Expanding Sparse Tuning for Low Memory Usage

Shufan Shen[1,2]   **Junshu Sun**[1,2]   **Xiangyang Ji**[3]   **Qingming Huang**[1,2,4]   **Shuhui Wang**[1,4*]

[1]Key Lab of Intell. Info. Process., Inst. of Comput. Tech., CAS
[2]University of Chinese Academy of Sciences   [3]Tsinghua University   [4]Peng Cheng Laboratory
{shenshufan22z, sunjunshu21s, wangshuhui}@ict.ac.cn   xyji@tsinghua.edu.cn
qmhuang@ucas.ac.cn

Abstract

Parameter-efficient fine-tuning (PEFT) is an effective method for adapting pre-trained vision models to downstream tasks by tuning a small subset of parameters. Among PEFT methods, sparse tuning achieves superior performance by only adjusting the weights most relevant to downstream tasks, rather than densely tuning the whole weight matrix. However, this performance improvement has been accompanied by increases in memory usage, which stems from two factors, *i.e.*, the storage of the whole weight matrix as learnable parameters in the optimizer and the additional storage of tunable weight indexes. In this paper, we propose a method named SNELL (**S**parse tuning with ker**NEL**ized **L**oRA) for sparse tuning with low memory usage. To achieve low memory usage, SNELL decomposes the tunable matrix for sparsification into two learnable low-rank matrices, saving from the costly storage of the whole original matrix. A competition-based sparsification mechanism is further proposed to avoid the storage of tunable weight indexes. To maintain the effectiveness of sparse tuning with low-rank matrices, we extend the low-rank decomposition by applying nonlinear kernel functions to the whole-matrix merging. Consequently, we gain an increase in the rank of the merged matrix, enhancing the ability of SNELL in adapting the pre-trained models to downstream tasks. Extensive experiments on multiple downstream tasks show that SNELL achieves state-of-the-art performance with low memory usage, endowing PEFT with sparse tuning to large-scale models. Codes are available at https://github.com/ssfgunner/SNELL.

1 Introduction

Fine-tuning has become a predominant way for adapting large pre-trained models to downstream tasks with limited training samples [13, 9, 24, 23]. Nevertheless, fine-tuning all model parameters requires substantial memory usage and is susceptible to over-fitting, making it costly and infeasible for large-scale models [58, 2, 11]. To address these limitations, parameter-efficient fine-tuning (PEFT) [64, 27, 62, 30, 8, 22] has been proposed to tune a small subset of parameters while keeping other parameters frozen. PEFT methods can be categorized into addition-based and reparameterization-based methods. The former attaches additional parameters to a frozen pre-trained backbone, while the latter adjusts the original parameters in the pre-trained backbone.

Addition-based methods [53, 62, 30] have achieved remarkable performance on vision tasks. However, adopting additional parameters incurs extra computational costs during the inference process. In contrast, reparameterization-based methods [5, 7, 27] directly fine-tune the original parameters. These methods select specific parameters, involving reduced memory usage compared to full-parameter fine-tuning. Based on the granularity of parameter selection, one primary approach focuses on
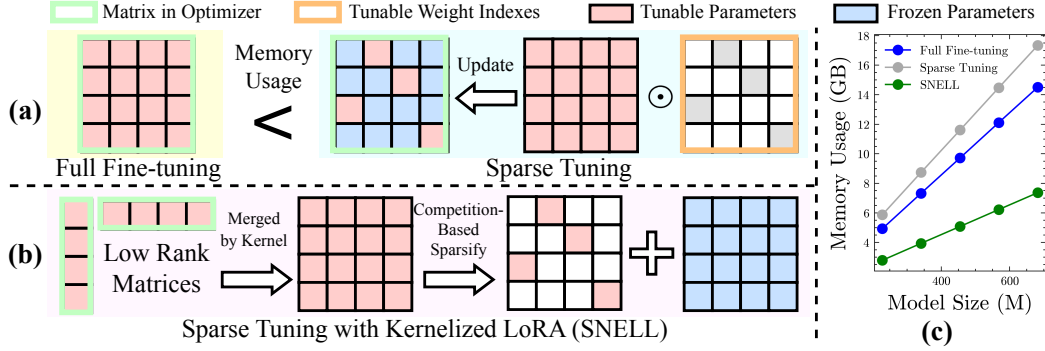

Figure 1: (a) The high memory usage of sparse tuning arises from taking the whole weight matrix as learnable parameters, in addition to the storage of the tunable weight indexes (typically represented as a binary mask). (b) Our framework (SNELL) only stores the learnable low-rank matrices in the optimizer. (c) Memory usage comparison on pre-trained models with different depths.

specific parameter matrices. For example, Bitfit [5] only adjusts bias to reduce the volume of tunable parameters while Partial-k [30] fine-tunes the last few layers to avoid back-propagation through the entire pre-trained backbone. To further reduce memory usage, LoRA [27] optimizes each selected weight matrix using two low-rank matrices to achieve memory-efficient fine-tuning. Although sufficient in reducing memory usage, these methods usually gain inferior performance compared to addition-based methods [30]. Recently, SPT [22] and GPS [63] found that combining existing PEFT methods with sparse tuning, which only adjusts the most task-related weights in a matrix, can achieve state-of-the-art performance on vision tasks. Concurrently, the effectiveness of sparse tuning has also been observed in NLP tasks [18]. By focusing on individual weights in a matrix, sparse tuning allows for more precise adjustments, thus achieving good performance and mitigated over-fitting risks [18].

