Sparse Fine-Tuning of Transformers for Generative Tasks

Supplementary Material

A. Analysis

A.1. Analysis of Adapted Feature Representation

With our formulation, the adapted feature representation $\Delta\mathbf{O}$ at each layer can be expressed as a linear combination of learned atoms $\mathbf{D}$. However, deep neural networks represent features non-linearly. Although $\Delta\mathbf{O}$ can be expressed linearly in terms of $\mathbf{D}$, its influence becomes non-linear as it interacts with activation functions and transformations in subsequent layers. In this section, we introduce an approximation method to address this non-linearity, enabling us to trace and quantify the contribution of each atom to the final outputs throughout the entire model.

Formulation of the pre-trained model. We represent each layer of the pre-trained model as $f^{(l)}(\mathbf{x})$. For an L-layer model, it writes as $f^{(L)} \circ \cdots \circ f^{(1)}(\mathbf{x})$, where $l = 1, \cdots, L$ is the index of the layer. Based on the functional approximation theory, we can approximate the function in an appropriate basis. Given a basis $\{g_k\}_{k=1}^{K}$, the coefficients of basis expansion are $\mu_k = \langle f, g_k \rangle$, such that $f(\mathbf{x}) = \sum_{k=1}^{K} \mu_k g_k(\mathbf{x})$, where $\langle \cdot, \cdot \rangle$ is inner product. For simplified analysis, we utilize the polynomial basis, and represent $f(\mathbf{x})$ as:

$$f(\mathbf{x}) = \sum_{k=1}^{K} c_k \mathbf{x}^k. \tag{10}$$

The influence of atoms on one layer. Fine-tuning introduces adapted feature representation at layer $l-1$, *i.e.*, $\Delta\mathbf{O}^{(l-1)} = \mathbf{SD}$, which become the perturbation of the input $\mathbf{X}^{(l)}$ at layer l. For each input, we have $\mathbf{x} + \sum_{m=1}^{M} s_m \mathbf{d}_m$, where $\mathbf{d}_m \in \mathbb{R}^{C_o}$ is the single atom in $\mathbf{D}$ and s_m is the corresponding sparse coefficient. With the perturbation, we have

$$f(\mathbf{x} + \sum_{m=1}^{M} s_m \mathbf{d}_m) = f(\mathbf{x}) + \sum_{m=1}^{M} \mathbf{B}_K(\mathbf{d}_m), \tag{11}$$

where $\mathbf{B}_K(\mathbf{d}_m) = \sum_{k=1}^{K} \beta_{k,m} \mathbf{d}_m^k$ and $\beta_{k,m}$ depends on $\mathbf{x}$, s_m, and c_k. We assume $\langle \mathbf{d}_i, \mathbf{d}_j \rangle = 0, \forall i \neq j$ to avoid correlated influence from different atoms. It can be achieved with a simple regularization term. With the influence of linear perturbations based on atoms at layer $l-1$, the different in the output at layer l is determined by the linear combination of higher-order terms of these atoms $\mathbf{d}_m^k$.

The accumulated influence of atoms on multiple layers. Applying the same formulation to layer $l+1$ results in a similar expression as above, but incorporates the combined influence of atoms from both layer $l-1$ and layer l. Specifically, the input of layer $l+1$ is written as,

$$\mathbf{x}^{(l+1)} + \sum_{m=1}^{M} s_m^{(l)} \mathbf{d}_m^{(l)} + \sum_{m=1}^{M} \mathbf{B}_{K^{(l-1)}}(\mathbf{d}_m^{(l-1)}), \tag{12}$$

where $\mathbf{x}^{(l+1)}$ is the original input at layer $l+1$, $\sum_{m=1}^{M} s_m^{(l)} \mathbf{d}_m^{(l)}$ is the linear perturbations based on the atoms from layer l, $\sum_{m=1}^{M} \mathbf{B}_{K^{(l-1)}}(\mathbf{d}_m^{(l-1)})$ is the influence of the atoms from layer $l-1$, which is a combination of higher-order terms. The output of layer $l+1$ is,

$$\begin{aligned} &f^{(l+1)}(\mathbf{x}^{(l+1)} + \sum_{m=1}^{M} s_m^{(l)} \mathbf{d}_m^{(l)} + \sum_{m=1}^{M} \mathbf{B}_{K^{(l-1)}}(\mathbf{d}_m^{(l-1)})) \\ =&f(\mathbf{x}) + \sum_{m=1}^{M} \mathbf{B}_{K^{(l)}}(\mathbf{d}_m^{(l)}) + \sum_{m=1}^{M} \mathbf{B}_{K^{(l-1)}+K^{(l)}}(\mathbf{d}_m^{(l-1)}). \end{aligned} \tag{13}$$

This formulation can be naturally extended to the whole model. We provide detailed analysis in Appendix A.

