LoRA can Replace Time and Class Embeddings in Diffusion Probabilistic Models

Joo Young Choi
Seoul National University
lthilnklover@snu.ac.kr

Jaesung Park
Seoul National University
ryanpark7@snu.ac.kr

Inkyu Park
KRAFTON
inkyupark@krafton.com

Jaewoong Cho
KRAFTON
jwcho@krafton.com

Albert No
Hongik University
albertno@hongik.ac.kr

Ernest K. Ryu
Seoul National University
ernestryu@snu.ac.kr

Abstract

We propose LoRA modules as a replacement for the time and class embeddings of the U-Net architecture for diffusion probabilistic models. Our experiments on CIFAR-10 show that a score network trained with LoRA achieves competitive FID scores while being more efficient in memory compared to a score network trained with time and class embeddings.

1 Introduction

Diffusion probabilistic models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021) perform image generation using a score network trained to approximate the time-dependent score function. For class-conditional image generation, the score function is further conditioned on the image class, and classifier-free guidance (Ho & Salimans, 2021) is often used. Within the score network architecture, time information is incorporated by a sinusoidal time embedding inspired by the positional encoding of transformer architectures (Vaswani et al., 2017), and class information is incorporated by a trainable token embedding. These architectural choices are certainly very effective, as evidenced by the quality of the state-of-the-art diffusion models, but they do feel arbitrary, and their optimality has certainly not been adequately explored.

Meanwhile, Hu et al. (2022) recently proposed Low-Rank Adaptation (LoRA) as a parameter-efficient fine-tuning mechanism for Large Language Models (LLM). Given a pre-trained set of dense weight W, LoRA parameterizes the fine-tuning update ΔW with a low-rank factorization. LoRA demonstrates strong performance even with low ranks, as small as $r = 4$ or even $r = 1$, and its usage has even been extended to fine-tuning diffusion models (Ryu, 2023; Go et al., 2023; Shi et al., 2023; Smith et al., 2023; Yang et al., 2023; Wang et al., 2023).

In this paper, we propose *TimeLoRA* and *ClassLoRA*, LoRA modules that replace the time and class embeddings of the U-Net architecture for diffusion probabilistic models. Our experiments on CIFAR-10 show that a score network trained with LoRA achieves competitive FID scores while being more efficient in memory compared to a score network trained with time and class embeddings.

2 Background

2.1 Diffusion probabilistic models

Diffusion probabilistic models (DPM) (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021) generate images by reversing the forward diffusion process. The forward process iteratively destroys

Neural Information Processing Systems (NeurIPS) 2023 Workshop on Diffusion Models.

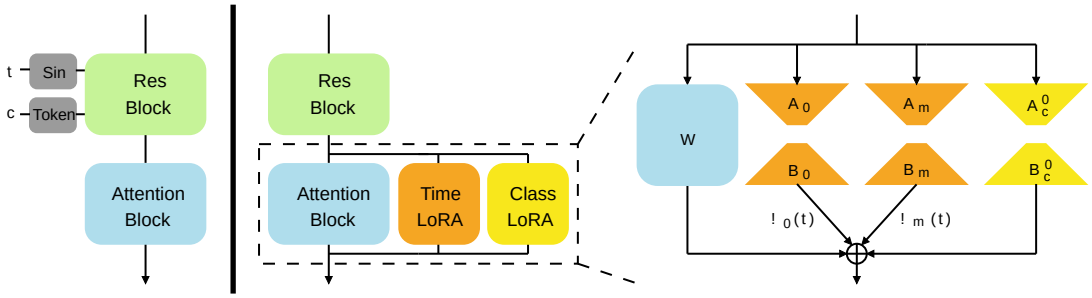


Figure 1: (Left) Conventional time and class embeddings. (Right) TimeLoRA and ClassLoRA

an image by injecting noise, and DPMs learn to progressively reconstruct the image by removing the injected noise. In this paper, we consider the speci c instance DDPM, which we further review in Appendix A.

Time embedding. DPMs generally use a variant of the U-Net architecture (Ronneberger et al., 2015) with time information incorporated with sinusoidal embeddings, inspired by the positional encodings of transformer architectures (Vaswani et al., 2017). Speci cally, each level of the U-Net has residual blocks with convolutional layers, and the sinusoidal time embedding goes through trainable linear layers to produce values for scale and shift operations injected into the residual blocks. On the other hand, the pixel-wise attention layers of the U-Net architecture used for DPMs do not have time information injected directly into them. See Figure 1. Alternatives to the U-Net architecture, such as DiT (Peebles & Xie, 2022) and RIN (Jabri et al., 2023), which use transformer and recurrent neural network architectures, also depend on sinusoidal embeddings to incorporate time.

