S^{2}FT: Parameter-Efficient Fine-Tuning in Sparse Spectrum Domain

Supplementary Material

Algorithm 1 PyTorch-style pseudocode for **S^{2}FT**.

```
class S2FT(nn.Module):
    def __init__(self, n, in_dim, out_dim, alpha,
    est_w, base_layer):
        super(SFourierFT, self).__init__()
        self.n = n
        self.in_dim = in_dim
        self.out_dim = out_dim
        self.alpha = alpha
        self.base_layer = base_layer
        # weight change pre-estimation by Eq.(4)
        self.est_w = est_w
        # seeking invertible transformation by Eq.(6)
        self.w, self.row_order, self.col_order =
        \self.transform(self.est_w)
        # magnitude spectrum calculation
        freq = torch.fft.fft2(self.w)
        magnitude = torch.abs(freq)
        # spectral coefficient sampling in section 3.2.3
        self.index = self.sample(magnitude, self.n)
        # spectral coefficient initialization
        self.c = nn.Parameter(torch.randn(n))

    def transform(self, matrix):
        row_dim = matrix.shape[0]
        col_dim = matrix.shape[1]
        current_index = 0
        # rows rearrangement in section 3.2.2
        for _ in range(row_dim - 1):
            last_row = matrix[current_index]
            # nearest neighbor search in Eq6
            dist = compute_dist(matrix, last_row)
            current_index = torch.argmin(dist)
            row_order.append(current_index)
            reordered = matrix[row_order]
        # cols rearrangement in section 3.2.2
        for _ in range(col_dim - 1):
            last_col = reordered[:, current_index]
            dist = compute_dist(reordered, last_col)
            current_index = torch.argmin(dist)
            col_order.append(current_index)
            reordered = reordered[:,row_order]
        return reordered, row_order, col_order

    def forward(self, x):
        F = torch.zeros(self.in_dim, self.out_dim)
        F[self.index[0, :], self.index[1, :]] = self.c
        W = torch.fft.ifft2(F).real * self.alpha
        # inverse re-indexing in section 3.2.2
        Delta_w = W[self.row_order,:][:,self.col_order]
        # merge
        h = self.base_layer(x)
        h += torch.einsum('ijk,kl->ijl', x, Delta_W)
        return h
```

1. Pseudocode for S^{2}FT

To facilitate reproducibility and clarify the implementation details, we provide PyTorch-style pseudocode for S^{2}FT in Algorithm 1. The algorithm begins with a pre-estimated weight change matrix (as defined in Eq.(4)), from which an invertible transformation is derived via row and column rearrangement (described in figuretion 3.2.3 and formalized in Eq.(5)). The transformed matrix is then converted to the frequency domain using a 2D Fourier transform, and a subset of spectral coefficients is selected based on magnitude (see figuretion 3.2.3). These coefficients are treated as learnable parameters during training. In the forward pass, the modified weights are reconstructed via inverse Fourier transform, reindexed to match the original structure, and integrated into the base model for fine-tuning.

2. Proof of Proposition 1.

For notational simplicity, throughout this proof we let $X = \Delta\bar{W} \in \mathbb{R}^{n\times c}$ denote the rearranged matrix in Eq. (5). Thus, X_k represents the k-th row of $\Delta\bar{W}$. We provide the proof for the row-wise case; the column-wise case is identical.

The objective in Eq. (5) can be rewritten as

$$\begin{aligned}\mathcal{J} = \sum_{k=1}^{n-1} \beta_{\pi_k,\pi_{k+1}} &= \sum_{k=1}^{n-1} |\Delta\bar{W}_{k+1} - \Delta\bar{W}_k|^2 \\ &= \sum_{k=1}^{n-1} |X_{k+1} - X_k|^2,\end{aligned} \tag{1}$$