However, the performance gained from sparse tuning has been accompanied by high memory usage, as Figure 1(a) shows. Although sparse tuning only updates part of weights in the pre-trained weight matrix, the whole matrix still needs to be stored as learnable parameters in the optimizer and computed for their corresponding gradients in practice. Additionally, sparse tuning necessitates storing the tunable weight indexes, further aggravating the memory demands. The above observation indicates that sparse tuning gains no advantage over full fine-tuning regarding memory usage, especially given the increasing parameter volumes in pre-trained models [58, 2]. A sparse tuning method with low memory usage is urgently required for applications on large-scale pre-trained models.

In this paper, we propose a method that conducts **S**parse tuning with ker**NEL**ized **L**oRA (SNELL) shown in Figure 1(b). SNELL can adapt pre-trained models to downstream tasks with both low memory usage and strong performance. To reduce memory usage, we decompose the tunable matrix for sparsification into low-rank learnable matrices to store fewer parameters in the optimizer and develop a competition-based method to avoid storing the tunable weight indexes. To improve the performance on downstream tasks, we extend LoRA from a kernel perspective and merge low-rank matrices with nonlinear kernel functions to obtain matrices with higher ranks.

Specifically, SNELL updates the pre-trained weight matrix using a sparse low-rank adaptation matrix. This adaptation matrix is first merged with two low-rank learnable matrices and then sparsified toward effective fine-tuning. Compared to storing the whole adaptation matrix, storing low-rank matrices in the optimizer results in lower memory usage. For the sparsification process, we propose a competition-based method inspired by the neuron competition phenomenon in neuroscience [49], avoiding the storage of the tunable weight indexes that incur additional memory usage. The proposed method promotes competition among weights based on their absolute values. Most task-relevant weights are encouraged to have larger absolute values and survive during the fine-tuning process. By setting a sparsity ratio as the hyperparameter and determining tunable weights based on their absolute values in an end-to-end manner, we can eliminate the storage of the tunable weight indexes.

In addition to low memory usage, the performance is also critical for model fine-tuning. However, directly merging two low-rank matrices through the inner product leads to the low-rank structure of the adaptation matrix, which narrows the optimization scope of tunable matrices and further limits the expressiveness of sparse tuning. To overcome this bottleneck, we draw inspiration from DyN [45]

on weight matrix interpretation based on low-dimensional dynamical systems, and reformulate the merging process with nonlinear kernel functions that increase the rank of the merged adaptation matrix. This new formulation enables a more expressive sparse tuning while maintaining a compact representation with low memory.

Extensive experiments are conducted on 24 downstream visual recognition tasks with both plain and hierarchical vision Transformer backbones under supervised and self-supervised pre-training. Results show that SNELL can gain the performance improvement of sparse tuning and the low memory usage of LoRA concurrently. SNELL obtains the state-of-the-art performance on FGVC (91.8% *vs.* 90.7%) and VTAB-1k (74.6% *vs.* 74.1%) benchmark with LoRA-level memory usage. Moreover, as Figure 1(c) shows, the low memory-usage advantage of SNELL becomes increasingly apparent as the model size grows, enabling sparse tuning on larger models.

2 Related Work

Parameter Sparsity. In early work, the parameter sparsity usually serves as an optimization objective in model pruning [21, 42]. These pruning methods remove the weights from pre-trained models irrelevant to a specific task, without significantly degrading model performance. The relevance of individual weights can be estimated based on activations [28], redundancy [48], per-layer second derivatives [15], and energy efficiency [57]. Except for the post-training pruning strategy, sparse networks [4, 41, 17] directly introduce parameter sparsity into the training process, removing redundant weights more precisely [17]. Motivated by the advantage of parameter sparsity in model optimization, recent studies introduce sparsity to the fine-tuning of pre-trained models and achieve enhanced model performance on downstream tasks [1, 21, 56]. The parameter sparsity gives rise to a reduced number of trainable parameters and serves as a regularization constraint during fine-tuning [18]. Among sparse tuning, pre-pruning methods adopt model pruning for fine-tuning. These methods sparsify the weight matrix [22, 63] or adapter [1] through pruning metrics [43, 17] to identify learnable parameters for the fine-tuning process. Other methods select trainable parameters during fine-tuning, including learnable mask [64] or diff vectors [20] with sparsity constraints. However, the parameter sparsification methods need to store the indexes of tunable weights, which incurs additional memory usage. For sparse tuning under low memory budget, our competition-based mechanism selects weights relevant to downstream tasks in a learnable manner without storing the tunable weight indexes.

Parameter-efficient Fine-tuning. Fine-tuning is the most predominant approach for adapting a pre-trained model to downstream tasks. However, for large pre-trained models, fine-tuning all parameters is costly and prone to overfit downstream datasets. To tackle these problems, parameter-efficient fine-tuning (PEFT) [8, 30, 62], which tunes only a tiny portion of parameters, becomes a desirable choice. Following the taxonomy of SPT [22], PEFT methods can be categorized into *addition-based* [3, 26, 46, 50, 14, 31, 37, 62] and *reparameterization-based* [5, 7, 20, 64, 27] methods.

Addition-based methods attach additional trainable parameters to a frozen pre-trained backbone. Adapters [3, 26, 46, 50, 61] adopt a residual pathway and learn a bottleneck layer including two linear projections and a non-linear activation. Prompt-tuning methods [14, 31, 37, 34] add trainable parameters to the input and keep the entire pre-trained model unchanged during training. Recent work [62] attempts to find the optimal configurations to combine multiple addition-based methods. Despite of the popularity and effectiveness of *addition-based* methods, the additional trainable parameters incur excess computational costs during the inference process [3, 33].