Class embedding. For class-conditional generation, the score networks of DPMs are conditioned on class labels, and classi er-free guidance of Ho & Salimans (2021) is widely used. Typically class information is injected into the residual blocks with convolutional layers alongside time embedding, as depicted in Figure 1.

2.2 Low-Rank Adaptation (LoRA)

LoRA (Hu et al., 2022) is a parameter-ef cient ne-tuning method originally designed for the transformer architectures of LLMs. To ne-tune a pre-trained dense weight matrix $W_{\text{trained}} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, LoRA parameterizes the ne-tuning update ΔW with a low-rank factorization

$$W + \Delta W = W + BA;$$

where $B \in \mathbb{R}^{d_{\text{out}} \times r}$, $A \in \mathbb{R}^{r \times d_{\text{in}}}$, and $r \ll \min\{d_{\text{in}}, d_{\text{out}}\}$.

LoRA and diffusion models. Recently the usage of LoRA has been extended to ne-tuning diffusion models (Ryu, 2023; Go et al., 2023; Shi et al., 2023; Smith et al., 2023; Yang et al., 2023; Wang et al., 2023). To the best of our knowledge, however, our work is the rst attempt to use LoRA as part of the base architecture for diffusion models.

LoRA and multi-task learning. In Audibert et al. (2023), LoRA is used in multi-task learning. More speci cally, they propose to use frozen task-agnostic weights shared across different tasks and task-speci c low-rank updates trained for each task. Since diffusion models also use a shared U-Net with time-dependent embedding layers, they can also be interpreted as an instance of multi-task learning with one task per timestep.

3 TimeLoRA

Consider DDPM with T (discrete) timesteps. We replace the time embedding of the score network with TimeLoRA. Typically, time information is injected into the residual blocks containing convolutional layers. We instead apply TimeLoRA to the dense weights in attention blocks. See Figure 1.

3.1 Non-compositional LoRA

We instantiate T independent LoRAs $A_1, A_2, \ldots, A_T$ and $B_1, B_2, \ldots, B_T$ and use the dense weight W_t of the attention blocks parameterized as

$$W_t = W + \Delta W(t) = W + B_t A_t$$

for $t = 1, \ldots, T$. However, this approach has two drawbacks. First, since T is typically large (≥ 4000), the T LoRAs occupy significant memory. Second, since each LoRA (A_t, B_t) is trained independently, it disregards the fact the LoRA of nearby time steps would likely be correlated/similar.

3.2 Compositional LoRAs

To address the drawbacks of the previous approach, we use a composition of $m+1$ basis LoRAs, $A_0, A_1, \ldots, A_m$ and $B_0, B_1, \ldots, B_m$, where $m \leq T$. Each basis LoRA (A_i, B_i) corresponds to time t_i for $1 \leq t_0 < \cdots < t_m \leq T$. As illustrated in Figure 1, we then use the dense weight W_t of the attention blocks parameterized as

$$W_t = W + \Delta W(t) = W + \sum_{i=0}^{m} \lambda_i(t) B_i A_i,$$

where $\lambda_i(t)$ is the time-dependent weight for the basis LoRA (A_i, B_i) for $i = 0, \ldots, m$. To clarify, the trainable parameters for each linear layer W is $A_0, A_1, \ldots, A_m, B_0, B_1, \ldots, B_m$. In particular, there is no pre-training, and W is concurrently trained with LoRAs. This approach was loosely inspired by LoraHub (Huang et al., 2023). It now remains to specify our choice of $\lambda_i(t)$ for $i = 0, \ldots, m$.

Linear interpolation. Since the score network is a continuous function of t, we expect $\lambda_i(t) \approx \lambda_i(t')$ if $t \approx t'$. Therefore, we first encode t by linearly interpolating the two closet basis LoRAs: for $t_i \leq t < t_{i+1}$, we choose $\lambda_i(t) = \frac{t_{i+1} - t}{t_{i+1} - t_i}$, $\lambda_{i+1}(t) = \frac{t - t_i}{t_{i+1} - t_i}$, and $\lambda_j(t) = 0$ for all $j \notin \{i, i+1\}$.