Define the forward difference operator acting on the sequence $\{X_k\}_{k=1}^n$ as

$$(\Delta X)_k = X_{k+1} - X_k, \qquad k = 1, \ldots, n. \tag{2}$$

Under this notation, Eq. (1) becomes

$$\mathcal{J} = \sum_{k=1}^{n} |(\Delta X)_k|^2. \tag{3}$$

Let $\widehat{X} = DFT(X)$ denote the spectral coefficients of X. By using the inverse Fourier transform, we can obtain:

$$X_{(x,y)} = \sum_u \sum_v \widehat{X}_{(u,v)} e^{j2\pi(\frac{ux}{n} + \frac{uy}{c})}, \tag{4}$$

where (x, y) are the spatial coordinates, (u, v) are the frequency coordinates. Hence,

$$\begin{aligned}\Delta X_{(x,y)} &= X_{(x+1,y)} - X_{(x,y)} \\ &= \sum_u \sum_v \left(e^{j2\pi u/n} - 1\right) \widehat{X}_{(u,v)} e^{j2\pi(\frac{ux}{n} + \frac{uy}{c})}.\end{aligned} \tag{5}$$

It means that:

$$DFT(\Delta X)_u = \left(e^{j2\pi u/n} - 1\right) \widehat{X}_u. \tag{6}$$

Method	Params. (%)	Natural	Specialized	Structured	Mean Acc
S^2FT(100%)	0.08	81.3	86.3	60.5	73.6
S^2FT(80%)	0.08	81.3	86.2	60.5	73.6
S^2FT(60%)	0.08	81.0	85.7	60.9	73.5
S^2FT(40%)	0.08	81.3	86.0	60.6	73.6
S^2FT(20%)	0.08	81.1	85.9	60.8	73.6
S^2FT(10%)	0.08	81.2	85.7	60.7	73.5

Table 1. Analysis of the train data scale on VTAB.

Then, by Parseval's theorem, we obtain

$$\begin{aligned}\mathcal{J} = \sum_{k=1}^{n} |(\Delta X)_k|^2 &= \sum_{u=0}^{n-1} |DFT(\Delta X)_u|^2 \\ &= \sum_{u=0}^{n-1} \left| \left(e^{j2\pi u/n} - 1\right) \widehat{X}_u \right|^2 \\ &= \sum_{u=0}^{n-1} \left| e^{j2\pi u/n} - 1 \right|^2 \left| \widehat{X}_u \right|^2 \\ &= \sum_{u=0}^{n-1} 4\sin^2(\frac{\pi u}{n}) |\widehat{X}_u|^2.\end{aligned} \tag{7}$$

Finally, since $X = \Delta\bar{W}$ is real-valued, its DFT spectrum exhibits conjugate symmetry, and therefore it suffices to sum over the first half frequencies:

$$\mathcal{J} = \sum_{u=0}^{n/2-1} 8\sin^2(\frac{\pi u}{n}) |DFT(\Delta\bar{W})_u|^2, \tag{8}$$

which establishes the equivalence.

□

3. Additional Ablation Study

Is there a frequency bias in our S^2FT? To analyze frequency bias, following [1], we also conduct an experiment using a sampling strategy biased toward different central frequencies to select spectrum points. As shown in Figure 1, our S^2FT achieves better performance when favoring lower central frequencies, indicating that the spectrum domain in our method is sparse. Moreover, we observe an interesting but previously unexplained phenomenon noted in the original FourierFT paper [1]: the optimal central frequency for FourierFT varies randomly across tasks, and low frequencies are often not the most effective. This can be attributed to the fact that the spectrum domain in FourierFT is not sparse but rather power-uniform.

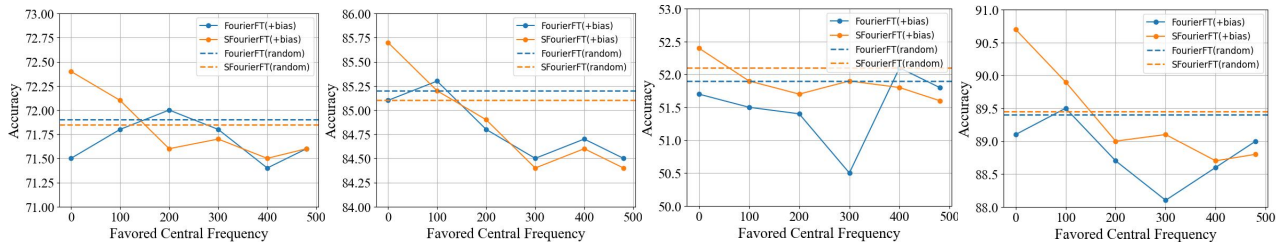
What is the computational cost and efficiency gain of weight change pre-estimation? Performing weight change pre-estimation typically requires computing gradient variations over multiple training samples, which can introduce significant computational overhead for large-scale models and datasets. To substantially improve efficiency, we adopt a lightweight estimation strategy that samples only a small subset of the training data. As shown in Table 1, using merely 20% of the training samples yields a stable approximation of the weight change while greatly reducing computation time, without compromising estimation accuracy.