Reparametization-based methods adjust the inherent parameters in the pre-trained backbone to avoid excess computational costs during inference. Early work directly selects parameters with low memory usage for fine-tuning, such as the bias terms [5] and the final few layers of the pre-trained model [7]. To further reduce the memory usage of the selected matrices, LoRA [27] optimizes low-rank matrices that can be reparameterized into the pre-trained weight matrices to reduce memory usage. Exploring finer-grained parameter selection, some studies [20, 64] propose sparse tuning, which involves selecting and tuning individual weights sparsely within the weight matrices. Recently, SPT [22] combines sparse tuning and LoRA in a hybrid framework that achieves state-of-the-art performances on visual PEFT tasks. SPT has revealed that optimizing the weights most relevant to the downstream task through sparse tuning can significantly enhance the performance, which is also supported by SAM [18] and GPS [63]. However, existing sparse tuning framework still faces the challenge of high memory usage brought by sparse tuning. Unlike existing methods, our SNELL

inherits high performance and low memory usage concurrently by sparsifying the adaptation matrix merged with low-rank matrices through nonlinear kernels.

3 Methodology

We first introduce the definitions of sparse tuning [64, 20, 1], LoRA [27] and kernel trick [32] (Section 3.1). Then we propose SNELL, a sparse tuning method including kernelized LoRA that enables high-performance tuning with low-rank learnable matrices (Section 3.2) and a competition-based mechanism that sparsifies weights without additional memory usage (Section 3.3).

3.1 Preliminaries

Sparse Tuning. Given a downstream training set $\mathcal{D} = \{x^{(n)}, y^{(n)}\}_{n=1}^{N}$, the objective of sparse tuning is to minimize the model's empirical risk on the downstream task, with the sparsity constraints on the volume of tunable weights in weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$. The sparsification is usually achieved through a binary mask $\mathbf{M} \in \{0, 1\}^{m \times n}$. The objective function can be formulated as

$$\min_{\mathbf{W} \odot \mathbf{M}} \frac{1}{N} \sum_{n=1}^{N} \mathcal{L} \left(f(x^{(n)}; \mathbf{W}), y^{(n)} \right) \tag{1}$$

where $f(\cdot; \cdot)$ is a parameterized function over the input (*e.g.*, a neural network), $\mathcal{L}(\cdot, \cdot)$ is a loss function (*e.g.*, cross-entropy), and $\odot$ denotes element-wise multiplication. The binary mask $\mathbf{M}$ can be either a fixed hyperparameter, pre-computed with heuristics such as pre-pruning [22], or a learnable parameter obtained through end-to-end fine-tuning [64]. All these methods require storing $\mathbf{M}$ to determine the tunable weights, which results in additional memory usage. More importantly, the tunable parameters $\mathbf{W} \odot \mathbf{M}$ occupy the same amount of memory as the weight matrix $\mathbf{W}$ in practice. As a result, the memory usage of sparse tuning is even higher than that of full fine-tuning.

LoRA. Given a pre-trained weight matrix $\mathbf{W}_0$, LoRA [27] optimizes two low-rank matrices $\mathbf{B} \in \mathbb{R}^{m \times r}, \mathbf{A} \in \mathbb{R}^{n \times r}$ to reduce the memory usage during fine-tuning. The low-rank matrices $\mathbf{A}$ and $\mathbf{B}$ can be reparameterized into the pre-trained weight $\mathbf{W}_0$,

$$\mathbf{W} = \mathbf{W}_0 + \Delta \mathbf{W} = \mathbf{W}_0 + \mathbf{B} \mathbf{A}^{\top} \tag{2}$$

With $r \ll \min(m, n)$, LoRA can achieve high training efficiency and low memory usage by only optimizing the smaller low-rank matrices.

Kernel Trick [32]. In many machine learning tasks, mapping the vectors into higher dimensions is frequently used to achieve linear separability [51]. However, the explicit mapping process incurs significant computational costs. To address this problem, the kernel trick is proposed to efficiently model data relationships in high-dimensional spaces, without the need to explicitly formulate the space. According to Mercer's theorem [6], a kernel function $\kappa : \mathbb{R}^r \times \mathbb{R}^r \to \mathbb{R}$ can express an inner product in some space as $\kappa(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^{\top} \phi(\mathbf{x}')$, if and only if κ is positive semi-definite (Appendix B). $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^r$, and $\phi : \mathbb{R}^r \to \mathbb{R}^d$ is an implicit feature map. By selecting an appropriate kernel function κ, we can obtain the inner product of two vectors in higher-dimensional space $\mathbb{R}^d$ ($d \geq r$) without explicitly formulating the feature map ϕ.

3.2 Kernelized LoRA

We leverage LoRA to reduce the memory usage of sparse tuning in light of its low memory usage. An intuitive solution is to sparsify the adaptation matrix $\Delta \mathbf{W}$ composed of the two low-rank matrices. However, the low-rank property of $\Delta \mathbf{W}$ can lead to the performance degradation of sparse tuning. For the original sparse tuning, the weight matrix $\mathbf{W}$ is free of the rank constraint, and weights are independent of each other. Therefore, we can independently select and optimize weights most relevant to the downstream task. For sparse tuning with LoRA, the adaptation matrix $\Delta \mathbf{W}$ with rank r is constrained in $\mathbb{R}^{r \times (m+n-r)}$, a subspace of $\mathbb{R}^{m \times n}$. When $r \ll \min(m, n)$, the weight optimization scope of sparse tuning contracts, hindering its performance on downstream tasks.