Trainable composition weights. We also experiment with letting $\lambda_i(t)$ be general trainable weights. We explore three different initialization schemes:

- Gaussian: $\lambda_i(t) \sim \mathcal{N}(\mu, \sigma^2)$ IID
- Onehot: $\lambda_i(t) = 1_{t_i \leq t < t_{i+1}}$
- Linear: initialize λ to match linear interpolation

4 ClassLoRA

Consider DDPM with C classes. As with TimeLoRA, we replace the class embedding with Class-LoRA Typically, class information is injected into the residual blocks containing convolutional layers. We instead apply ClassLoRA to the dense weights in attention blocks. See Figure 1.

Since C is small for CIFAR-10 and the correlations between different classes are not clear, we only use the non-compositional ClassLoRA:

$$W_c = W + \Delta W(c) = W + B_c^0 A_c^0$$

for $c = 1, \ldots, C$. When C is large, such as in the case of ImageNet1k, we may use compositional ClassLoRA, but we leave the investigation of this to future work.

5 Experiments

As a proof-of-concept experiment, we implement TimeLoRA and ClassLoRA on IDDPM (Nichol & Dhariwal, 2021) (and remove the time and class embeddings) and train the score network with the CIFAR-10 dataset. For each setting, we train the model for 500k iterations with 128 batch size. We sample 50k samples every 50k iterations and report the best FID score (Heusel et al., 2017).

For simplicity, we assume that m divides $T - 1$ and that the basis LoRAs are uniformly spaced, i.e., $t_i = 1 + ik$, where $k = (T - 1)/m$. As in Ryu (2023), we set the rank of LoRA $r = 4$. Also, unlike the prior works that use LoRA as a ne-tuning method, we train the U-Net alongside TimeLoRA and ClassLoRA from scratch. (There is no pre-training.) For detailed experiment settings, refer to Appendix C.1.

5.1 Replacing time embedding with TimeLoRA

We rst compare IDDPM with time embedding and with TimeLoRA (with no class conditioning). As reported in Table 1, with only 88-90% of the parameter count, utilizing TimeLoRA yields a competitive FID score. We provide randomly selected samples in Figure 4 in Appendix C.4.

Table 1: Sampling of unconditional CIFAR-10 images with and without TimeLoRA

	T	m	r	Type	FID	# Params
IDDPM	4000			time emb. (no LoRA)	3.69	52546438
IDDPM + T-LoRA	1000	1000	4	non-compositional	4.78	137417350
IDDPM + T-LoRA	4001	10	4	linear interpolation	3.79	46271110
IDDPM + T-LoRA	4001	10	4	train (Gaussian)	3.72	47591440
IDDPM + T-LoRA	4001	10	4	train (onehot)	3.61	47591440
IDDPM + T-LoRA	4001	10	4	train (linear)	3.64	47591440

5.2 Replacing time and class embedding with TimeLoRA and ClassLoRA

Next, we compare class-conditional IDDPM with time and class embedding and with TimeLoRA and ClassLoRA. As reported in Table 2, TimeLoRA and ClassLoRA seem to be more compatible with the classi er-free guidance, showing signi cant improvement over unconditional image generation with only 90-92% of the parameter count compared to the conventional time and class embeddings. We provide randomly selected samples in Figure 5 in Appendix C.5.

Table 2: Sampling of class conditional CIFAR-10 images with and without Time and ClassLoRA.

	T	$(m; C)$	r	Type	FID	# Params
IDDPM	4000			time & class emb. (no LoRA)	3.38	52551558
IDDPM + T&C-LoRA	4001	(10, 10)	4	linear interpolation	3.08	47192710
IDDPM + T&C-LoRA	4001	(10, 10)	4	train (Gaussian)	2.85	48513040
IDDPM + T&C-LoRA	4001	(10, 10)	4	train (onehot)	2.93	48513040
IDDPM + T&C-LoRA	4001	(10, 10)	4	train (linear)	2.91	48513040

5.3 Using Time and Class LoRAs alongside with time and class embeddings

Finally, we investigate combining Time and Class LoRAs with conventional time and class embeddings. This way, both the residual blocks containing convolutional layers and the attention blocks are conditioned by time and class. We run 3 independent experiments for each setting, and report the best FID score in Table 3.