After incorporating atoms into each layer to linearly capture the adapted feature representations, the model output can be interpreted as the original pre-trained output with perturbations introduced by the atoms at each layer. The atoms from the shallow layer (e.g., layer 1), introduce more higher-order terms to the final output, i.e., $\mathbf{B}_{\sum_{l=1}^{L} K^{(l)}}(\mathbf{d}_m^{(1)})$. It contributes more to the overall structure of the output. The atoms from the deep layer ((e.g., layer L)), introduce fewer higher-order terms, or only linear terms to the final output. It contributes more to the details of the output.

A.2. Proof

The proof of (11). To prove (11), we assume $\langle \mathbf{d}_i, \mathbf{d}_j \rangle = 0, \forall i \neq j$ to avoid correlated influence from different atoms. It can be achieved with a simple regularization term.

Proof. Inserting (10) to the LHS of (11), we have

$$f(\mathbf{x} + \sum_{m=1}^{M} s_m \mathbf{d}_m) = \sum_{k=1}^{K} c_k (\mathbf{x} + \sum_{m=1}^{M} s_m \mathbf{d}_m)^k. \tag{14}$$

After expanding this equation, we have

$$\begin{aligned}&\sum_{k=1}^{K} c_k(\mathbf{x}+\sum_{m=1}^{M} s_m\mathbf{d}_m)^k \\ =&\sum_{k=1}^{K} c_k\left[\sum_{j=0}^{k}\binom{k}{j}\mathbf{x}^{k-j}\left(\sum_{m=1}^{M} s_m\mathbf{d}_m\right)^j\right].\end{aligned} \tag{15}$$

Considering $\langle \mathbf{d}_i, \mathbf{d}_j\rangle = 0, \forall i \neq j$, we have,

$$\left(\sum_{m=1}^{M} s_m\mathbf{d}_m\right)^j = \sum_{m=1}^{M}(s_m\mathbf{d}_m)^j. \tag{16}$$

Thus,

$$\begin{aligned}&f(\mathbf{x}+\sum_{m=1}^{M} s_m\mathbf{d}_m) \\ =&\sum_{k=1}^{K} c_k\left[\mathbf{x}^k+\sum_{j=1}^{k}\binom{k}{j}\mathbf{x}^{k-j}\sum_{m=1}^{M}(s_m\mathbf{d}_m)^j\right] \\ =&f(\mathbf{x})+\sum_{m=1}^{M}\sum_{k=1}^{K} c_k\sum_{j=1}^{k}\binom{k}{j}\mathbf{x}^{k-j}(s_m\mathbf{d}_m)^j.\end{aligned} \tag{17}$$

We define

$$\mathbf{B}_K(\mathbf{d}_m) = \sum_{k=1}^{K} c_k\sum_{j=1}^{k}\binom{k}{j}\mathbf{x}^{k-j}(s_m\mathbf{d}_m)^j, \tag{18}$$

which represents the higher-order terms of $\mathbf{d}_m$ up to order K. □

The proof of (13). Here we only provide a sketch of proof of (13). Compared with (11), (13) contains an additional term $\sum_{m=1}^{M}\mathbf{B}_{K^{(l-1)}}(\mathbf{d}_m^{(l-1)})$. After applying the expansion (10), we will have

$$\sum_{j=0}^{k}\binom{k}{j}(\mathbf{B}_{K^{(l-1)}}(\mathbf{d}_m^{(l-1)}))^{k-j}\left(\sum_{m=1}^{M} s_m\mathbf{d}_m\right)^j, \tag{19}$$

which produces the higher-order terms of $\mathbf{d}_m^{(l-1)}$ up to order $K^{(l-1)}+K^{(l)}$, thus we have $\mathbf{B}_{K^{(l-1)}+K^{(l)}}(\mathbf{d}_m^{(l-1)})$ in (13).

