
Efficient Resource-Constrained Training of Vision Transformers via Subspace Optimization

Supplementary Material

1 A Additional Theoretical Details

2 A.1 Details of Backpropagation in Low-rank Subspace

3 For simplicity, in this section we denote the activation tensor \mathcal{A}_i as \mathcal{I} , the output gradient $\frac{\partial \mathcal{L}}{\partial \mathcal{W}_i}$ as $\Delta \mathcal{W}$, and the gradient with respect to the output $\frac{\partial \mathcal{L}}{\partial \mathcal{A}_{i+1}}$ as $\Delta \mathcal{Y}$.

5 **3D Activation Maps.** For each activation map $\mathcal{I} \in \mathbb{R}^{B \times N \times I}$, applying ASI with optimal rank $\mathbf{r} \in \mathbb{R}^3$, resulting in the following approximation:

$$\tilde{\mathcal{I}}_{b,n,i} = \sum_{r_1=1}^{\mathbf{r}_1} \sum_{r_2=1}^{\mathbf{r}_2} \sum_{r_3=1}^{\mathbf{r}_3} \tilde{S}_{r_1,r_2,r_3} \tilde{U}_{b,r_1}^{(1)} \tilde{U}_{n,r_2}^{(2)} \tilde{U}_{i,r_3}^{(3)} \quad (12)$$

7 which transforms the weight gradient calculation (Eq. (9) in the main paper) into:

$$\widetilde{\Delta \mathcal{W}_{o,i}} = \sum_{b=1}^B \sum_{n=1}^N \sum_{o=1}^O \sum_{i=1}^I \tilde{\mathcal{I}}_{b,n,i} \widetilde{\Delta \mathcal{Y}_{b,n,o}} \quad (13)$$

$$= \sum_{b=1}^B \sum_{n=1}^N \sum_{o=1}^O \sum_{i=1}^I \sum_{r_1=1}^{\mathbf{r}_1} \sum_{r_2=1}^{\mathbf{r}_2} \sum_{r_3=1}^{\mathbf{r}_3} \tilde{S}_{r_1,r_2,r_3} \tilde{U}_{b,r_1}^{(1)} \tilde{U}_{n,r_2}^{(2)} \tilde{U}_{i,r_3}^{(3)} \widetilde{\Delta \mathcal{Y}_{b,n,o}} \quad (14)$$

8 By reordering and grouping terms, we obtain:

$$\mathcal{Z}_{n,o,r_1}^{(1)} = \sum_{b=1}^B \widetilde{\Delta \mathcal{Y}_{b,n,o}} \tilde{U}_{b,r_1}^{(1)}, \quad (15)$$

$$\mathcal{Z}_{r_1,r_3,n}^{(2)} = \sum_{r_2=1}^{\mathbf{r}_2} \tilde{S}_{r_1,r_2,r_3} \tilde{U}_{n,r_2}^{(2)}, \quad (16)$$

$$\mathcal{Z}_{r_1,i,n}^{(3)} = \sum_{r_3=1}^{\mathbf{r}_3} \mathcal{Z}_{r_1,r_3,n}^{(2)} \tilde{U}_{i,r_3}^{(3)}, \quad (17)$$

$$\widetilde{\Delta \mathcal{W}_{o,i}} = \sum_{n=1}^N \sum_{r_1=1}^{\mathbf{r}_1} \mathcal{Z}_{n,o,r_1}^{(1)} \mathcal{Z}_{r_1,i,n}^{(3)}. \quad (18)$$

9 **4D Activation Maps.** In some transformer-based models, such as SwinT, the activation maps are 4D:
 10 $\mathcal{I} \in \mathbb{R}^{B \times H \times W \times I}$. When applying ASI with optimal rank $\mathbf{r} \in \mathbb{R}^4$, the activation map is approximated
 11 as:

$$\tilde{\mathcal{I}}_{b,h,w,i} = \sum_{r_1=1}^{\mathbf{r}_1} \sum_{r_2=1}^{\mathbf{r}_2} \sum_{r_3=1}^{\mathbf{r}_3} \sum_{r_4=1}^{\mathbf{r}_4} \tilde{S}_{r_1,r_2,r_3,r_4} \tilde{U}_{b,r_1}^{(1)} \tilde{U}_{h,r_2}^{(2)} \tilde{U}_{w,r_3}^{(3)} \tilde{U}_{i,r_4}^{(4)} \quad (19)$$