T-LoRA and t-embedding. As reported in Table 3, unconditional IDDPM with both time embedding and TimeLoRA yields improved FID score with only a 2% addition of the parameter count compared to using time embedding alone. Moreover, adding time embedding shows improvement compared to using only TimeLoRA consistently over all composition schemes. So, time embedding and TimeLoRA are compatible, and combining them yields better results than using them individually. We provide randomly selected samples in Figure 4 in Appendix C.4.

T&C-LoRA and t&c-embedding. Combining Time and Class LoRAs to conditional IDDPM with both time and class embeddings improves FID scores with only a 0.32% addition of the parameter

count. However, compared to using Time and Class LoRAs alone, combining LoRAs with trainable weights and conventional embeddings shows lower FID score. Moreover, using LoRA with linear interpolation shows better result than using LoRA with trainable composition weights. These results might indicate that using Time and Class LoRAs with both time and class embeddings endows model with enough capacity to over t, or even memorize (Yoon et al., 2023), causing a decline in the FID score.

Table 3: IDDPM sampling of CIFAR-10 images with LoRAs and conventional embeddings.

	T	(m; C)	r	Type	FID	# Params
T-LoRA + tembed	4001	(10, -)	4	linear interpolation	3.38	53560198
T-LoRA + tembed	4001	(10, -)	4	train (Gaussian)	3.54	54880528
T-LoRA + tembed	4001	(10, -)	4	train (onehot)	3.48	54880528
T-LoRA + tembed	4001	(10, -)	4	train (linear)	3.37	54880528
T&C-LoRA + t&c-embed	4001	(10, 10)	4	linear interpolation	3.01	54486918
T&C-LoRA + t&c-embed	4001	(10, 10)	4	train (Gaussian)	3.28	55807248
T&C-LoRA + t&c-embed	4001	(10, 10)	4	train (onehot)	3.11	55807248
T&C-LoRA + t&c-embed	4001	(10, 10)	4	train (linear)	3.12	55807248

6 Future Works

We propose TimeLoRA and ClassLoRA as a replacement for the conventional class and time embeddings in the score network architecture of diffusion probabilistic models and experimentally demonstrate their promise. There are many avenues of future work to investigate the full potential of LoRA as a general mechanism to condition score networks. For instance, following Alfassy et al. (2022), we may apply LoRA also to convolution layers to further improve the performance. Following Choi et al. (2022), we may identify "important" time steps and use larger values of r for them, as opposed to using uniformly sized LoRAs. Seeing whether TimeLoRA and ClassLoRA are compatible with other state-of-the-art models such as EDM (Karras et al., 2022) and EDM-G++ (Kim et al., 2022) would be interesting. Finally, using LoRA for text-to-image models such as Stable Diffusion (Rombach et al., 2022), Imagen (Saharia et al., 2022), and DALL·E (Ramesh et al., 2022) would be a valuable research direction, but it would unfortunately be out of practical reach of most academic research labs.

References

A. Alfassy, A. Arbelle, O. Halimi, S. Harary, R. Herzig, E. Schwartz, R. Panda, M. Dol , C. Auer, P. W. J. Staar, K. Saenko, R. Feris, and L. Karlinsky. FETA: Towards specializing foundational models for expert task applications. In *NeurIPS Datasets and Benchmarks Track*, 2022.

A. Audibert, M. R. Amini, K. Usevich, and M. Clausel. Low-rank updates of pre-trained weights for multi-task learning. In *ACL*, 2023.

J. Choi, J. Lee, C. Shin, S. Kim, H. Kim, and S. Yoon. Perception prioritized training of diffusion models. In *CVPR*, 2022.

H. Go, Y. Lee, J.-Y. Kim, S. Lee, M. Jeong, H. S. Lee, and S. Choi. Towards practical plug-and-play diffusion models. In *CVPR*, 2023.

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. In *NeurIPS*, 2017.

J. Ho and T. Salimans. Classi er-free diffusion guidance. In *NeurIPS Workshop on Deep Generative Models and Downstream Applications*, 2021.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

E. J. Hu, y. shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

C. Huang, Q. Liu, B. Y. Lin, T. Pang, C. Du, and M. Lin. LoraHub: Efficient cross-task generalization via dynamic LoRA composition. *arXiv preprint arXiv:2307.13269*, 2023.

A. Jabri, D. J. Fleet, and T. Chen. Scalable adaptive computation for iterative generation. In *ICML*, 2023.

T. Karras, M. Aittala, T. Aila, and S. Laine. Elucidating the design space of diffusion-based generative models. *arXiv preprint arXiv:2206.00364*, 2022.

