Transformed Low-rank Adaptation via Tensor Decomposition and Its Applications to Text-to-image Models

Supplementary Material

Contents

A. Details and more results in Sec. 3.2

In this section, we present the detailed settings for experiments of approximating fine-tuned weights in Sec. 3.2. Moreover, we provide results of more layers in the SDXL(-Inpaint) models, and models from other domains, e.g., large language models.

First, we present the experimental settings. For results in Fig. 2, the pre-trained weight $\boldsymbol{W}_0$ is taken from the pre-trained SDXL model and the key for the layer is `mid_block.attentions.0.transformer_blocks.0.attn1.to_k`. The shape of $\boldsymbol{W}_0$ is 1280×1280. The detailed settings for three cases in Fig. 2 are as follows.

1. **Additive low-rank difference.** The target weight is computed by $\boldsymbol{W}_* = \boldsymbol{W}_0 + \sigma \boldsymbol{B}_* \boldsymbol{A}_*$, where the rank of $\boldsymbol{B}_* \boldsymbol{A}_*$ is 128. Each element of $\boldsymbol{B}_*$ and $\boldsymbol{A}_*$ is drawn i.i.d. from $\mathcal{N}(0, 1)$. After $\boldsymbol{B}_*$ and $\boldsymbol{A}_*$ are generated, we compute σ to scale this difference part so that $\boldsymbol{W}_* - \boldsymbol{W}_0$ has the same standard deviation with the case where $\boldsymbol{W}_*$ is the true fully fine-tuned weight from SDXL-Inpaint.
2. **Orthogonal rotation.** The target weight is computed by $\boldsymbol{W}_* = \boldsymbol{W}_0 \boldsymbol{T}_*$, where $\boldsymbol{T}_*$ is a random orthogonal matrix. To generate this orthogonal matrix, we firstly generate a random matrix $\boldsymbol{X} = \boldsymbol{I}_{1280} + \boldsymbol{F}$, where each element of $\boldsymbol{F}$ is drawn i.i.d. from $\mathcal{N}(0, 0.05^2)$. Then, $\boldsymbol{T}_*$ is computed by the QR decomposition of $\boldsymbol{X}$.
3. **True fully fine-tuned weight.** The target weight $\boldsymbol{W}_*$ is taken from the SDXL-Inpaint model [56] in the same layer with $\boldsymbol{W}_0$.

All approximation methods are optimized by minimizing the Mean Squared Error (MSE) with Adam optimizer using learning rate $1\mathrm{e}{-3}$ for 500 epochs, which is sufficient for them to converge. The settings of these methods are as follows.

I. **OFT.** We take the block diagonal structure and Cayley transform as in [47]. We vary the block size to get different parameter sizes.
II. **LoRA.** We vary the rank to get different parameter sizes.
III. **TR.** We use the TR form defined in Eq. (5) to replace the low-rank matrix decomposition in Eq. (1). The size of the pre-trained weight is 1280×1280. We reshape it into $1280 = 8 \times 8 \times 4 \times 5$ and choose different TR ranks to get different parameter sizes.
IV. **OFT + LoRA.** For OFT, we set the block size to 5, and combine it with LoRA of different ranks.
V. **TRM + LoRA.** We choose TRM with rank 1, and combine it with LoRA of different ranks.
VI. **TRM + TR.** We choose TRM with rank 1, and combine it with TR of different ranks.

Then, we provide results of different layers in the SDXL(-Inpaints) models in Fig. 6. To investigate the potential application of our method in large language models, we also conduct similar investigations in Llama2 7B and Llama2-chat 7B models [57], as shown in Fig. 7. The size of SDXL weights is 1280×1280, while the size of the Llama2 7B weights is 4096×4096. For both models, we have similar observations as in Sec. 3.2. Moreover, in the SDXL model, we find the effect of adding the transform is more significant for *key* and *query* weights. It would be interesting to analyze this effect more theoretically in the future. Besides, compared to SDXL, the TR structure appears to be more effective in the Llama2 models. This may be because that the tensor decomposition is more suitable for compression of weight matrices with larger sizes. A future direction would be exploration of our method in fine-tuning large language models.