Furthermore, collecting gradients through full-parameter fine-tuning in large models leads to extremely high memory consumption. To enhance memory efficiency under resource-constrained scenarios, we introduce an alternative approach: instead of computing the full weight gradient, we accumulate the gradients of the LoRA matrices A and B separately, and approximate the corresponding weight update by computing their product. This strategy significantly reduces memory usage while still providing a reliable estimation of weight change.

Model	Time (min)	GPU Memory (GB)
LLaMA-7B	14.2	11.4
LLaMA-13B	18.1	16.8

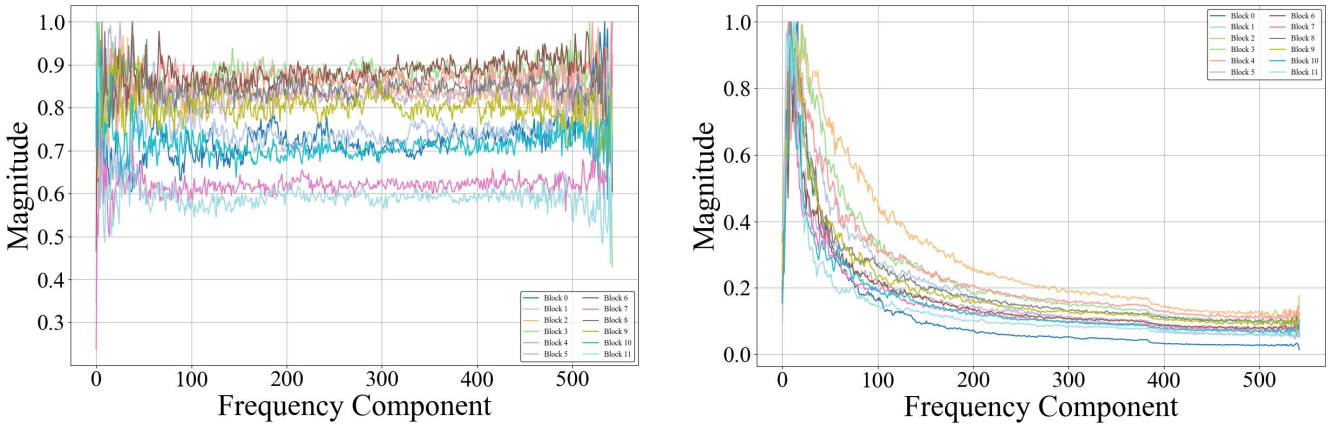
Table 2. Cost of weight change pre-estimation.

Is the observation that the power spectrum of weight change is power-uniform consistent across datasets and layers? To investigate the generality of this observation, we present additional visualization results across multiple datasets in Figures 2 $\sim$ 5, where each figure displays the power spectrum of weight changes across all layers of the model. Specifically, we include results on Caltech101 and CIFAR100, using both the query (q) and value (v) projection matrices. The left subfigures in each case show the power spectrum of the original weight change ΔW when using FourierFT. It can be clearly seen that the spectral energy is distributed in a nearly uniform pattern across all frequencies—exhibiting the so-called power-uniform characteristic. This confirms that directly treating ΔW as a spatial-domain matrix does not yield a sparse spectrum, making it difficult to model effectively by tuning only a few spectral coefficients. In contrast, the right subfigures show the spectrum of the transformed matrix $\Delta\bar{W}$ obtained via our S^2FT method. Here, the spectral energy is significantly concentrated in low-frequency regions, exhibiting a clear sparsity pattern. This demonstrates that our proposed invertible transformation successfully maps the weight change into a latent spatial-domain matrix with a sparse spectrum, which enables more effective and parameter-efficient fine-tuning. These consistent results across different datasets and projection types strongly support the robustness and generalizability of our method.