D. Kim, Y. Kim, W. Kang, and I.-C. Moon. Refining generative process with discriminator guidance in score-based diffusion models. *arXiv preprint arXiv:2211.17091*, 2022.

A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, 2021.

W. Peebles and S. Xie. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022.

A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with CLIP latents. *arXiv preprint arXiv:2204.06125*, 2022.

R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

S. Ryu. Low-rank adaptation for fast text-to-image diffusion fine-tuning. https://github.com/cloneofsimo/lora, 2023.

C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 2022.

Y. Shi, C. Xue, J. Pan, W. Zhang, V. Y. Tan, and S. Bai. DragDiffusion: Harnessing diffusion models for interactive point-based image editing. *arXiv preprint arXiv:2306.14435*, 2023.

J. S. Smith, Y.-C. Hsu, L. Zhang, T. Hua, Z. Kira, Y. Shen, and H. Jin. Continual diffusion: Continual customization of text-to-image diffusion with c-lora. *arXiv preprint arXiv:2304.06027*, 2023.

J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015.

Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Z. Wang, C. Lu, Y. Wang, F. Bao, C. Li, H. Su, and J. Zhu. ProlificDreamer: High-fidelity and diverse text-to-3D generation with variational score distillation. *arXiv preprint arXiv:2305.16213*, 2023.

S. Yang, Y. Zhou, Z. Liu, and C. C. and Loy. Rerender a video: Zero-shot text-guided video-to-video translation. In *ACM SIGGRAPH Asia*, 2023.

T. Yoon, J. Y. Choi, S. Kwon, and E. K. Ryu. Diffusion probabilistic models generalize when they fail to memorize. In *ICML 2023 Workshop on Structured Probabilistic Inference & Generative Modeling*, 2023.

A Denoising Diffusion Probabilistic Model (DDPM)

DDPM (Ho et al., 2020) generates an image by iteratively restoring the data from a noise. Such restoration process is defined by the reverse process of the forward diffusion process: given a data $x_0 \quad q_0$, progressively inject noise to x_0 by

$$q(x_t|x_{t\ 1}) = \mathcal{N}\quad {}^{\mathsf{p}}\overline{1\quad \beta_t}x_{t\ 1}, \beta_t \mathsf{I}$$

for $t = 1, \quad , T$ and $0 < \beta < 1$. Then assuming T is sufficiently large, one can explicitly compute the reverse process as

$$q(x_{t\ 1}|x_t, x_0) = \mathcal{N}\quad \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t \mathsf{I}$$

where

$$\tilde{\mu}_t(x_t, x_0) = \frac{p\overline{\tilde{\alpha}_{t\ 1}}\beta_{t\ 1}}{1\quad \tilde{\alpha}_t} x_0 + \frac{p\overline{\tilde{\alpha}_t}(1\quad \tilde{\alpha}_{t\ 1})}{1\quad \tilde{\alpha}_t} x_t,$$

and $\alpha_t = 1\quad \beta_t, \tilde{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. Hence DDPM $p\quad$ parameterzied by θ is trained to estimate the reverse process by

$$p\ (x_{t\ 1}|x_t) = \mathcal{N}\ (\mu\ (x_t, t), \Sigma\ (x_t, t))$$

with the objective

$$L = \mathop{\mathbb{E}}_{q}\left[\sum_{t=1}^{T} D_{KL}(q(x_{t\ 1}|x_t, x_0)||p\ (x_{t\ 1}|x_t)) \quad \log p\ (x_0|x_1) \right].$$

In practice, assuming $\Sigma\ $ is fixed, one can train $\epsilon\ $ by reformulate the objective as

$$L = \mathop{\mathbb{E}}_{t;x_0\quad q_0;\quad \mathcal{N}(0;\mathbf{I})} \left[\|\epsilon\quad \epsilon\ (x_t, t)\|^2 \right],$$

with $x_t = {}^{p}\overline{\tilde{\alpha}_t} x_0 + {}^{p}\overline{1\quad \tilde{\alpha}_t}\epsilon$.

B Illustration of UNet

We provide an illustration of UNet used in IDDPM. Note as demonstrated in Figure 1, conventional time and class embeddings are added to Res blocks, whereas TimeLoRA and ClassLoRA are attached to attention blocks.