12 Reorganizing the terms yields:

$$\widetilde{\Delta\mathcal{W}}_{o,i} = \sum_{b=1}^B \sum_{h=1}^H \sum_{w=1}^W \sum_{o=1}^O \sum_{i=1}^I \tilde{\mathcal{L}}_{b,h,w,i} \widetilde{\Delta\mathcal{Y}}_{b,h,w,o} \quad (20)$$

$$= \sum_{b=1}^B \sum_{h=1}^H \sum_{w=1}^W \sum_{o=1}^O \sum_{i=1}^I \sum_{r_1=1}^{\mathbf{r}_1} \sum_{r_2=1}^{\mathbf{r}_2} \sum_{r_3=1}^{\mathbf{r}_3} \sum_{r_4=1}^{\mathbf{r}_4} \tilde{S}_{r_1,r_2,r_3,r_4} \tilde{U}_{b,r_1}^{(1)} \tilde{U}_{h,r_2}^{(2)} \tilde{U}_{w,r_3}^{(3)} \tilde{U}_{i,r_4}^{(4)} \widetilde{\Delta\mathcal{Y}}_{b,h,w,o} \quad (21)$$

13 Again, by reordering and grouping operator, Eq. (21) becomes:

$$\mathcal{Z}_{r_1,h,w,o}^{(1)} = \sum_{b=1}^B \widetilde{\Delta\mathcal{Y}}_{b,h,w,o} \tilde{U}_{b,r_1}^{(1)}, \quad (22)$$

$$\mathcal{Z}_{r_1,h,r_3,r_4}^{(2)} = \sum_{r_2=1}^{\mathbf{r}_2} \tilde{S}_{r_1,r_2,r_3,r_4} \tilde{U}_{h,r_2}^{(2)}, \quad (23)$$

$$\mathcal{Z}_{r_1,h,r_3,o}^{(3)} = \sum_{r_3=1}^{\mathbf{r}_3} \mathcal{Z}_{b,h,w,o}^{(1)} \tilde{U}_{w,r_3}^{(3)}, \quad (24)$$

$$\mathcal{Z}_{r_1,h,i,r_3}^{(4)} = \sum_{r_4=1}^{\mathbf{r}_4} \mathcal{Z}_{r_1,h,r_3,r_4}^{(2)} \tilde{U}_{i,r_4}^{(4)}, \quad (25)$$

$$\widetilde{\Delta\mathcal{W}}_{o,i} = \sum_{h=1}^H \sum_{r_1=1}^{\mathbf{r}_1} \sum_{r_3=1}^{\mathbf{r}_3} \mathcal{Z}_{r_1,h,r_3,o}^{(3)} \mathcal{Z}_{r_1,h,i,r_3}^{(4)}. \quad (26)$$

14 A.2 Details of Computational Speedup and Space Complexity

15 **Computational Speedup.** We derive the computational speedup as the ratio between the total FLOPs
16 required for vanilla training and those required by WASI.

17 First, the number of FLOPs required to perform forward pass in vanilla training (Eq. (1) in the main
18 paper) is:

$$F_{\text{vanilla}} \approx 2BN_i I_i O_i \quad (27)$$

19 Meanwhile, the backward pass includes Eq. (2) and Eq. (3) in the main paper, costing:

$$B_{\text{vanilla}} \approx 4BN_i I_i O_i \quad (28)$$

20 In contrast, WASI performs the forward pass in a low-rank subspace (Eq. (8) in the main paper) with
21 a complexity of:

$$F_{\text{WASI}} \approx 2BN_i K_i (I_i + O_i) \quad (29)$$

22 However, this does not account for the overhead from weight subspace decomposition (Algorithm 1
23 in the main paper) and ASI decomposition. These add the following costs:

$$O_{\text{WSI}} = 4I_i O_i K_i + 2O_i K_i^2, \quad (30)$$

$$O_{\text{ASI}} = \sum_{m=1}^3 (4dd' \mathbf{r}_{i,m} + 2d\mathbf{r}_{i,m}^2), \quad \text{where } d = \mathcal{D}_{i,m}, d' = \mathcal{D}_i \setminus \{d\} \quad (31)$$