(a) DTD (b) RESISC45 (c) DMLab (d) SVHN

Figure 1. Impact of favored coefficients central frequency on S^{2}FT and FourierFT.



(a) Power spectrum of ΔW on FourierFT (b) Power spectrum of $\Delta \bar{W}$ on our S^{2}FT

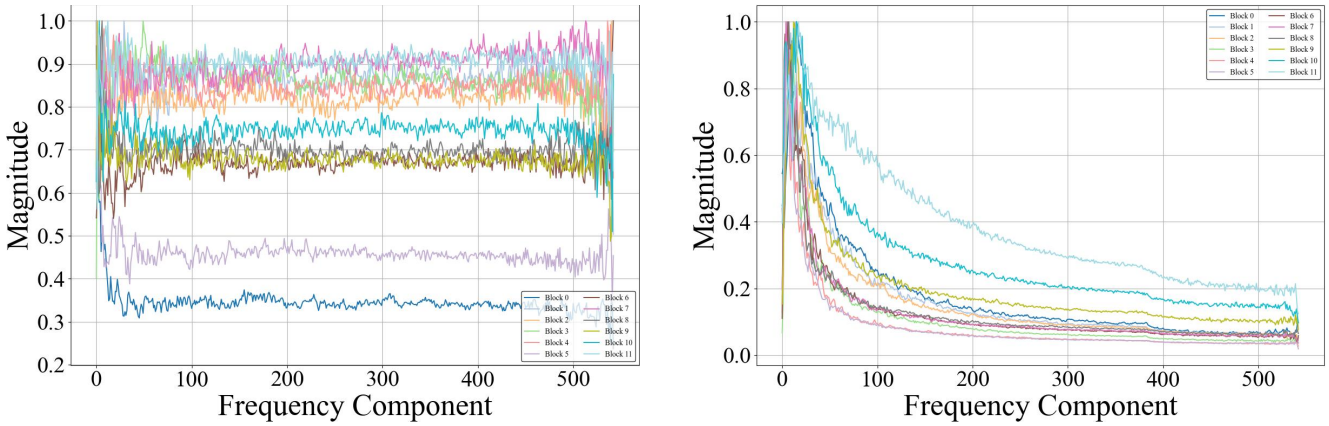
Figure 2. Distribution analysis of the query projection matrix on the Caltech101 dataset.

4. Qualitative Result on Subject-driven Generation Task

In addition to quantitative evaluations, we provide qualitative comparisons in Figure 12 to further validate the effectiveness of our proposed method. As shown, our S^{2}FT produces images with noticeably improved object consistency while preserving a harmonious and coherent visual style. These results clearly demonstrate the advantages of our approach in maintaining subject identity and ensuring high visual fidelity, particularly in subject-driven image generation tasks.