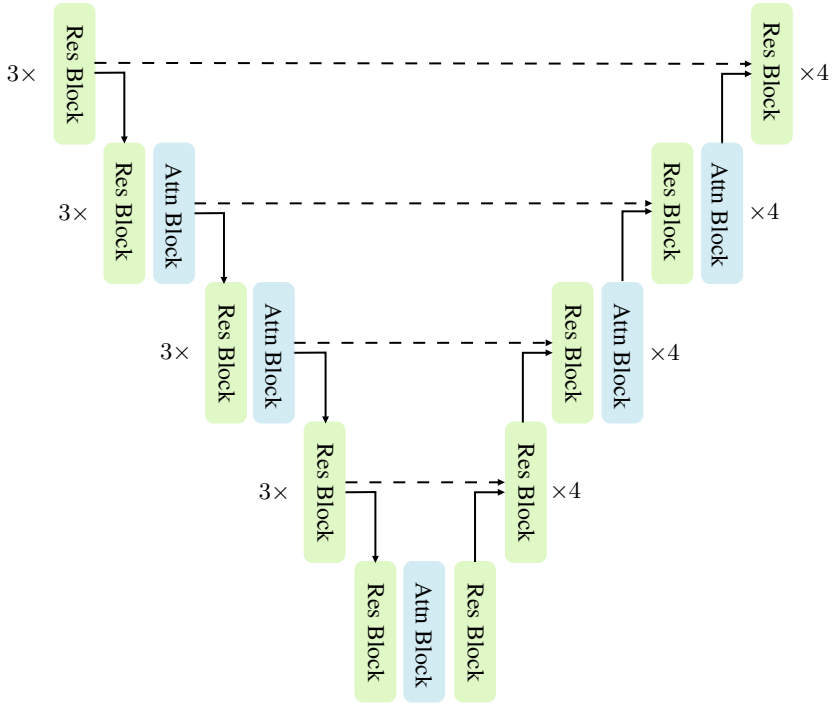


Figure 2: Illustration of UNet architecture

C Experiment

C.1 Experiment settings

IDDPM training flags. Here we provide the training setting used for IDDPM experiments:

```
MODEL_FLAGS="--image_size 32 --num_channels 128 --num_res_blocks 3 --learn_sigma True --dropout 0.3"
DIFFUSION_FLAGS="--diffusion_steps 4000 --noise_schedule cosine"
TRAIN_FLAGS="--lr 1e-4 --batch_size 128"
```

Initialization. Here we provide the initialization details:

- Dense weights: W in the attention block is initialized as in IDDPM (Nichol & Dhariwal, 2021)
- LoRA: As in Hu et al. (2022), we initialize A_i, A_c^0 $N(0, 1/r)$, and B_i, B_c^0 0
- Gaussian init: We use the default `init._normal` provided by Pytorch; $\mu = 0, \sigma = 1$

C.2 Results on varying r and m

We report some preliminary experiments on varying LoRA rank r and the number of basis LoRA m in Table 4. Note in this experiment, we sample 50k samples every 100k iterations and report the best FID score. Similar to the observation by Hu et al. (2022) increasing r does not translate into improved performance. Surprisingly, similar observation was made even for m.

Table 4: Unconditional IDDPM + TimeLoRA sampling with varying m and r (CIFAR-10)

	m	r	FID	# Params
train (onehot)	10	4	3.61	47591440
	10	8	3.77	48605200
train (linear)	10	4	3.64	47591440
	10	8	3.69	48605200

(a) Varying r

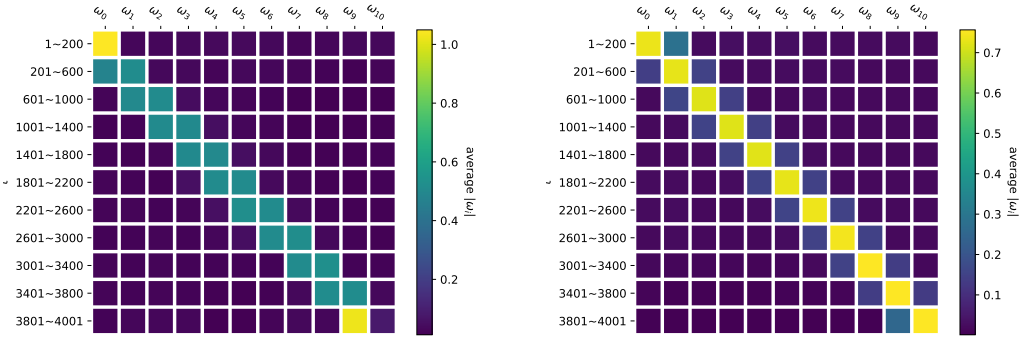
	m	r	FID	# Params
train (onehot)	10	4	3.61	47591440
	20	4	3.64	48605200
	40	4	3.76	53957140
train (linear)	10	4	3.64	47591440
	20	4	4.03	48605200
	40	4	3.76	53957140