B. Proposed model

B.1. Expressiveness of TRM

In this subsection, we present details of the simulation study in Fig. 3, Sec. 3.3. To empirically compare TRM with Butterfly matrices in BOFT, we randomly generate orthogonal matrices and evaluate the approximation error. In specific, we generate a random matrix $\boldsymbol{X}$ of shape 512×512, where each element is i.i.d. from unit Normal distribution $\mathcal{N}(0, 1)$. Then, the target orthogonal matrix $\boldsymbol{T}_*$ is obtained from the QR decomposition of $\boldsymbol{X}$. We approximate $\boldsymbol{T}_*$ using BOFT

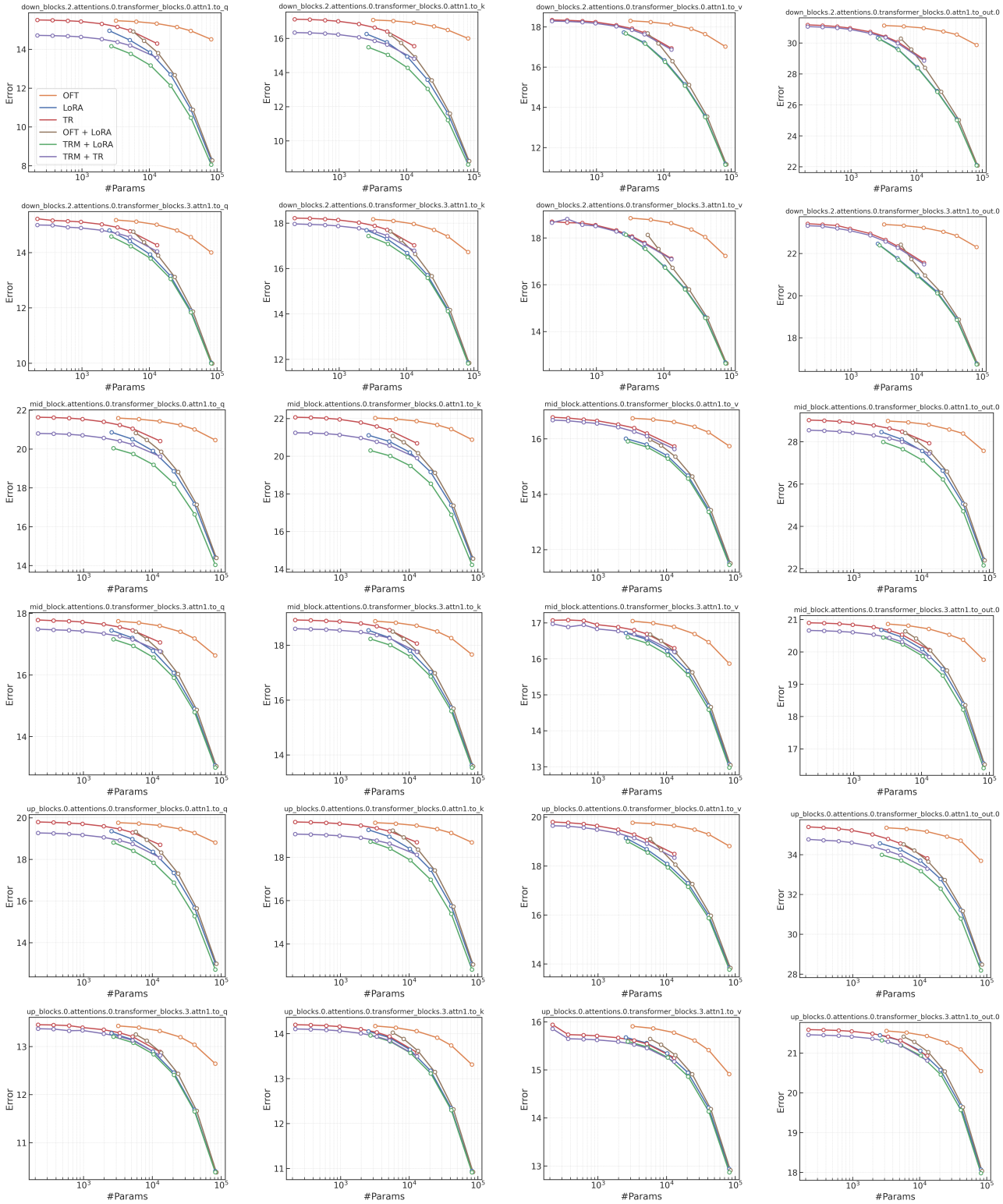

Figure 6. Simulation on different layers of the SDXL and SDXL-Inpaint models.