References

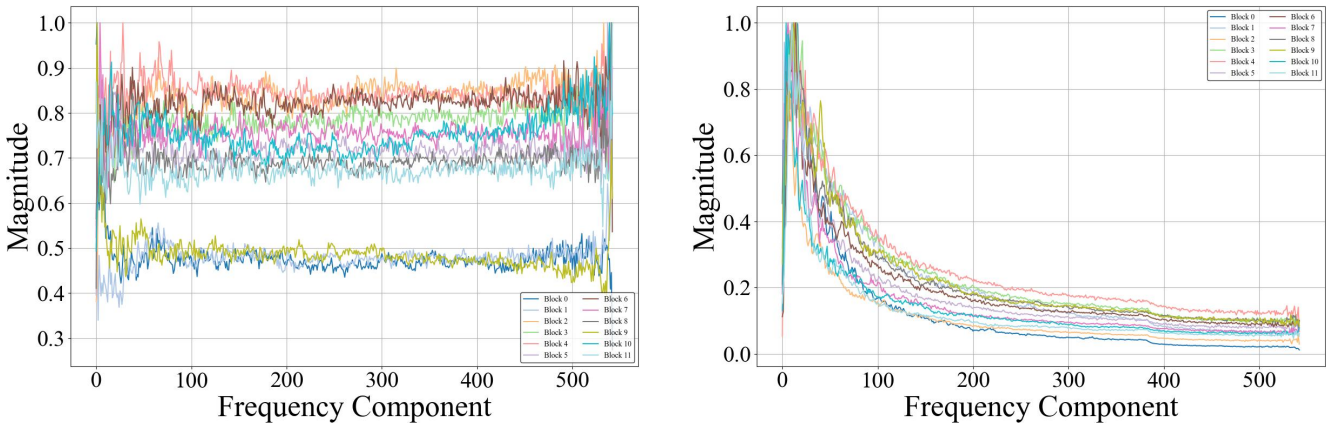
[1] Ziqi Gao, Qichao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li. Parameter-efficient fine-tuning with discrete fourier transform. *arXiv preprint arXiv:2405.03003*, 2024.



(a) Power spectrum of ΔW on FourierFT

(b) Power spectrum of $\Delta \bar{W}$ on our S^{2}FT

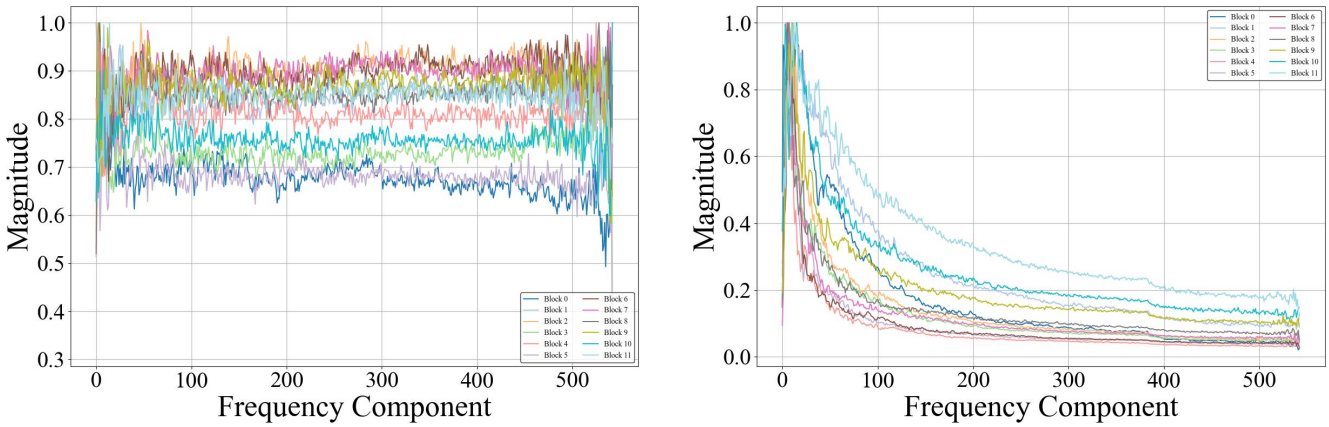
Figure 3. Distribution analysis of the value projection matrix on the Caltech101 dataset.