24 The backward pass in WASI follows Eq. (10) in the main paper, Eq. (15), Eq. (16), Eq. (17), and
25 Eq. (18), with a total FLOPs cost of:

$$B_{\text{WASI}} = \underbrace{2BN_i K_i (I_i + O_i)}_{\text{Eq. (10) in the main paper}} + \underbrace{BN_i O_i \mathbf{r}_{i,1} + \mathbf{r}_{i,1} \mathbf{r}_{i,2} \mathbf{r}_{i,3} N_i + \mathbf{r}_{i,1} \mathbf{r}_{i,3} I_i N_i + \mathbf{r}_{i,1} I_i O_i N_i}_{\text{Eq. (15) to Eq. (18)}} \quad (32)$$

26 The speedup ratios between vanilla training and WASI are defined as:

$$S_{\text{training}} = \frac{F_{\text{vanilla}} + B_{\text{vanilla}}}{F_{\text{WASI}} + O_{\text{WSI}} + O_{\text{ASI}} + B_{\text{WASI}}} \quad (33)$$

$$S_{\text{inference}} = \frac{F_{\text{vanilla}}}{F_{\text{WASI}}} \quad (34)$$

27 **Memory Usage.** In vanilla training, the total memory consists of the memory for weights and the
 28 memory for storing activations:

$$M_{\text{vanilla}}^{(\mathcal{W}_i)} = I_i O_i \quad (35)$$

$$M_{\text{vanilla}}^{(\mathcal{A}_i)} = B N_i I_i \quad (36)$$

29 In WASI, these become:

$$M_{\text{WASI}}^{(\mathcal{W}_i)} = K_i (I_i + O_i) \quad (37)$$

$$M_{\text{WASI}}^{(\mathcal{A}_i)} = \prod_{m=1}^3 \mathbf{r}_{i,m} + \sum_{m=1}^3 \mathcal{D}_{i,m} \mathbf{r}_{i,m} \quad (38)$$

30 Thus, the memory reduction ratios between vanilla and WASI are:

$$C_{\text{training}} = \frac{M_{\text{vanilla}}^{(\mathcal{W}_i)} + M_{\text{vanilla}}^{(\mathcal{A}_i)}}{M_{\text{WASI}}^{(\mathcal{W}_i)} + M_{\text{WASI}}^{(\mathcal{A}_i)}} \quad (39)$$

$$C_{\text{inference}} = \frac{M_{\text{vanilla}}^{(\mathcal{W}_i)}}{M_{\text{WASI}}^{(\mathcal{W}_i)}} \quad (40)$$

31 Similar ratios can be derived for the case of 4D activation maps.

32 A.3 Limitations of SVD-LLM

33 We analyze the 3D activation map $\mathcal{A}_i \in \mathbb{R}^{B \times N_i \times I_i}$, as considered throughout the paper. The core
 34 idea of SVD-LLM is to incorporate ‘‘Truncation-aware Data Whitening’’ mechanism that enables a
 35 direct relationship between singular values and the resulting compression loss.

36 To do this, SVD-LLM whitens the activation using a transformation of the form $S_i^{-1} X_i$, where
 37 $X_i \in \mathbb{R}^{N_i \times I_i}$ is obtained by summing \mathcal{A}_i over the batch dimension. The goal is to make the
 38 transformed activation orthonormal, i.e., $(S_i^{-1} X_i)(S_i^{-1} X_i)^T = I$. Here, S_i is computed via Cholesky
 39 decomposition on X .