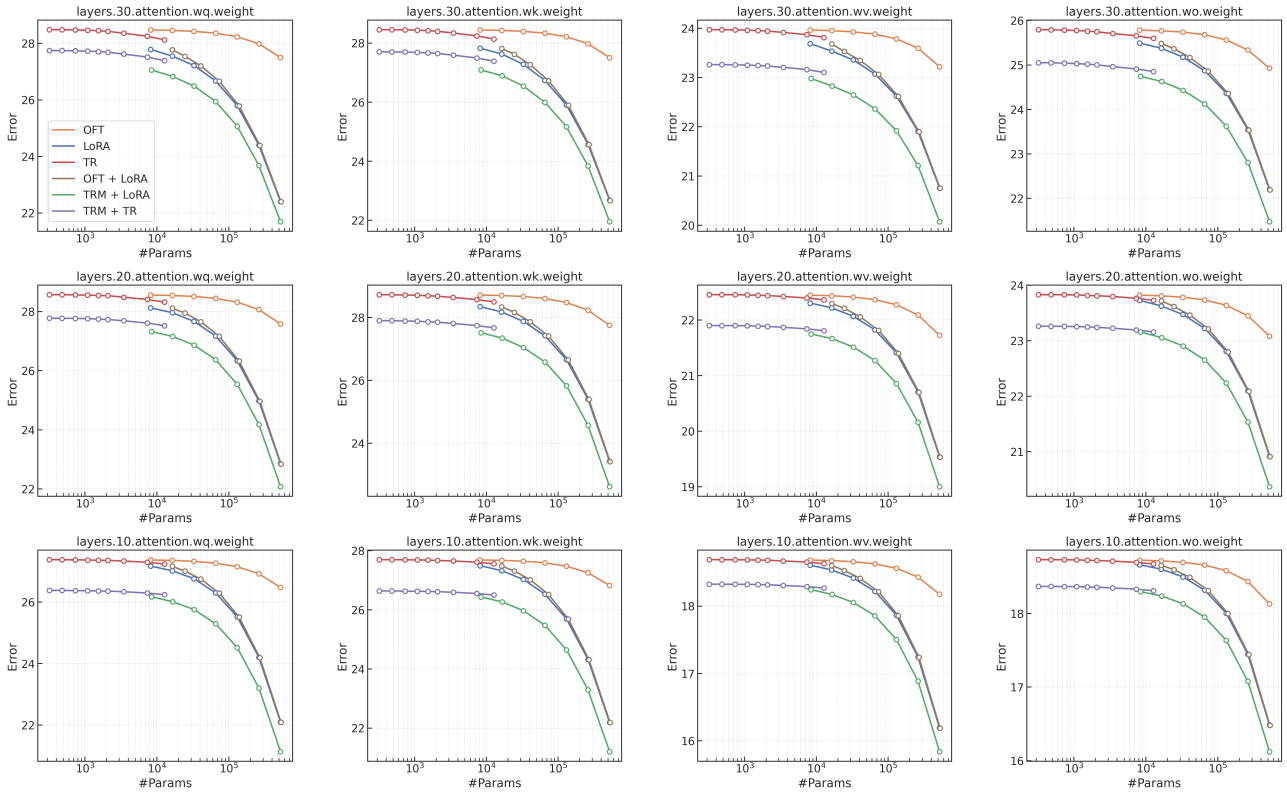

Figure 7. Simulation on different layers of the Llama2 7B and Llama2-chat 7B models.

and TRM by minimizing the MSE with Adam optimizer.

For BOFT, we test 2, 4, and 16 Butterfly factors, as in [32]. For each setting, we test the number of diagonal blocks ranging from 1 to 9, 1 to 8, and 1 to 6, denoted as BOFT(1:9, 2), BOFT(1:8, 4), and BOFT(1:6, 16) respectively. For TRM, we test different tensor shape, namely, $4 \times 4 \times 4 \times 8$, $8 \times 8 \times 8$, and 16×32. Then, we set different TR ranks to obtain results of different parameter sizes. In Fig. 3, we find the results of TRM are robust to the choices of tensor sizes, and are consistently better than BOFT. Besides, even there is no orthogonal constraints for TRM, it can approximate orthogonal matrices well.

B.2. Proof of Props. 1 and 2

In this subsection, we provide the derivation of Props. 1 and 2.

Proof of Prop. 1. According to the definition of TRM in Eq. (4), we have

$$\boldsymbol{T}[\overline{i_1 \cdots i_D}, \overline{j_1 \cdots j_D}] = \operatorname{tr}(\mathbf{A}^1[i_1, j_1, :, :] \cdots \mathbf{A}^D[i_D, j_D, :, :]).$$

Following the initialization in Prop. 1, for $d = 1, \ldots, D$, if $i_d = j_d$, all the elements of $\mathbf{A}^d[i_d, j_d, :, :]$ equal to $1/R$, else zero. Then we can discuss the diagonal and non-diagonal elements of $\boldsymbol{T}$ separately.