B. Experimental Settings

MNIST generation with VAE. In this experiment, the input of MNIST has the size of 28×28. The VAE consists with an encoder and decoder, each with two layers, a feature dimension of 128, and four attention heads. It first reshape the input with a patch size of 7. The latent space is mapped to a dimension of 32. We train the model using the Adam optimizer with a learning rate of 0.001 for 20 epochs.

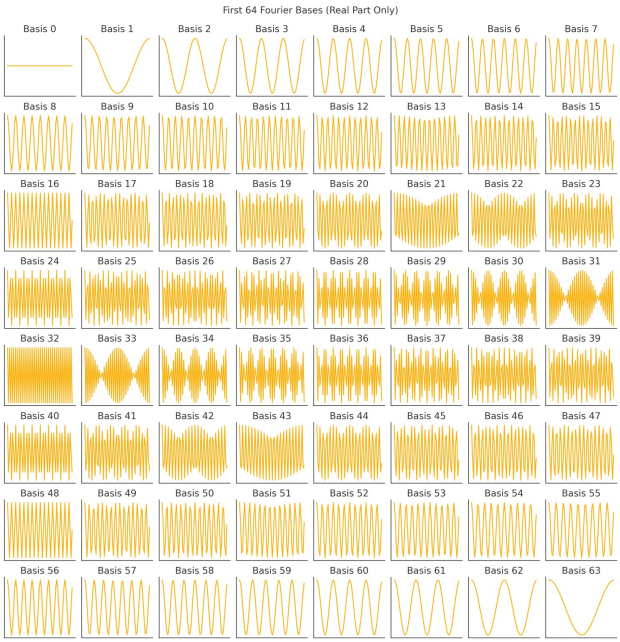

Figure 8. The illustration of Fourier basis.

Generative tasks with DiT. In this experiment, we use the CAME [29] optimizer with a learning rate of 1×10^{-4} to fine-tune the Pixart-Σ [7]. For the baseline methods, LoRA and DoRA are assigned a rank of $r = 16$, while OFT is set to $r = 4$. We generated five images for each prompt at a resolution of 1024×1024. We run the experiment on Nvidia A5000 with 24GB memory. For the image editing task, we train the model on 1 image for 60 epochs, which takes about 5min. For the concept customization task, we train the model on 4-6 images for 15 epochs, which takes about 5min.

C. Additional Experimental Results

C.1. Toy Experiment

Experimental setting. In this experiment, we only use the attention block to transform the signal, the feature dimension is 128, and the sequence length is 64. The synthetic signals are generated by randomly combining 5 Fourier bases, which is shown in Figure 8. To leverage the transformer, we first project the 1D signal into a 64D space using a randomly initialized projection matrix, which remains frozen during training. The output signal is then mapped from 64D back to 1D by summing across all 64 dimensions.

Sparse coefficients provide interpretability. The sparse coefficients $\mathbf{S}$ enhance interpretability by establishing a direct connection between a small subset of atoms and the output feature, *i.e.*, $\mathbf{O}_i = \sum_{j=1}^{M}\mathbf{S}_{ij}\mathbf{D}_j$. This behavior