40 SVD is then applied to the transformed weight matrix $\mathcal{W}_i S_i$, resulting in U_i , Σ_i , and V_i . Based on
 41 the optimal rank K_i found by a desired compression ratio, the smallest singular values in Σ_i are
 42 truncated (denoted by $\Sigma_{i,(K_i)}$) to obtain two low-ranking matrices:

$$\mathcal{W}'_i^{(u)} = U_{i,(K_i)} \Sigma_{i,(K_i)}^{1/2}, \quad \mathcal{W}'_i^{(v)} = \Sigma_{i,(K_i)}^{1/2} V_{i,(K_i)}^T S_i^{-1} \quad (41)$$

43 and the final compressed matrix is given by:

$$\tilde{\mathcal{W}}_i = \mathcal{W}'_i^{(u)} \mathcal{W}'_i^{(v)} = U_{i,(K_i)} \Sigma_{i,(K_i)} V_{i,(K_i)}^T S_i^{-1}. \quad (42)$$

44 However, a current limitation of SVD-LLM is that ‘‘Truncation-aware Data Whitening’’ is only
 45 defined for 3D activation maps. It does not generalize to 4D activations, which are common in certain
 46 architectures such as SwinT. As a result, SVD-LLM cannot be directly applied to models that rely on
 47 4D activation structures.

48 B Additional Experimental Details

49 B.1 Details of Experimental Setup

50 To ensure a fair comparison, we follow the same experimental setup as described in [3]. The key
 51 details are as follows:

52 **General hyperparameters.** All models are first pretrained on ImageNet-1K, then fine-tuned on a
 53 different downstream dataset using an 80% – 20% train-validation split. We use cross-entropy loss

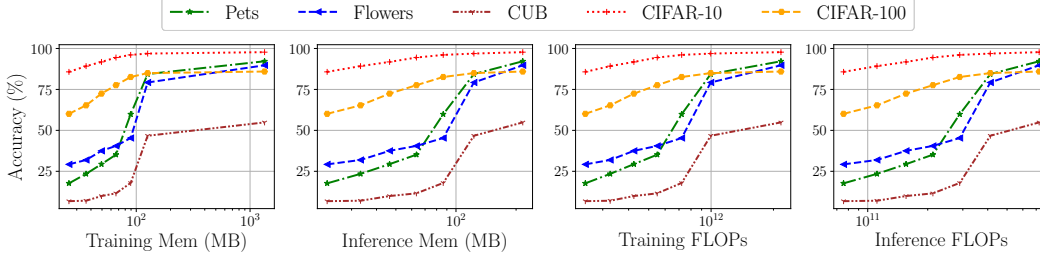


Figure 8: WASI performance when fine-tuning ViT across multiple datasets. In each plot, markers from left to right represent increasing values of ε ; the rightmost marker corresponds to vanilla training.

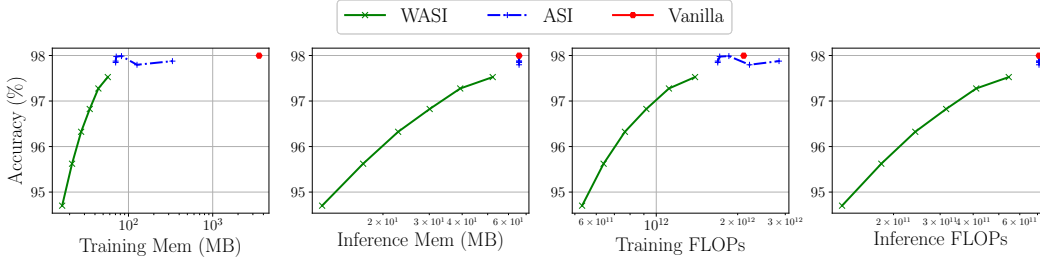


Figure 9: Comparison of different methods when fine-tuning SwinT on CIFAR-10. For each plot, markers from left to right correspond to increasing values of ε .

and optimize the models with SGD. The initial learning rate is set to 0.05 and decayed using a cosine annealing schedule. Momentum is set to 0, and weight decay is fixed at 1×10^{-4} . We apply L2 gradient clipping with a threshold of 2.0. For data augmentation, we use random resizing, horizontal flipping, normalization, and a mini-batch size of 128.

ASI. We follow the same strategy as proposed in [7]. Specifically, in our main experiments, we apply AMC with a range of ε values: 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9. For each value of ε , we record the peak activation memory consumption of AMC and use it as the activation memory budget for ASI. The perplexity of ASI is also measured based on these ε values.