1. For non-diagonal elements, i.e., $\overline{i_1 \cdots i_D} \neq \overline{j_1 \cdots j_D}$, there is at least one sub-index $i_d \neq j_d$. Therefore, $\boldsymbol{T}[\overline{i_1 \cdots i_D}, \overline{j_1 \cdots j_D}] = 0$, since $\mathbf{A}^d[i_d, j_d, :, :] = \mathbf{0}$ if $i_d \neq j_d$.
2. For diagonal elements, i.e., $\overline{i_1 \cdots i_D} = \overline{j_1 \cdots j_D}$, we have $i_d = j_d, \forall d = 1, \ldots, D$. Now the core tensors become $\mathbf{A}^d[i_d, j_d, :, :] = \mathbf{1}_{R \times R}/R, \forall d = 1, \ldots, D$ and $i_d, j_d = 1, \ldots, I_d$, where $\mathbf{1}_{R \times R}$ is a matrix of shape $R \times R$ with all elements being one. Therefore, $\boldsymbol{T}[\overline{i_1 \cdots i_D}, \overline{j_1 \cdots j_D}] = 1$, since

$$\operatorname{tr}\left(\frac{\mathbf{1}_{R \times R}}{R} \cdots \frac{\mathbf{1}_{R \times R}}{R}\right) = 1.$$

Therefore, $\boldsymbol{T}$ is an identity matrix.

Proof of Prop. 2. For simplicity, in this proof, we denote $\mathbf{A}^d[i_d, j_d] = \mathbf{A}^d[i_d, j_d, :, :]$ and $\mathbf{B}^d[i_d, j_d] = \mathbf{B}^d[i_d, j_d, :, :]$. Suppose $\boldsymbol{X}$ has shape $I \times K$ with $I = \prod_{d=1}^D I_d$ and $K = \prod_{d=1}^D K_d$, and $\boldsymbol{Y}$ has shape $J \times K$ with $J = \prod_{d=1}^D J_d$. According to the definition of TRM in Eq. (4), we can com-

pute the product $\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{Y}^\intercal$ as follows,

$$
\begin{aligned}
&\boldsymbol{Z}[\overline{i_1 \cdots i_D}, \overline{j_1 \cdots j_D}] \\
&= \sum_{k_1,\ldots,k_D=1}^{K_1,\ldots,K_D} \boldsymbol{X}[\overline{i_1 \cdots i_D}, \overline{k_1 \cdots k_D}]\boldsymbol{Y}[\overline{j_1 \cdots j_D}, \overline{k_1 \cdots k_D}] \\
&= \sum_{k_1,\ldots,k_D=1}^{K_1,\ldots,K_D} \operatorname{tr}(\mathbf{A}^1[i_1, k_1] \cdots \mathbf{A}^D[i_D, k_D]) \\
&\qquad \cdot \operatorname{tr}(\mathbf{B}^1[j_1, k_1] \cdots \mathbf{B}^D[j_D, k_D]) \\
&= \sum_{k_1,\ldots,k_D=1}^{K_1,\ldots,K_D} \operatorname{tr}\Big\{(\mathbf{A}^1[i_1, k_1] \cdots \mathbf{A}^D[i_D, k_D]) \\
&\qquad \otimes (\mathbf{B}^1[j_1, k_1] \cdots \mathbf{B}^D[j_D, k_D])\Big\} \\
&= \sum_{k_1,\ldots,k_D=1}^{K_1,\ldots,K_D} \operatorname{tr}\Big\{(\mathbf{A}^1[i_1, k_1] \otimes \mathbf{B}^1[j_1, k_1]) \\
&\qquad \cdots (\mathbf{A}^D[i_D, k_D] \otimes \mathbf{B}^D[j_D, k_D])\Big\} \\
&= \operatorname{tr}\Bigg\{\sum_{k_1,\ldots,k_D=1}^{K_1,\ldots,K_D} \Big[(\mathbf{A}^1[i_1, k_1] \otimes \mathbf{B}^1[j_1, k_1]) \\
&\qquad \cdots (\mathbf{A}^D[i_D, k_D] \otimes \mathbf{B}^D[j_D, k_D])\Big]\Bigg\} \\
&= \operatorname{tr}\Bigg\{\sum_{k_1=1}^{K_1} (\mathbf{A}^1[i_1, k_1] \otimes \mathbf{B}^1[j_1, k_1]) \\
&\qquad \cdots \sum_{k_D}^{K_D} (\mathbf{A}^D[i_D, k_D] \otimes \mathbf{B}^D[j_D, k_D])\Bigg\},
\end{aligned}
$$

which follows a TR format. Therefore $\boldsymbol{X}\boldsymbol{Y}^\intercal = \mathrm{TRM}(\mathbf{C}^{1:D})$, where each core tensor $\mathbf{C}^d[i_d, j_d, :, :] = \sum_{l_d} (\mathbf{A}^d[i_d, l_d, :, :] \otimes \mathbf{B}^d[j_d, l_d, :, :])$.

To make $\boldsymbol{X}$ orthogonal, i.e., $\boldsymbol{X}\boldsymbol{X}^\intercal = \boldsymbol{I}$, we can regularize $\mathbf{C}^{1:D}$ according to the initialization scheme in Prop. 1.