To achieve sparse tuning with both strong performance and low memory usage, we propose to construct a high-rank matrix using low-rank matrices. Inspired by DyN [45] that fits a high-rank
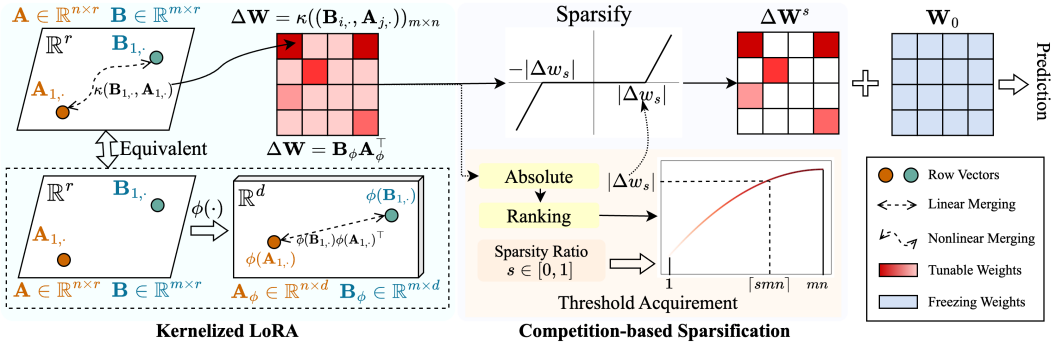
4

Figure 2: Overview of our SNELL strategy. Given two learnable low-rank matrices, we merge them using a non-linear kernel function (*left*). This merging process is equivalent to mapping the matrices to higher-rank matrices and then performing matrix multiplication. Then we sparsified this merged adaptation matrix using a competition-based sparsification mechanism (*right*). This mechanism zeros out weights with small absolute values based on the specified percentage of s.

matrix using the distance matrix of a low-dimension dynamical system, we extend the distance function to general kernel functions and investigate LoRA in the kernel perspective. Given two vectors $x, x' \in \mathbb{R}^r$, the kernel function $\kappa(x, x')$ can be formulated as an inner product $\phi(x)^\top \phi(x')$ with an implicit feature map $\phi : \mathbb{R}^r \to \mathbb{R}^d$. The merging process of LoRA can be seen as applying linear kernel function $\kappa_l(\cdot, \cdot)$ on the rows of the learnable parameters $\mathbf{A}$ and $\mathbf{B}$,

$$\Delta\mathbf{W}_{ij} = \kappa_l(\mathbf{A}_{i,\cdot}, \mathbf{B}_{j,\cdot}) = \phi_l(\mathbf{B}_{j,\cdot})\phi_l(\mathbf{A}_{i,\cdot})^\top = \mathbf{B}_{j,\cdot}\mathbf{A}_{i,\cdot}^\top, \tag{3}$$

where $\mathbf{A}_{i,\cdot}, \mathbf{B}_{j,\cdot} \in \mathbb{R}^r$, $\phi_l : \mathbb{R}^r \to \mathbb{R}^r$ denotes the identity mapping. By replacing $\kappa_l(\cdot, \cdot)$ with more complex non-linear kernel functions, we can approximate relations in higher-dimensional spaces $\mathbb{R}^d$ and obtain matrices with rank larger than r. The merged adaptation matrix in SNELL can be represented by

$$\Delta\mathbf{W} = (\kappa(\mathbf{A}_{i,\cdot}, \mathbf{B}_{j,\cdot}))_{m \times n} = [\phi(\mathbf{B}_{1,\cdot})^\top, ..., \phi(\mathbf{B}_{n,\cdot})^\top]^\top[\phi(\mathbf{A}_{1,\cdot})^\top, ..., \phi(\mathbf{A}_{n,\cdot})^\top] = \mathbf{B}_\phi\mathbf{A}_\phi^\top. \tag{4}$$

Note that in practice, explicit computation of $\mathbf{A}_\phi \in \mathbb{R}^{n \times d}$ and $\mathbf{B}_\phi \in \mathbb{R}^{m \times d}$ is unnecessary. $\Delta\mathbf{W}$ can be directly derived based on $\mathbf{A}$ and $\mathbf{B}$ with the kernel function κ. By extending LoRA in a kernel perspective, SNELL can build high-rank adaptation matrices based on low-rank learnable matrices, empowering strong sparse tuning with low memory usage. We utilize the piecewise linear kernel introduced in Appendix B without a specific statement.

3.3 Competition-based Sparsification Mechanism

Existing methods store tunable weight indexes $\mathbf{M} \in \{0, 1\}^{m \times n}$ for sparsifying the update of the weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$. The storage of $\mathbf{M}$ leads to additional memory usage. Inspired by the neuron competition phenomenon in neuroscience [49], we design a competition-based parameter sparsification mechanism to avoid this additional storage. Instead of determining the learnable weights in the optimization process based on $\mathbf{M}$, our objective is to encourage the weights to compete based on their contributions to performance improvement. Weights with stronger contributions survive in the sparsification while the remaining low-contributed weights are zeroed out. The weight contribution is reflected in their absolute values during the end-to-end optimization. During optimization, weights contributing more to the loss reduction are encouraged to have more significant values, while weights contributing less approach zero. By retaining higher importance to significant weights and zeroing out the less impactful weights, we can achieve end-to-end tunable parameter selection by solely relying on the absolute values of weights, avoiding the storage of $\mathbf{M}$.