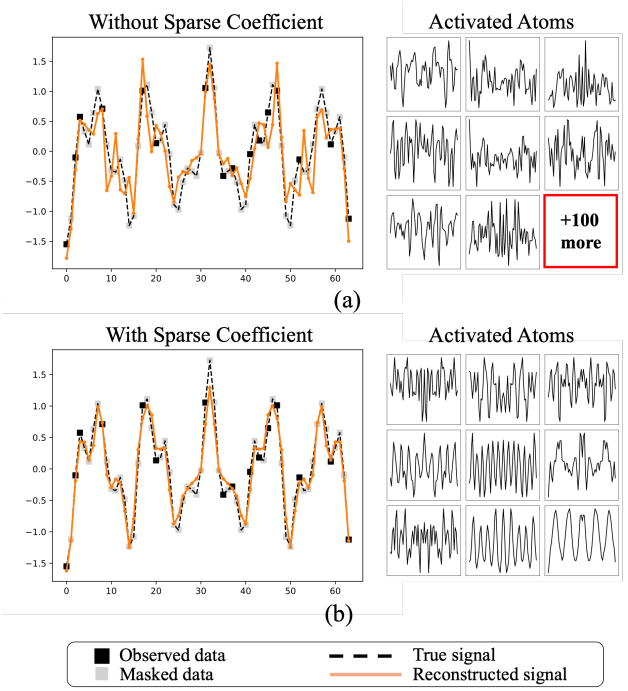

Figure 9. Compared to the performance of (a) standard attention $\mathbf{AXW}_v$, (b) the sparse combination of atoms $\sigma_\lambda(\mathbf{AXW}_s)\mathbf{D}$ provides more interpretability. The reconstructed signal is a linear combination of basic signals. Without sparse coefficients, constructing the signal requires 100 atoms. However, with sparse coefficients, the reconstructed signal is effectively represented using only 9 atoms, demonstrating a more compact and interpretable composition.

is also validated in the literature on sparse coding, compressed sensing, and dictionary learning [6]. To explore this within the context of an attention block, we perform an experiment where masked one-dimensional signals are reconstructed using an attention mechanism. The signals, with a length of 64, are synthesized by selecting 5 random Fourier bases, with a low frequency ranging from $\frac{0}{64}$ to $\frac{24}{64}$. The objective is to reconstruct the signal after masking out 75% of its values with only 16 randomly sampled observations. As illustrated in Figure 9, for both $\mathbf{O} = \mathbf{AXW}_v$ and $\mathbf{O} = \sigma_\lambda(\mathbf{AXW}_s)\mathbf{D}$, the attention block successfully reconstructs signals after jointly learning the coefficients and atoms. $\sigma_\lambda(\mathbf{AXW}_s)\mathbf{D}$ takes advantage of sparse coding, and requires only 9 atoms to fully reconstruct the signal. In comparison, $\mathbf{AXW}_v$ requires 100 atoms to construct the signal. Interestingly, the atoms $\mathbf{D}$ in $\sigma_\lambda(\mathbf{AXW}_s)\mathbf{D}$ naturally resemble certain Fourier basis functions from the ones used to synthesize real data.

Description of reconstructed images. Figure 3 also shows that the number of selected atoms impacts the generated results. For example, when learning the concept described in the top row, "*A grey* $\langle V \rangle$ *wolf plushie ...*", by adjusting the number of active atoms we can see that some atoms influence the texture of the fur, while others shape the posture of the object or determine the position of the bow tie. For the concept in the second row, "*A blue* $\langle V \rangle$ *sports car ...*", we observe that certain atoms influence the orientation of the sports car, while others affect the position of the rear wing and the shape of the grille. For the concept in the third row, "*A* $\langle V \rangle$ *wooden barn ...*", we observe that certain atoms influence the number of lean-tos on the barn, while others affect the slope of the roof.

C.2. Personalization Results Comparison

In this section, we showcase the comparison of personalization results for various concepts selected from the DreamBooth [47] dataset.

As shown in Figures 11–16, our approach produces outputs that not only align more accurately with the text prompts but also preserve the fine details of each learned concept more effectively than the baselines.

C.3. Sparse Fine-tuning Modules Merging

We conduct concept-style merging experiments following [14, 48]. We begin by fine-tuning the pre-trained diffusion model separately on the concept and style images using our method, producing their respective fine-tuned weights. For a given concept–style pair, we then merge the weights by directly summing the corresponding fine-tuned parameters. With the merged weights applied to the pre-trained model, it can generate images that reflect both the desired concept and style. We observe that summing the weights does not lead to forgetting of either attribute, since the activated atoms from the fine-tuned weights rarely overlap due to the highly sparse nature of the learned dictionaries. Fig. 10 demonstrates that our method preserves composability, indicating that our method enables multi-task behavior through simple weight merging.

Figure 10. Examples of concept-style merging.