C. Experiments

In this section, we provide experimental details and more results. All the experiments are conducted on single Nvidia H100 or A100 GPU with 80GB memory. The code is available at https://github.com/taozerui/tlora_diffusion.

C.1. Experimental details

Datasets. For the subject-driven generation, we use the DreamBooth dataset [52], which includes 30 subjects from 15 different classes. For text prompts, we follow the setting in Lee et al. [27], Zhang et al. [71]. For each subject, there are several images with 10 testing text prompts. The dataset is available at the Github repository[1] of Zhang et al. [71].

For the controllable generation task, we consider three tasks and two datasets. The settings basically follow Qiu et al. [47]. For the Landmark to Image (L2I) task, we use the CelebA-HQ [59] dataset. The whole dataset consists of $30k$ images of faces, captions generated by BLIP [29], and face landmarks detected by the `face-alignment` library [6]. The test set contains 2987 samples. For the Segmentation to Image (S2I) task, we use the ADE20K 2017 dataset [74], which consists of $20k$ training images, segmentations and captions generated by BLIP. The test set contains 2000 samples. For the Canny to Image (C2I) task, we also use the ADE20K dataset, where the canny edges are detected using the same detector as in Zhang et al. [68].

Baselines. We choose the following baselines, which are highly related to our work and competitive methods.

- LoRA [20], which is the original method.
- DoRA [31], which is a popular extension of LoRA. Moreover, it shares some similarities with our method, as discussed in Sec. 3.5.
- OFT [47], which applies orthogonal transforms for finetuning. It uses block diagonal transforms for parameter efficiency.
- BOFT [32], which extends OFT by using Butterfly matrices to construct dense transforms.
- ETHER [4], which adopts Householder reflection to parameterize orthogonal transforms. In particular, we adopt the implementation called ETHER+, which relaxes the orthogonal constraint and applies transforms on the left and right sides of the pre-trained weight. It is more flexible and has shown better results in Bini et al. [4].
- LoRETTA [65], which is an extension of LoRA. It further factorized LoRA matrices using TT decomposition for parameter efficiency.

LoRA, DoRA, OFT, and BOFT are provided in the `PEFT` library [36]. For ETHER[2] and LoRETTA[3], we apply their official implementations in our experiments.

General settings. For all the baselines and our model, we inject the trainable parameters into the attention modules on *key*, *value*, *query*, and *output* layers. This setting is consistent with previous works [4, 32, 47] for a fair comparison.

For the subject-driven generation, we fine-tune the SDXL [46] model using the Direct Consistency Optimization [DCO, 27] algorithm. Following previous works [27, 52], we set the batch size to 1 and use the AdamW optimizer with constant learning rates. For all methods, learning rates are tuned from $\{5\mathrm{e}{-4}, 1\mathrm{e}{-4}, 5\mathrm{e}{-5}\}$. We train the model for 20 epochs for each individual subject.

[1] https://github.com/phymhan/SODA-Diffusion
[2] https://github.com/mwbini/ether
[3] https://github.com/yifanycc/loretta

Table 3. Hyper-parameters of the subject-driven generation experiment. For OFT, b means the block size. For BOFT, m means the number of Butterfly factors and b means the block size, which is the same with OFT. For ETHER+, n means the number of blocks. For LoRETTA, r_{LoRA} means the rank of LoRA and r_{TT} means the rank of TT decomposition. For our method TLoRA, r_{TRM} means the rank of the TRM transform and r_{TR} means the rank of the TR residual adaptation.

Method	LoRA	DoRA	OFT	BOFT	ETHER+	LoRETTA	TLoRA
Setting	r=1	r=1	b=2	(m=2, b=2)	n=1	(r_{LoRA}=8, r_{TT}=6)	(r_{TRM}=1, r_{TR}=2)
Learning rate	5e−5	5e−5	1e−4	1e−4	5e−4	5e−4	5e−4

Table 4. Hyper-parameters of the controllable generation experiment. For OFT, b means the block size. For BOFT, m means the number of Butterfly factors and b means the block size, which is the same with OFT. For ETHER+, n means the number of blocks. For LoRETTA, r_{LoRA} means the rank of LoRA and r_{TT} means the rank of TT decomposition. For our method TLoRA, r_{TRM} means the rank of the TRM transform, r_{LoRA} means the rank of the LoRA residual adaptation and r_{TR} means the rank of the TR residual adaptation. λ is the scale of regularization.