Specifically, given a merged adaptation matrix $\Delta\mathbf{W}$ and a sparsity ratio $s \in [0, 1]$, we sparsify weights with a soft-threshold function. To induce weight competition during end-to-end fine-tuning, we propose a dynamic threshold Δw_s, *i.e.*, the weight having the $\lceil smn \rceil$-th smallest absolute value in $\Delta\mathbf{W}$. This threshold ensures that only a fixed proportion ($s \times 100\%$) of weights remain non-zero. Therefore, the weights have to compete with each other to be selected instead of just having a larger absolute value than a fixed threshold.

$$\Delta\mathbf{W}_{ij}^s = \Delta\mathbf{W}_{ij}\max(|\Delta\mathbf{W}_{ij}| - |\Delta w_s|, 0), \tag{5}$$

where $\Delta\mathbf{W}^s = (\Delta\mathbf{W}^s_{ij})_{m \times n}$ denotes the sparse matrix with sparsity ratio s. In practice, the sparsity ratio s is manually determined regarding specific downstream tasks. Given a sparsity ratio s, the training objective in Equation 1 can be reformulated as

$$\min_{\mathbf{A},\mathbf{B}} \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}\left(f(x^{(n)}; \mathbf{W}_0 + \Delta\mathbf{W}^s), y^{(n)}\right). \tag{6}$$

This objective encourages weights that are most relevant to the downstream task to gain more significant values for survival. Adjusting the sparsity ratio s allows us to control the sparsification process precisely and identify the optimal number of tunable parameters for different tasks.

4 Experiments

4.1 Experimental Setup

Datasets and Metrics. We evaluate our methods on 24 downstream tasks categorized into two groups following SPT [22]. *(i) FGVC* [30] is a benchmark for fine-grained image classification. This benchmark includes 5 downstream tasks, which are CUB-200-2011 [55], NABirds [25], Oxford Flowers [44], Stanford Dogs [19] and Stanford Cars [12]. We follow the validation splits in [22] if the official validation set is unavailable. *(ii) VTAB-1k* [59] is a large-scale transfer learning benchmark consisting of 19 visual classification tasks. VTAB-1k can be further divided into three groups, *i.e.*, natural tasks with natural images, specialized tasks with images captured by specialized equipment, and structured tasks with images mostly generated from synthetic environments. We use top-1 accuracy averaged within each group as our main metric following [22].

Pre-trained Backbones. We conduct experiments on the plain vision Transformer backbone ViT-B/16 [16] that is pre-trained on ImageNet [47] with different pre-training strategies following [22], including supervised pre-training and self-supervised pre-training with MAE [23] and MoCo v3 [10]. We also conduct experiments on the representative hierarchical vision Transformer backbone Swin-B [38] and CNN backbone ConvNeXt-Base [39] under supervised pre-training. In addition, we fine-tune the supervised pre-trained large-scale models (ViT-L/16 [16], ViT-H/14 [16]) on VTAB-1k to demonstrate the memory-efficiency and high-performance of SNELL.

Competitors. We compare our methods with addition-based methods including MLP-k, VPT-Shallow [30], VPT-Deep [30], Adapter-r [26], and SPT-Adapter [22]. For reparameterization-based methods, we compare with Linear, Partial-1, Bias [5], LoRA-r [27], SSF [35], and SPT-LoRA [22]. Here r represents the number of bottleneck dimensions in Adapter-r and the value of rank in LoRA-r and our proposed SNELL-r. Details of the competitors are presented in Appendix A.1. We also provide additional comparisons with other approaches [63, 53] in Appendix C.1.

Implementation Details. Following SPT [22], we use the AdamW optimizer [40] with cosine learning rate decay. The batch size, learning rate, and weight decay are 32, $1e-3$, and $1e-4$, respectively. We also follow SPT [22] to implement the standard data augmentation pipeline for VTAB-1K and follow SSF [35] for FGVC as well. SNELL is applied on the pre-trained weight matrix of all linear layers. For each task, we fine-tune the model with different sparsity ratios s to search the optimal volume of tunable parameters for this task. Without specific stating, we adopt the piecewise linear kernel (introduced in Appendix B) as the kernel function for SNELL. Ablation studies on different kernel functions are presented in Figure 4.

4.2 Performance on Downstream Tasks

Performance on Different Benchmarks. Experiments on FGVC and VTAB-1k benchmarks indicate that SNELL achieves the best performance with supervised pre-trained ViT-B/16 backbone as shown in Table 1. SNELL gains large performance improvements over LoRA variants, *e.g.*, SNELL-8 surpasses LoRA-8 significantly by 5.5% in terms of mean accuracy on the FGVC benchmark. Moreover, SNELL outperforms the state-of-the-art method SPT-LoRA by a clear margin of 0.5% in terms of mean top-1 accuracy on the VTAB-1k benchmark. This stems from the fact that SPT-LoRA only performs sparse tuning on a portion of the weight matrices while employing LoRA for the remaining part. In contrast, the low memory property of SNELL empowers sparse tuning on all the weight matrices, allowing for more precise adjustments and giving rise to superior performance.

Table 1: Top-1 accuracy (%) on FGVC and VTAB-1k benchmarks using ViT-B/16 pre-trained on ImageNet-21k supervisedly. The best result is in **bold**, and the second-best result is underlined.