(a) Power spectrum of ΔW on FourierFT

(b) Power spectrum of $\Delta \bar{W}$ on our S^{2}FT

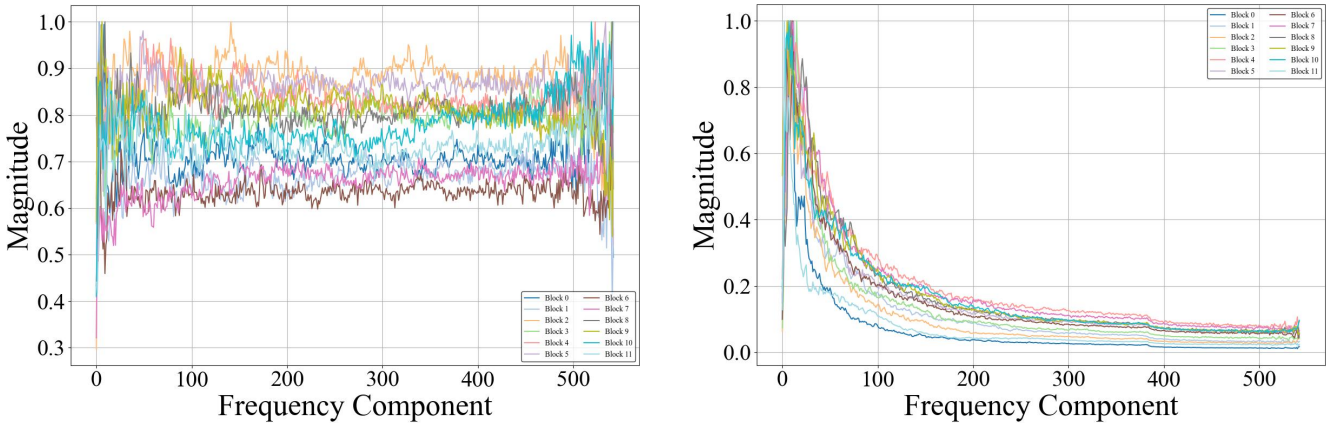
Figure 4. Distribution analysis of the query projection matrix on the CIFAR-100 dataset.



(a) Power spectrum of ΔW on FourierFT

(b) Power spectrum of $\Delta \bar{W}$ on our S^{2}FT

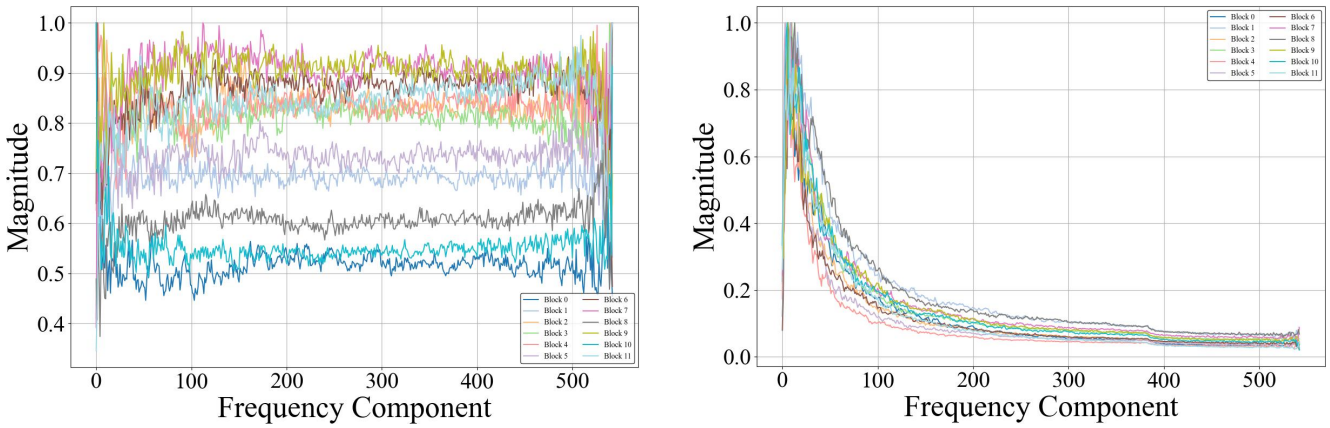
Figure 5. Distribution analysis of the value projection matrix on the CIFAR-100 dataset.