(b) Varying m

C.3 Visualization of composition weights

We visualize the trained composition weights of LoRA from a randomly selected attention block in Figure 3. Note we only report onehot initialization, and linear initialization, because visualization of Gaussian initialization did not provide any meaningful implication. We average the absolute values of $\omega_i(t)$ for ts near each basis LoRA. For example, on the first row, we report $\frac{1}{200}\sum_{t=1}^{200}\omega_i(t)$, and on the second row, we report $\frac{1}{400}\sum_{t=201}^{600}\omega_i(t)$, and so on.

From the visualization, we can see that time t is encoded by the composition of *closet* basis LoRAs. This justifies our intuition from Section 3.2: DDPM should perform similar task for nearby timesteps t and t'.

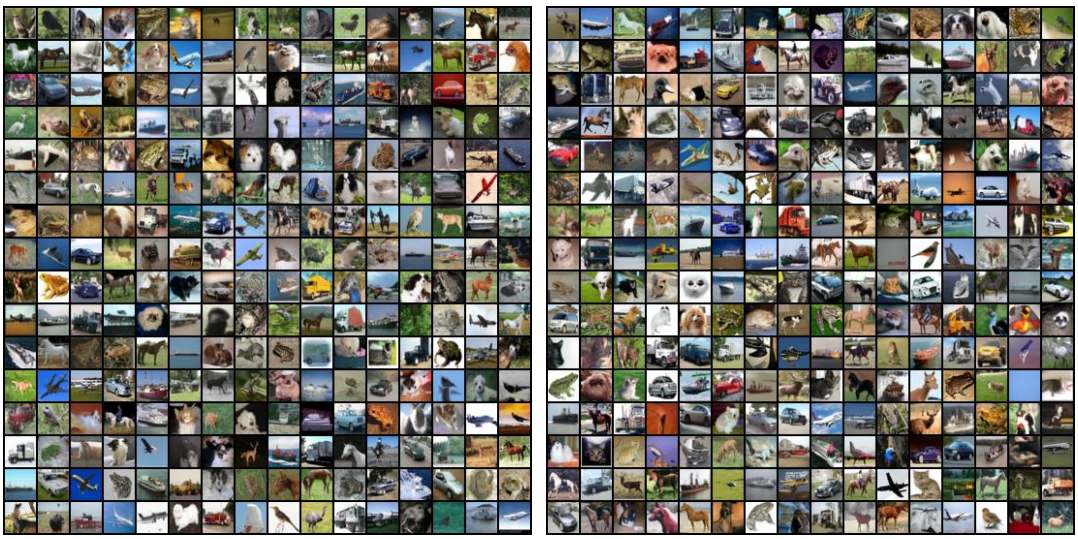