Method	FGVC						VTAB-1k			
	CUB-200	NABirds	Oxford Flowers	Stanford Dogs	Stanford Cars	Mean Acc.	Natural	Specialized	Structured	Mean Acc.
Full	87.3	82.7	98.8	89.4	84.5	88.5	75.9	83.4	47.6	69.0
Additional-based methods										
MLP-3 [30]	85.1	77.3	97.9	84.9	53.8	79.8	67.8	72.8	30.6	57.1
VPT-Shallow [30]	86.7	78.8	98.4	90.7	68.7	84.6	76.8	79.7	47.0	67.8
VPT-Deep [30]	88.5	84.2	99.0	90.2	83.6	89.1	78.5	82.4	55.0	72.0
Adapter-8 [26]	87.3	84.3	98.4	88.8	68.4	85.5	79.0	84.1	58.5	73.9
Adapter-32 [26]	87.2	84.3	98.5	89.6	68.4	85.6	79.6	84.0	58.3	74.0
SPT-Adapter [22]	89.1	83.3	99.2	91.1	86.2	89.8	82.0	85.8	61.4	76.4
MoSA [60]	89.3	85.7	99.2	91.9	83.4	89.9	79.9	84.0	50.3	71.4
Reparameter-based methods										
Linear [30]	85.3	75.9	97.9	86.2	51.3	79.3	68.9	77.2	26.8	57.6
Partial-1 [30]	85.6	77.8	98.2	85.5	66.2	82.6	69.4	78.5	34.2	60.7
Bias [5]	88.4	84.2	98.8	91.2	79.4	88.4	73.3	78.3	44.1	65.2
LoRA-8 [27]	84.9	79.0	98.1	88.1	79.8	86.0	79.5	84.6	60.5	74.9
LoRA-16 [27]	85.6	79.8	98.9	87.6	72.0	84.8	79.8	84.9	60.2	75.0
SPT-LoRA [22]	88.6	83.4	99.5	91.4	87.3	90.1	81.9	85.9	61.3	76.4
SSF [35]	89.5	85.7	99.6	89.6	89.2	90.7	81.6	86.6	59.0	75.7
SNELL-8 (ours)	89.6	86.8	99.3	92.1	89.9	91.5	82.0	85.7	61.6	76.4
SNELL-16 (ours)	89.9	87.0	99.3	92.2	90.3	91.7	82.4	86.1	61.7	76.7
SNELL-32 (ours)	89.9	87.0	99.4	92.0	90.5	91.8	82.7	86.1	61.8	76.9

Table 2: Top-1 accuracy (%) on VTAB-1k benchmarks using ViT-B/16 backbone pre-trained on ImageNet using MAE and MoCo v3 strategies. The best result is in **bold**.

Methods	VTAB-1k MAE				VTAB-1k MoCo v3			
	Natural	Specialized	Structured	Mean Acc.	Natural	Specialized	Structured	Mean Acc.
Full	59.3	79.7	53.8	64.3	72.0	84.7	42.0	69.6
Additional-based methods								
Adapter-8 [26]	57.2	78.4	54.7	63.4	27.6	70.9	48.4	49.0
Adapter-32 [26]	55.3	78.8	53.3	62.5	74.2	82.7	47.7	68.2
VPT-Shallow [30]	40.0	69.7	27.5	45.7	67.3	82.3	37.6	62.4
VPT-Deep [30]	36.0	60.6	26.6	41.1	70.3	83.0	42.4	65.2
SPT-Adapter [22]	65.6	82.7	60.7	69.7	76.6	85.0	61.7	74.4
Reparameterization-based methods								
Linear [30]	18.9	52.7	23.7	32.1	67.5	81.1	30.3	59.6
Partial-1 [30]	58.4	78.3	47.6	61.5	72.3	84.6	47.9	68.3
Bias [5]	54.6	75.7	47.7	59.3	72.9	81.1	53.4	69.2
LoRA-8 [27]	57.5	77.7	57.7	64.3	21.2	66.7	45.1	44.3
LoRA-16 [27]	57.3	77.1	59.9	64.8	16.0	64.0	48.7	42.9
SPT-LoRA [22]	65.4	82.4	61.5	69.8	76.5	**86.0**	63.6	75.3
SNELL-8 (ours)	**68.3**	**83.8**	**63.5**	**71.8**	**76.8**	**86.0**	**63.7**	**75.5**

Performance on Different Pre-training Strategies. Experimental results on models pre-trained using different strategies are presented in Table 2. SNELL outperforms the state-of-the-art performances on models pre-trained with MAE (71.8% *vs.* 69.8%) and MoCo v3 (75.5% *vs.* 75.3%). Furthermore, SNELL consistently outperforms other PEFT methods on every group of downstream datasets. This demonstrates the general effectiveness of SNELL under different pre-training strategies.

Performance on Different Architectures. Following VPT [30] and SPT [22], we apply SNELL to the hierarchal vision transformer Swin-B and the CNN architecture ConvNeXt-Base. Experimental results are shown in Table 3. Results on Swin-B demonstrate that SNELL-8 outperforms existing reparameterization-based PEFT methods by 0.3% and achieves comparable performance to the state-of-the-art addition-based method SPT-Adapter. For ConvNeXt-Base, SNELL achieves a performance improvement of 0.4% compared to the best-reported result. These results obtained on different architectures further validate the versatility and effectiveness of our SNELL approach.

Memory Usage Comparison. We illustrate the effectiveness of SNELL in terms of memory usage by comparing it with various PEFT methods. Figure 3(a) shows the accuracy and memory usage of different methods on ViT-B/16. Although some methods achieve satisfactory performance, their memory usage is excessively large, even surpassing that of full fine-tuning (*e.g.* SPT-Adapter and VPT-Deep). In comparison, SNELL achieves superior performance on downstream tasks with memory usage comparable to memory-efficient methods, including LoRA and Adapter.