Figure 11. Personalization generated results comparison

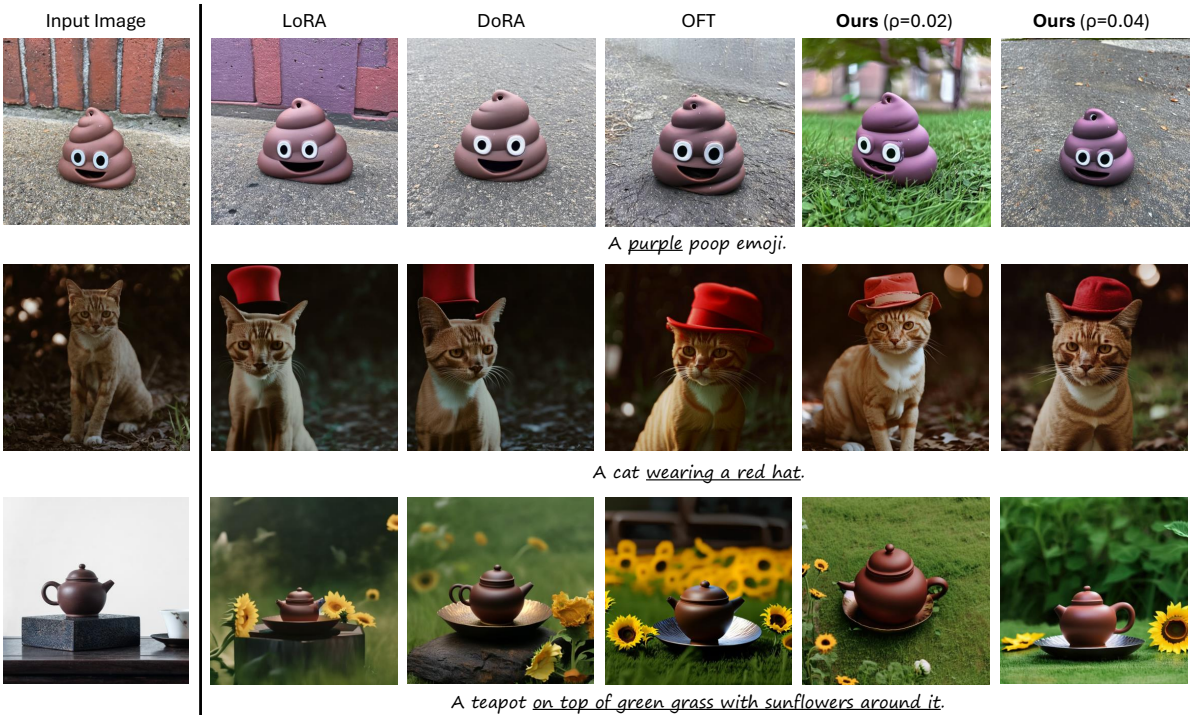

Figure 12. Personalization generated results comparison

Figure 13. Personalization generated results comparison