Method	LoRA	DoRA	OFT	BOFT	ETHER+	LoRETTA	TLoRA*	TLoRA*	TLoRA	TLoRA
Setting	r=1	r=1	b=2	(m=2, b=2)	n=1	(r_{LoRA}=8, r_{TT}=6)	(r_{TRM}=2, r_{LoRA}=2)	(r_{TRM}=2, r_{LoRA}=4)	(r_{TRM}=2, r_{TR}=6)	(r_{TRM}=1, r_{TR}=8)
Learning rate	5e−4	5e−4	1e−4	1e−4	1e−3	1e−5	5e−4	5e−4	1e−3	1e−3
λ	-	-	-	-	-	-	0	0	1e−3	1e−3

For the controllable generation, we follow the implementation of ControlNet [68]. It contains a shallow 8-layer CNN to encode the control signals. For a fair comparison and being consistent with previous works [4, 32, 47], we report the number of training parameters for adaptation parts of all methods. The optimizer is AdamW. For the CelebA-HQ dataset in the L2I task, the batch size is 16 and we fine-tune the model for 22 epochs. For the ADE20K dataset in the S2I and C2I tasks, the batch size is 8 and we fine-tune the model for 20 epochs. For all methods, learning rates are tuned from $\{1\text{e}{-3}, 5\text{e}{-4}, 1\text{e}{-4}, 5\text{e}{-5}\}$. We do some preliminary learning rate search on the L2I and S2I tasks, and find the optimal learning rate of each method is similar for these two tasks. Due to the computational cost, we use the same learning rate for each method on three tasks. Moreover, as the training of LoRETTA is unstable, we additionally adopt a smaller learning rate $1\text{e}{-5}$ for it.

Hyper-parameters. The hyper-parameters for the subject-driven generation is provided in Tab. 3.

The hyper-parameters for the controllable generation is provided in Tab. 4. For this experiment, we find adding the identity regularization $\mathcal{R}_I$ sometimes helps the performance. We apply a scale λ before adding the regularization on the original loss. The scale λ is chosen from $\{0, 1\text{e}{-3}\}$. Moreover, for tensorization of large matrices, we choose the following setting, where the keys are original dimensions and values are dimensions of sub-indices. We do not test other tensorization shapes.

```
TENSOR_SHAPE_DICT = {
    '320': [4, 8, 10], '640': [8, 8, 10],
    '768': [8, 8, 12], '1280': [8, 10, 16],
    '2048': [8, 16, 16], '2560': [10, 16, 16],
    '5120': [20, 16, 16], '10240': [32, 20, 16],
}
```

Evaluation. The evaluation of the subject-driven generation is described in Sec. 4.1.

For the L2I task, we use the same face landmark detector provided in the `face-alignment` library [6] to detect landmarks from generated images. The Mean Squared Error (MSE) between ground truth and detected landmarks is reported. For the S2I task, we adopt the SegFormer B4 model [60] pre-trained on the ADE20K dataset for segmentation of generated images. The model is downloaded from HuggingFace[4]. Then, the mean Intersection over Union (mIoU), all ACC (aACC) and mean ACC (mACC) metrics are reported. For C2I, we compute the canny edges of generate images using the same detector as in [68]. Then, we evaluate the IoU and F1 score of the Canny edges of generated images. For each test sample, we generate six images to report the mean and standard deviation.

C.2. Computational cost

Our method is computationally efficient, since the tensor decomposition structure consists of several small linear layers Eqs. (4) and (5). The main computational overhead comes from multiplying pretrained weights with transform matrices, which is inherent to related methods such as ETHER, OFT and BOFT. Training times measured on an NVIDIA A100 GPU for controllable generation

[4] https://huggingface.co/nvidia/segformer-b4-finetuned-ade-512-512

Table 5. Computing time.

	LoRA	DoRA	ETHER+	BOFT	OFT	TLoRA*(2,4)	TLoRA(2,8)
Iter./Sec.↑	0.58	0.55	0.30	0.35	0.34	0.37	0.34

tasks demonstrate competitive computational efficiency, as shown in Tab. 5.

C.3. Failure of LoRETTA

When applying the LoRETTA [65] on controllable generation tasks, we find the training is unstable. It either generates unrealistic images or diverges. We test learning rates from $\{1e{-}3, 5e{-}4, 1e{-}4, 5e{-}5, 1e{-}5\}$. For large learning rates, it diverges quickly, while for small learning rates, it does not learn the control signals well and the image quality is low. To illustrate this, we showcase the training process of LoRETTA on the C2I task in Fig. 8. This may be because of its non-zero initialization, which destroys the information from the pre-trained model. As a comparison, our method, which also adopts TR residual adaptation, works well for these datasets.

C.4. Additional visualization results

We present more visualization results in Figs. 9 to 11.

D. Related work

Due to the space limit, we present a more comprehensive review of related work here.