(a) Trained with onehot initialization

(b) Trained with linear initialization

Figure 3: Visualization of trained composition weights

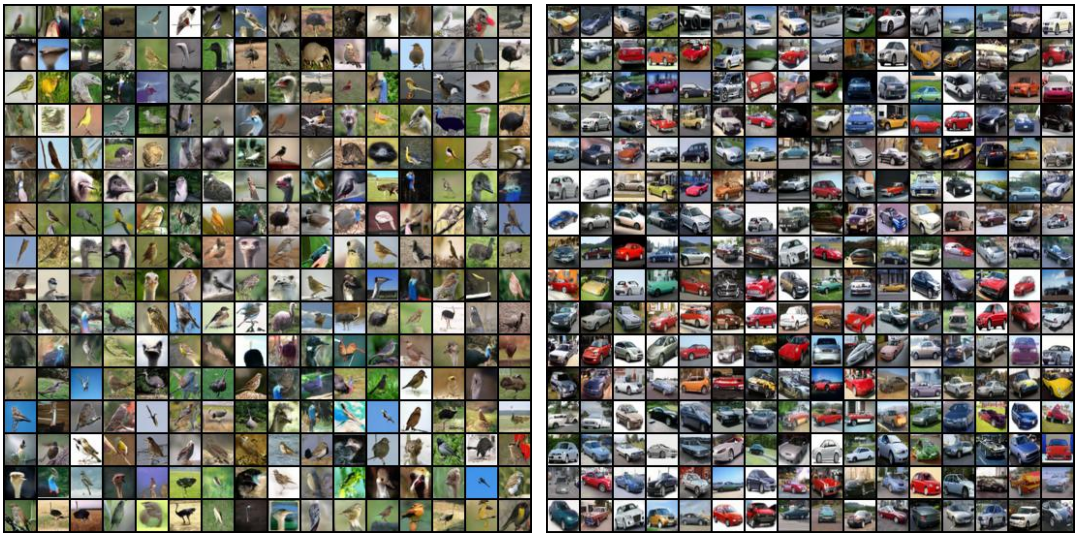
C.4 Unconditional samples



(a) TimeLoRA (b) Time embedding and TimeLoRA

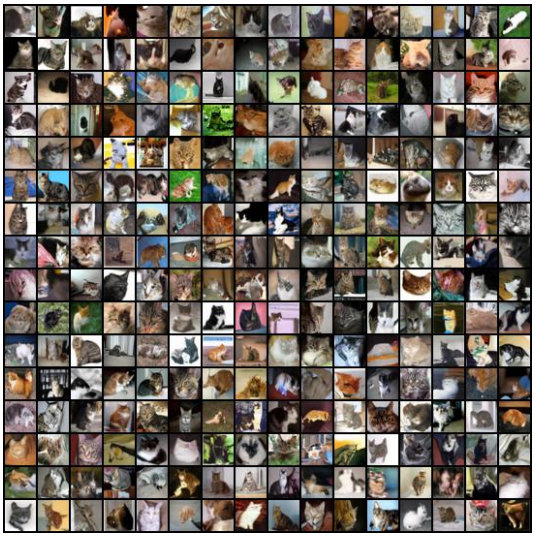
Figure 4: Unconditional CIFAR-10 image samples

C.5 Class-conditional samples with TimeLoRA and ClassLoRA

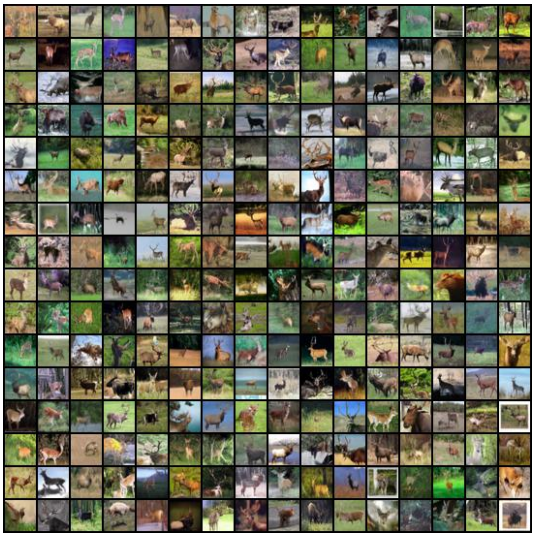


(a) Bird (b) Car

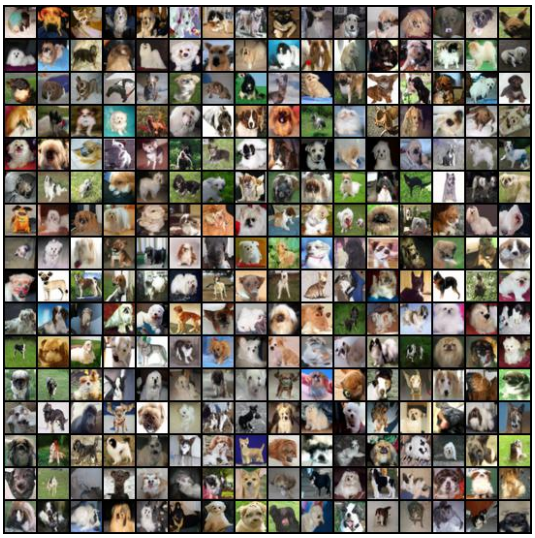
Figure 5: Class conditional CIFAR-10 image samples using TimeLoRA and ClassLoRA



(c) Cat



(d) Deer



(e) Dog

(f) Frog

Figure 5: Class conditional CIFAR-10 image samples using TimeLoRA and ClassLoRA

(g) Horse

(h) Plane

(i) Ship

(j) Truck

Figure 5: Class conditional CIFAR-10 image samples using TimeLoRA and ClassLoRA