(a) Power spectrum of ΔW on FourierFT

(b) Power spectrum of $\Delta \bar{W}$ on our S^{2}FT

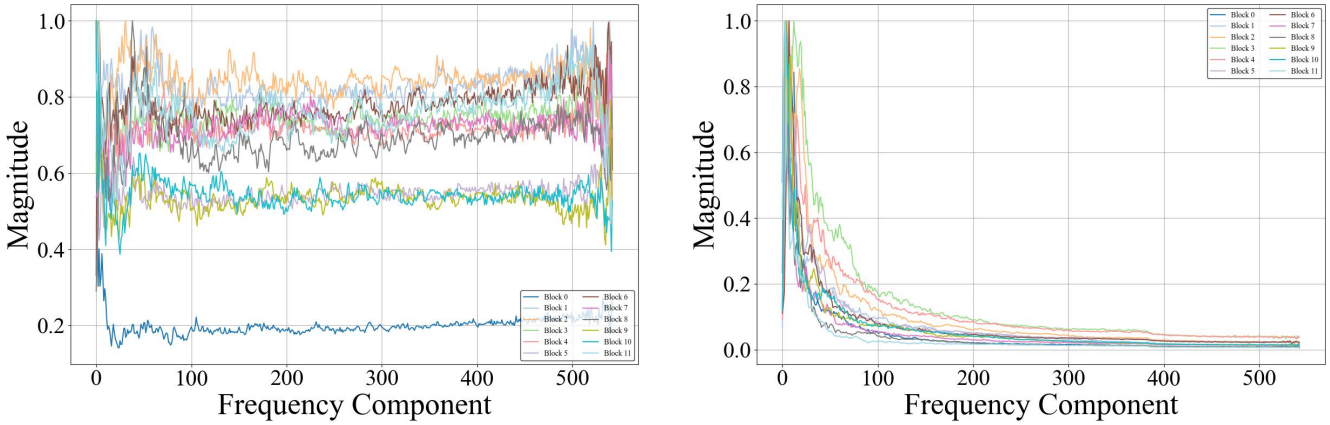
Figure 6. Distribution analysis of the query projection matrix on the EuroSAT dataset.



(a) Power spectrum of ΔW on FourierFT

(b) Power spectrum of $\Delta \bar{W}$ on our S^{2}FT

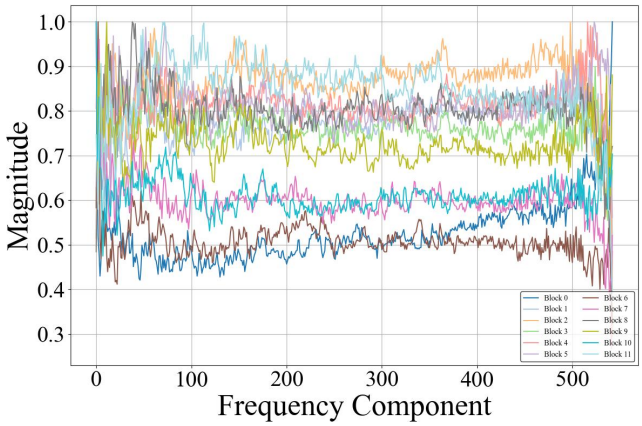
Figure 7. Distribution analysis of the value projection matrix on the EuroSAT dataset.



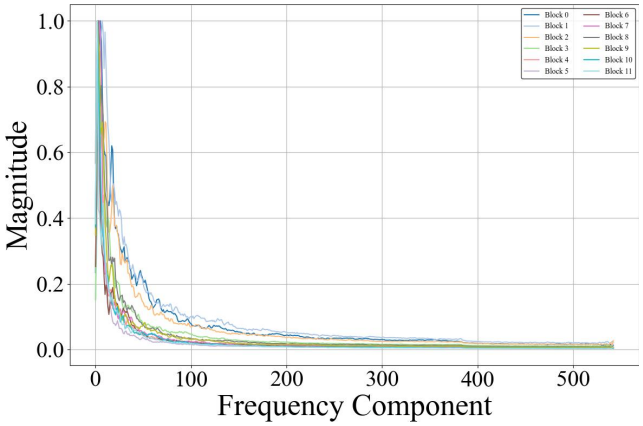
(a) Power spectrum of ΔW on FourierFT

(b) Power spectrum of $\Delta \bar{W}$ on our S^{2}FT

Figure 8. Distribution analysis of the query projection matrix on the KITTI dataset.

