 (2.3)	3.54	69.4	869	191
HA-GAN [23]	36.8 (42.4)	226	477	1090	554
3D-DPM [3]	79.7 (0.5)*	15.2*	188	-	-
Ours (cDPM)	**78.6 (0.6)**	**3.14**	**32.4**	**45.8**	**91.1**

Figure 7. Screenshot from the cDPM paper [5].

MS_SSIM (%)	MMD (10^3)	FID-A	FID-C	FID-S
83.6	8.7	126	279	239

Table 1. Results from finetuned DiT. Note that the MS_SSIM among the 32 real test data samples is 78.5.

- 3D multi-scale Structure Similarity (MS-SSIM) [5]. This metric is to measure how similar two images are, and its range is from 0 (completely different) to 1 (completely the same). I applied it to every pair of the 32 synthetic images (using the code `https://github.com/cyclomon/3dbraingen/tree/master/pytorch_ssim`), and report the average.

- 3D Maximum-Mean Discrepancy (MMD) score [1], computed among all pairs formed by one synthetic image and one real test image. The smaller the better.

- 2D Frèchet Inception Distance (FID) [2], for three views of the center slices. FID measures how close two distributions of images are. When applied on two 2D image sets with equal sizes, the smaller FID is, the more similar the two distributions are. I applied the implementation in the `torchmetrics` package `https://lightning.ai/docs/torchmetrics/stable/image/frechet_inception_distance.html`.

I include the quantitative results in [7] here in Figure 7, which shows that their cDPM model outperforms all GAN models for 3D brain MRI generation. My quantitative results based on finetuned DiT are shown in Table 1. Overall, although they are not as good as cDPM, my results are on par with the GAN models in general. Actually, among the seven models mentioned in Figure 7, my results outperforms four of them wrt MMD, four of them wrt FID-A, two of them wrt FID-C, and three of them wrt FID-S. Similar to all previous models, I gets a higher MS_SSIM score than the real data, which suggests my synthetic data also suffer from lack of diversity.

For the above comparison, there are two caveats. First, all metrics in Figure 7 are computed based on 500 synthetic samples and 500 randomly selected test samples. But for my paper, I compute these metrics based on 32 synthetic samples and 32 test samples. Since all the metrics are some sort of averages, I think my results can still be viewed as fair comparisons with the results in Figure 7, but the variances of my metrics are larger and thus more prone to change due to randomness in the sampling process.

Second, the training data sizes are very different. In [7], the training data contains 1261 samples. However, for my results, the training sample size is only 63. Thus, I feel my results are pretty decent. My work confirms the general belief that, finetuning large models (that are originally trained on big data, such as DiT here) may not require a lot of new training data points.

6. Conclusion and Future Work

In this work, I apply parameter efficient finetuning to the DiT model to generate 3D brain MRI images. With a creative pre-processing, I convert a 3D image into a tiled 2D image without information loss. The synthetic images generated from the finetuned model are of satisfactory quality; they display correct 3D structures and contain relatively fine details.

I think a main reason why the 3D structure can be correctly learned by such a 2D image generation model is thanks to the transformer architecture. Because transformers apply self-attention across all pairs of elements in a sequence, it is able to capture correlation among all pairs of patches, no matter they are near or far away from each other.

On the other hand, the inferior image quality of the 2D slices taken along the first and second dimension suggests rooms for improvement. I suspect that using three color channel to hold three neighboring slices may not preserve enough amount of information, especially that this color image need to go through a VAE decoder before entering the transformer model. To confirm my guess, an alternative study of using only black and white images (by copying the same image into three times into the three color channels) may be helpful. But in this case, an even larger 12×12 block of tiled image will be needed, which is of size 1536×1536.

For future work, a straight-forward extension is to use more training data (over a thousand samples would be ideal) and train it for longer. In addition, we can fintune based on newer and better transformer architecture that can handle longer sequences, for example, the masked DiT [12], which may make the training on 1536×1536 size images feasible.

7. Contributions and Acknowledgements

I would like to thank my non-CS231N mentor Wei Peng `<wepeng@stanford.edu>` for his advice and help along the project. During our weekly meetings, Wei suggested the theme of using transformer based diffusion model for the 3D MRI generation, and provided a lot of valuable advice and feedback on the data, model, and re-

sults.

This project is not combined with another course or research project.

References

[1] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[2] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[3] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[4] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[5] G. Kwon, C. Han, and D.-s. Kim. Generation of 3d brain mri using auto-encoding generative adversarial networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 118–126. Springer, 2019.

[6] W. Peebles and S. Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

[7] W. Peng, E. Adeli, T. Bosschieter, S. H. Park, Q. Zhao, and K. M. Pohl. Generating realistic brain mris via a conditional diffusion probabilistic model. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 14–24. Springer, 2023.

[8] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[9] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[10] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[11] E. Xie, L. Yao, H. Shi, Z. Liu, D. Zhou, Z. Liu, J. Li, and Z. Li. Difffit: Unlocking transferability of large diffusion models via simple parameter-efficient fine-tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4230–4239, 2023.

[12] H. Zheng, W. Nie, A. Vahdat, and A. Anandkumar. Fast training of diffusion models with masked transformers. *arXiv preprint arXiv:2306.09305*, 2023.