Table 3: Comparisons on VTAB-1k benchmark with supervised pre-trained Swin-B and ConvNeXt-B. Top-1 accuracy (%) is reported. The best result is in **bold**.

| Methods	VTAB-1k Swin-B				VTAB-1k ConvNeXt-B			
	Natural	Specialized	Structured	Mean Acc.	Natural	Specialized	Structured	Mean Acc.
Full	79.1	86.2	59.7	75.0	78.0	83.7	60.4	74.0
Additional-based methods								
MLP-3 [30]	73.6	75.2	35.7	61.5	73.8	81.4	35.7	63.6
VPT-Deep [30]	76.8	84.5	53.4	71.6	78.5	83.0	44.6	68.7
Adapter-8 [26]	81.7	87.3	61.2	76.7	83.1	84.9	64.6	77.5
SPT-Adapter [22]	83.0	87.3	**62.1**	**77.5**	83.7	86.2	65.3	78.4
Reparameterization-based methods								
Linear [30]	73.5	80.8	33.5	62.6	74.5	81.5	34.8	63.6
Partial-1 [30]	73.1	81.7	35.0	63.3	73.8	81.6	39.6	65.0
LoRA-8 [27]	81.7	87.2	60.1	76.3	82.2	84.7	64.1	77.0
SPT-LoRA [22]	83.1	87.4	60.4	77.2	83.4	86.7	**65.9**	78.7
SNELL-8 (**ours**)	**83.3**	**87.7**	61.4	**77.5**	**84.5**	**87.4**	65.6	**79.1**
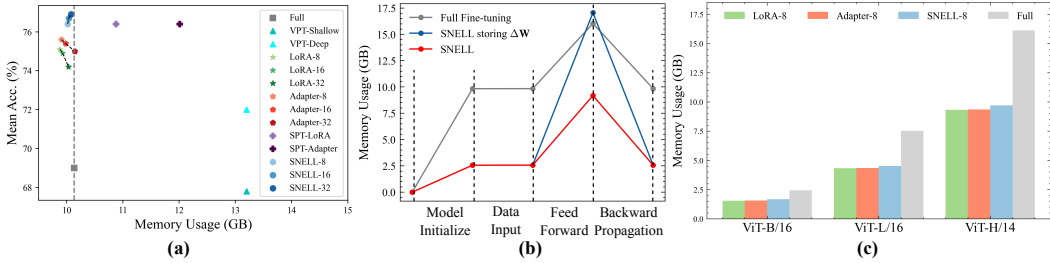
Figure 3: (a) Accuracy *vs.* memory usage (batchsize=64) with supervised pre-trained ViT-B/16 on VTAB-1k. (b) Memory usage evolutions of full fine-tuning, SNELL, and SNELL storing the merged adaptation matrix (SNELL storing $\Delta \mathbf{W}$) on ViT-H/14 during fine-tuning (batchsize=8). (c) Model parameter volumes *vs.* memory usage (batchsize=8). As the model gets larger, SNELL's advantage of low memory usage over full fine-tuning becomes more obvious.

Additionally, we present the memory usage evolutions during the fine-tuning process in Figure 3(b) to provide a detailed explanation of how SNELL can save memory. In the model initialization stage, SNELL exhibits a significantly smaller memory usage compared to full fine-tuning. This is because full fine-tuning stores all weight matrices as learnable parameters in the optimizer, whereas SNELL only stores low-rank matrices with smaller parameter volumes. In the feed-forward phase, the memory usage increases with the storage of intermediate variables for backpropagation. Unlike other intermediate variables, the adaptation matrix $\Delta \mathbf{W}$ in SNELL solely relies on the low-rank parameter matrices, which are already stored in the optimizer. Therefore, it can be dumped in the feed-forward phase and recovered in backpropagation immediately, saving from a large amount of memory usage (SNELL *vs.* SNELL storing $\Delta \mathbf{W}$).

Scaling to Larger Models. To investigate the scalability of SNELL to large models, we apply it to ViT models of varying sizes (ViT-B/16, ViT-L/16, and ViT-H/16 pre-trained on ImageNet21K). We follow the experimental setup presented in Section 4.1, except for modifying the batch size for experiments on ViT-H/14 to 8 and changing the search scope of sparsity ratios $s \in \{0, 0.9\}$.

As depicted in Figure 3(c), the memory usage of full fine-tuning increases rapidly as the model size grows. This observation highlights that existing PEFT methods like VPT and SPT, despite their advanced performances, incur substantial memory costs when applied to large-scale models due to even higher memory usage than full fine-tuning. In contrast, SNELL exhibits a notable advantage in terms of memory usage for larger models (similar to LoRA-8). When applied to ViT-H/14, the memory usage of SNELL is only approximately 50% of that required for full fine-tuning, exemplifying its significant memory-saving capability on large models.

Regarding the performance, as shown in Table 4, SNELL-8 outperforms LoRA-8 on all dataset groups (Natural, Specialized, and Structured) as well as the mean accuracy for both ViT-L and ViT-H on the VTAB-1k benchmark. This demonstrates the effectiveness of SNELL for adapting large pre-trained models to downstream tasks.

8

Table 4: Comparisons on VTAB-1k benchmark with supervised pre-trained ViT-L/16 and ViT-H/16. Top-1 accuracy is reported. The best result is in **bold**.