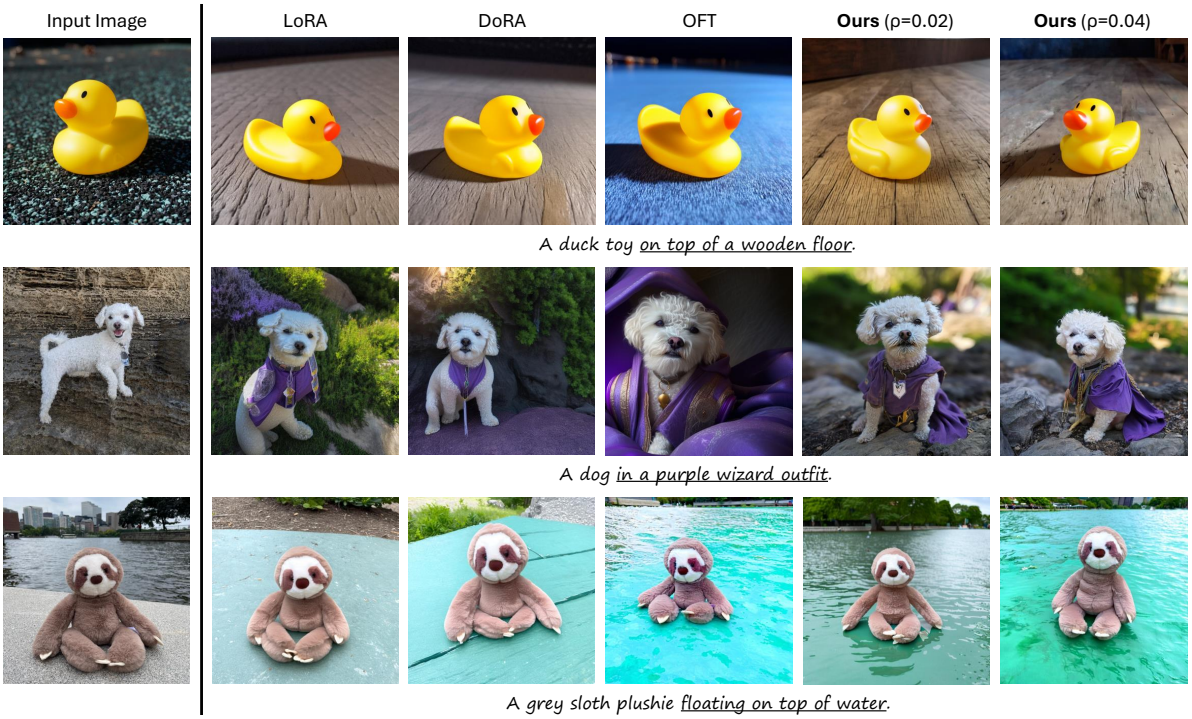

Figure 14. Personalization generated results comparison

Figure 15. Personalization generated results comparison

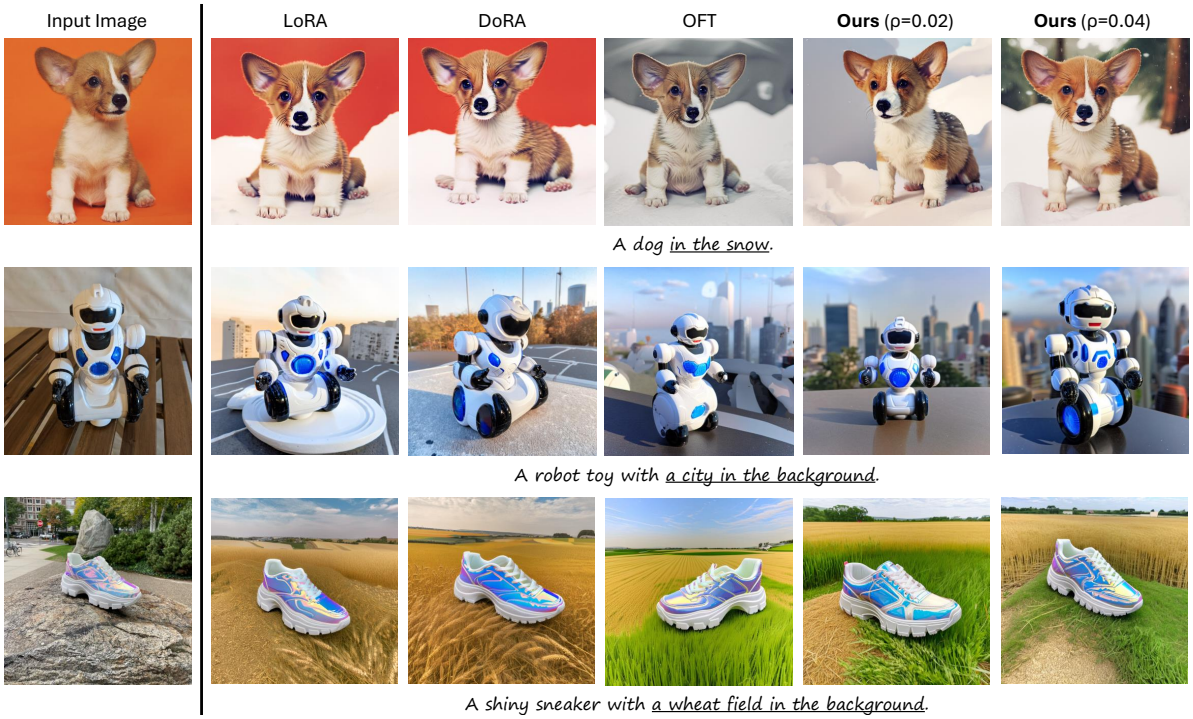

Figure 16. Personalization generated results comparison