Text-to-image model personalization. Text-to-image generative models have shown exceptional results in image synthesis [10, 46, 49, 50, 53]. Given the various pre-trained models available, many works aim to fine-tune these models for personalized datasets or tasks, such as subject-driven generation and controllable generation. Gal et al. [12] propose learning given subjects via textual inversion, while Ruiz et al. [52] fine-tune the whole model. ControlNet [68] incorporates an additional network branch, which can learn datasets of paired control signals and images. While Ruiz et al. [52], Zhang et al. [68] have large numbers of trainable parameters, Kumari et al. [26] show that fine-tuning the attention layers alone is also effective for these tasks. More recently, many works have focused on developing PEFT methods for these attention layers and have shown promising results [3, 5, 7, 17, 18, 21, 47, 61, 66, 71, 75]. In particular, Zhuang et al. [75] propose time-varying adapters that match the denoising process of diffusion models; this can be an interesting direction to further improve our method. There are also training-free approaches [51], which could be slow at inference.

Parameter-efficient fine-tuning. PEFT has become a hot topic with the emergence of large foundation models including text-to-image models and large language models. Popular PEFT methods include Adapter [19], Prefix-tuning [30], Prompt-tuning [28], low-rank adaptation [LoRA, 20], and many of their variants. Adapter adds additional layers after pretrained feed forward layers. Prompt-tuning introduces learnable prompts for specific tasks. LoRA has become the most popular PEFT method due to its simplicity and impressive performance [11]. Many variants of LoRA have been proposed [22, 23, 25, 31–33, 39, 43, 47, 69]. In particular, DoRA [31] proposes decomposing the pre-trained weight into magnitude and direction, where vanilla LoRA is applied to the direction. In this work, we show that DoRA can also be connected to our work by using a diagonal transform. OFT [47] applies a learnable orthogonal transform for adaptation. However, for parameter efficiency, OFT adopts block diagonal matrices, which are highly sparse. Subsequently, many methods aim to improve OFT by applying more efficient dense transform structures, including Butterfly matrix [32], Given rotation [34], Householder reflection [4, 67] and Kronecker product [71]. Our method adopts a similar idea of using a transform, but we design a different efficient dense matrix parameterization using tensor decomposition. Some methods also share similarities with ours by using *pre-defined and fixed* transforms on the pre-trained weights to project onto some low-rank spaces [5, 13, 54]. There are also works aim to design memory-efficient *full-parameter* fine-tuning/pre-training optimizers [33, 43, 72]. However, they do not provide light-weight adapters that can be plugged into foundation models.

Tensor decomposition. TD is a classical tool in signal processing and machine learning [9]. In particular, tensor-train (TT) decomposition [42] and its extension, tensor-ring (TR) decomposition [73], have shown exceptional results in model compression, including MLP [40], CNN [14, 58], RNN/LSTM [37, 44, 55, 64] and Transformer [35, 45]. Recently, TDs have also been applied to fine-tuning tasks. Jie and Deng [24] parameterize the Adapter layers using a TT format and show ultra-parameter-efficiency in ViT adaptation. Yang et al. [65] extend this idea to large language model PEFT, and apply the TT format to both Adapters and LoRA factors. Similarly, Anjum et al. [2] propose to directly parameterize the adaptation using the TT format. Chen et al. [8] adopt a quantum-inspired tensor network, which is a generalization of the TT-Matrix (TTM) form. While these works use similar TT/TR structures to our model, they do not apply the transform adaptation. Moreover, we study a different initialization strategy for the TR factors, which would be more stable, as we show in our experiments.

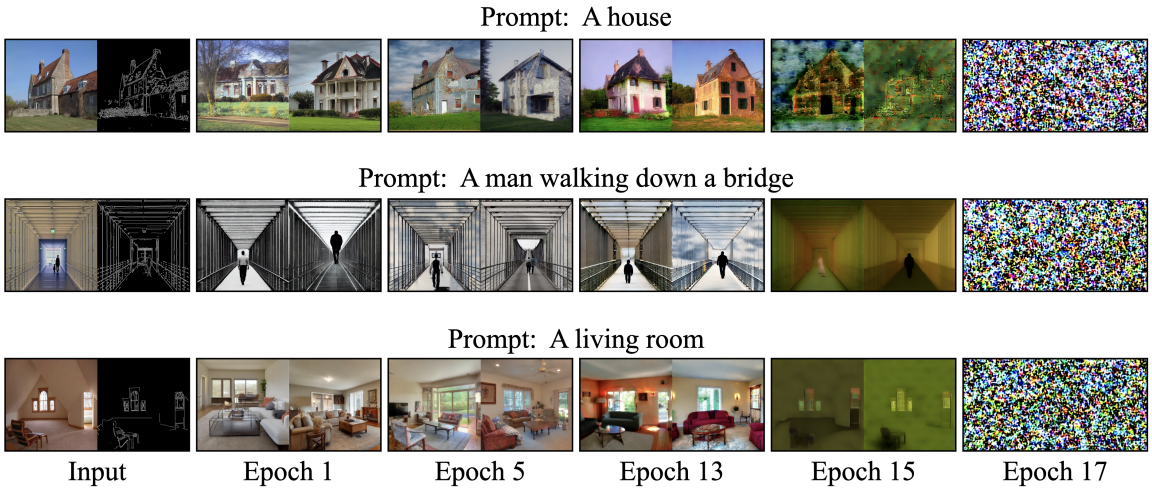

Figure 8. Training process of LoRETTA on the C2I task.