SVD-LLM. Unlike WASI, SVD-LLM compresses models based on a fixed compression ratio. To compare fairly, we compute the compression ratios achieved by WASI at each ε and use the same ratios for SVD-LLM. We also adopt the same LoRA adapter settings as used in their original paper: $\alpha = 16$ and rank = 8.

B.2 Variance Across Different Random Seeds

Fig. 10 presents the results of fine-tuning a ViT model pretrained on ImageNet-1K using WASI on Pets dataset. We test different values of ε from $\{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$, with each marker on the plot (from left to right) corresponding to one of these settings. For each ε , we report the average accuracy and peak training memory usage over three random seeds (233, 234, and 235), with error bars representing standard deviation.

As expected, increasing ε leads to higher accuracy and greater memory usage, reflecting the trade-off between compression and performance. Importantly, the variance across seeds is minimal, which is reasonable given that WASI mainly uses deterministic components such as

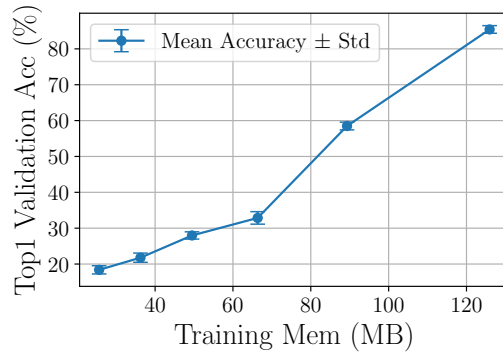


Figure 10: WASI performance on the Pets dataset with different ε values, showing mean accuracy and memory usage across three random seeds. Error bars represent standard deviation.

83 SVD, Gram-Schmidt orthogonalization, and matrix multiplications. Therefore, we fix the random
84 seed to 233 in all other experiments.

85 B.3 Additional Results

86 **Additional Transformer-based Results.** We conduct experiments similar to those in Sec. 4.3. Fig. 8
87 and Fig. 9 show consistent trends with our earlier findings. The accuracy of WASI improves steadily
88 as ε increases, demonstrating the effectiveness of controlling compression error through the explained
89 variance threshold. Overall, WASI achieves up to one order of magnitude reduction in training
90 memory when fine-tuning ViT, with similarly favorable results observed in terms of computational
91 cost and inference memory.

92 **WSI on Convolutional Neural Network.** In
93 this experiment, we extend the application
94 scope of WSI to convolutional neural networks.
95 Specifically, we use MCUNet [2], pretrained on
96 ImageNet-1K, with Pets dataset as the down-
97 stream task. WSI is applied to fine-tune the last
98 1 to 4 convolutional layers of the model. When
99 $\varepsilon = 0.9$, applying WSI unexpectedly increases
100 the weight memory. This is due to the fact that
101 the optimal rank found is too high, resulting
102 in principal components whose total number of
103 elements exceeds that of the original weight ten-
104 sor. In contrast, for smaller values of ε (0.75
105 and 0.8), applying WSI to more layers leads to
106 reduced memory usage for the weights, as ex-
107 pected, at the cost of some accuracy degradation.
108 However, this trade-off is not worthwhile, as
109 the memory savings are marginal and the con-
110 volutional layers are already highly compact by
111 design.

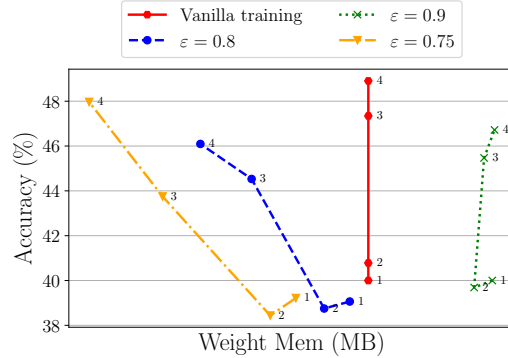


Figure 11: Performance of WSI when applied to fine-tune MCUNet (pretrained on ImageNet-1K) on Pets dataset. The number next to each marker indicates how many convolutional layers WSI was applied to.

112 C Limitations

113 Our goal is to enable the training of transformer models on edge devices. Thus far, our experiments
114 have focused primarily on vision tasks, which facilitates direct comparison with existing work in this
115 domain. In future work, we plan to extend our approach to a wider range of tasks, including LLMs.