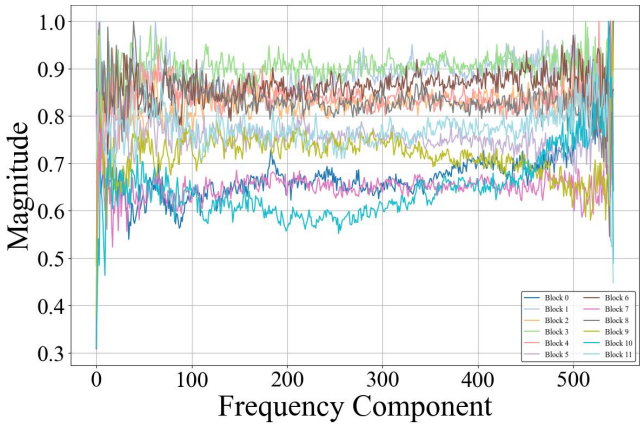


(a) Power spectrum of ΔW on FourierFT

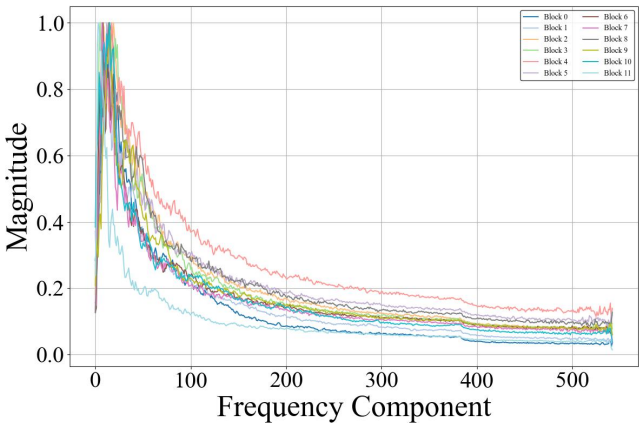


(b) Power spectrum of $\Delta \bar{W}$ on our S^{2}FT

Figure 9. Distribution analysis of the value projection matrix on the KITTI dataset.

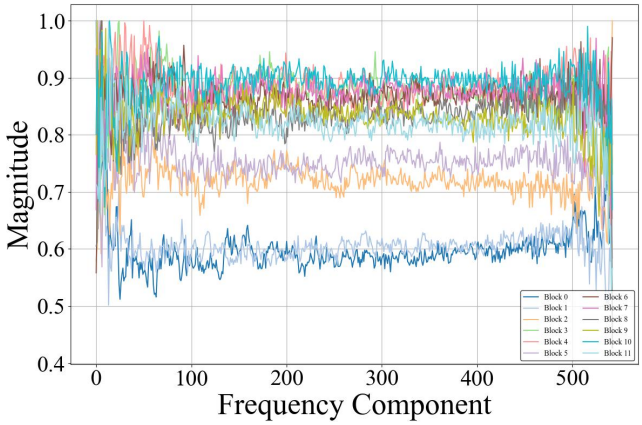


(a) Power spectrum of ΔW on FourierFT

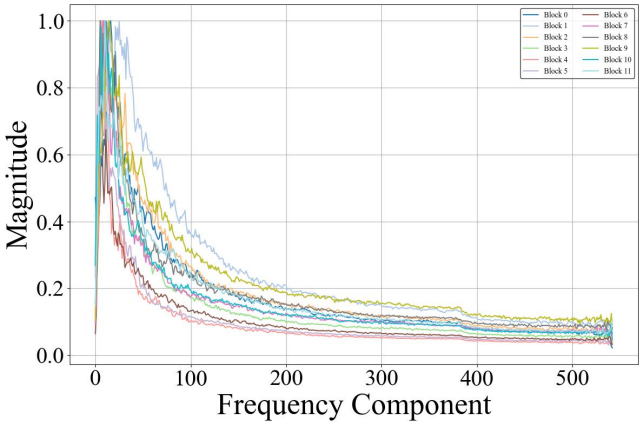


(b) Power spectrum of $\Delta \bar{W}$ on our S^{2}FT

Figure 10. Distribution analysis of the query projection matrix on the Resisc45 dataset.



(a) Power spectrum of ΔW on FourierFT



(b) Power spectrum of $\Delta \bar{W}$ on our S^{2}FT

Figure 11. Distribution analysis of the value projection matrix on the Resisc45 dataset.

Figure 12. Qualitative results on subject-driven generation task.