Figure 9. Qualitative comparison of the subject-driven generation results.

Figure 10. Qualitative comparison of the subject-driven generation results.

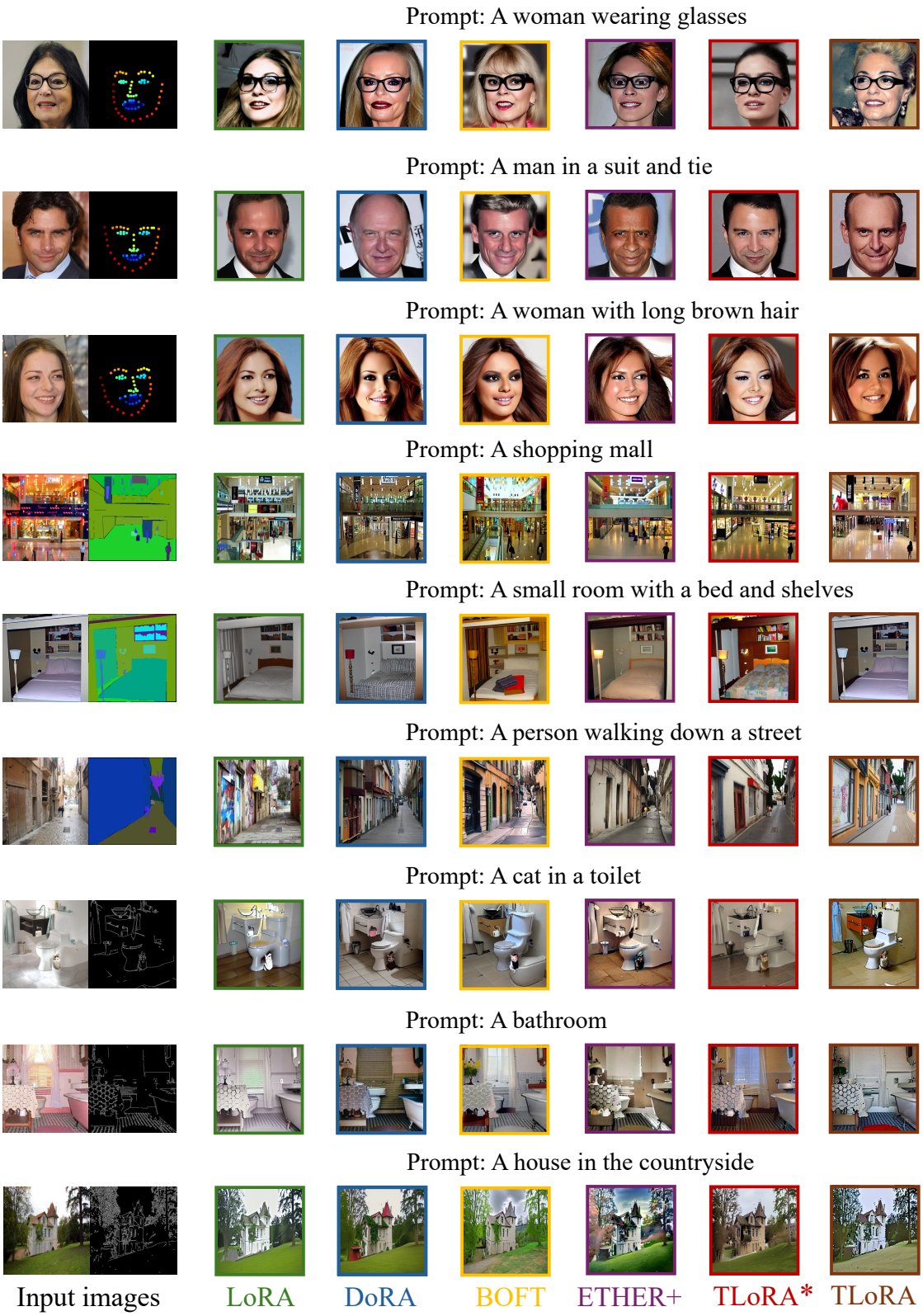

Figure 11. Qualitative comparison of the subject-driven generation results.