Methods	VTAB-1k ViT-L/16				VTAB-1k ViT-H/14			
	Natural	Specialized	Structured	Mean Acc.	Natural	Specialized	Structured	Mean Acc.
LoRA-8	81.2	86.6	53.4	73.7	77.9	84.8	55.9	72.9
SNELL-8	82.3	86.9	56.6	**75.3**	79.5	85.1	56.9	**73.8**

Table 5: (a) Performance on VTAB-1k of sparsifying a full-rank matrix, the merged adaptation matrix of LoRA-8 and kernelized LoRA-8 (KLoRA-8) with sparsity ratio $s = 0.9$. (b) The mean accuracy on VTAB-1k of kernelized LoRA (KLoRA) and SNELL (KLoRA+sparsifying) with different ranks of learnable matrices. Perf. Imp. denotes the performance improvement of SNELL over KLoRA.

(a)

Matrix	Natural	Specialized	Structured	Mean Acc.
Full-Rank	80.5	85.1	57.6	74.4
LoRA-8	61.1	81.2	54.7	65.7
KLoRA-8	79.4	84.5	57.9	73.9

(b)

Method	$r = 8$	$r = 16$	$r = 32$
KLoRA	73.2	73.0	72.7
SNELL	74.2	74.4	74.6
Perf. Imp.	+1.0	+1.4	+1.9

4.3 Ablation Studies

Effect of Kernelized LoRA. We explore the effectiveness of kernelized LoRA by comparing the performance of sparsifying a full-rank matrix, the merged adaptation matrix of LoRA, and the merged adaptation matrix of kernelized LoRA. Experimental results are presented in Table 5(a). We can see that sparsifying the merged adaptation matrix of LoRA significantly underperforms a full-rank matrix. This reveals that the low-rank property of the merged adaptation matrix in LoRA greatly compromises the weight selection scope, leading to performance degradation for sparse tuning. However, when we replace LoRA with kernelized LoRA, the performance becomes notably comparable to that of the full-rank matrix under the strong sparsity constraint ($s = 0.9$). This indicates that kernelized LoRA can effectively leverage sparse tuning while maintaining a low memory usage.

Effect of Sparse Tuning. Table 5(b) shows the performance comparison between SNELL and kernelized LoRA to explore the effectiveness of sparse tuning. For kernelized LoRA with different ranks, applying sparse tuning can consistently improve their performance. Moreover, as the rank of the learnable matrix increases, the performance of kernelized LoRA decreases while that of SNELL increases. This difference stems from the model regularization. Similar to sparse regularization, the low-rank property of LoRA that constrains the dependence between individual weights, can also be taken as a form of regularization. As the rank of the learnable matrix increases, the effect of low-rank regularization diminishes. Consequently, kernelized LoRA becomes more susceptible to over-fitting and encounters performance degradation. In contrast, SNELL employs both low-rank and sparse regularization. Higher ranks enable better sparsification towards downstream tasks, boosting sparse regularization that counteracts the diminished low-rank regularization. Therefore, a higher rank may lead to over-fitting in kernelized LoRA, but it can further enhance performance with sparse tuning.

Effect of Different Kernel Function. We investigate the effectiveness of different kernel functions in kernelized LoRA. First, we explore the ability of different kernel functions to fit randomly generated full-rank matrices based on low-rank matrices using the gradient descent algorithm (introduced in Appendix A.3). As shown in Figure 4(a), we explored four kinds of kernel functions. Compared with the linear kernel function, nonlinear kernel functions can reconstruct the full-rank target matrix more accurately based on the low-rank matrices. Subsequently, we explore the performance of kernelized LoRA with different kernel functions for pre-trained model fine-tuning in Figure 4(b). We find that using piecewise linear distance as the kernel function can achieve better results compared to linear kernel function (LoRA), while using Sigmoid and RBF kernels leads to severe performance degradation. This is because the complex non-linear kernel functions such as the exponential function increase the optimization difficulty in deep networks shown in Figure 4(c). More comparisons between LoRA and kernelized LoRA (with piecewise linear kernel) are presented in Appendix C.3.2.

Optimal Sparsity Ratio for Different Downstream Tasks. We provide the optimal sparsity ratio of SNELL-8 on tasks from VTAB-1k benchmarks in Figure 5. The optimal sparsity ratio varies
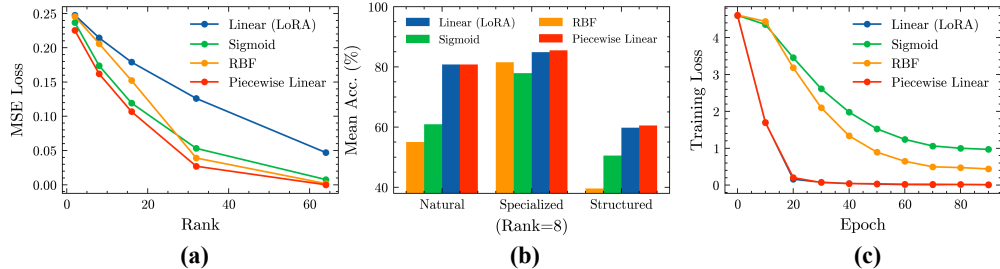
Figure 4: (a) The fitting ability of different kernel functions. We fit random sparse matrices by merging two learnable low-rank matrices with different kernel functions and compute the MSE loss. (b) Performance comparison on groups of datasets in VTAB-1k. (c) Training loss on CIFAR-100 dataset in VTAB-1k benchmark of kernelized LoRA with different kernel functions.
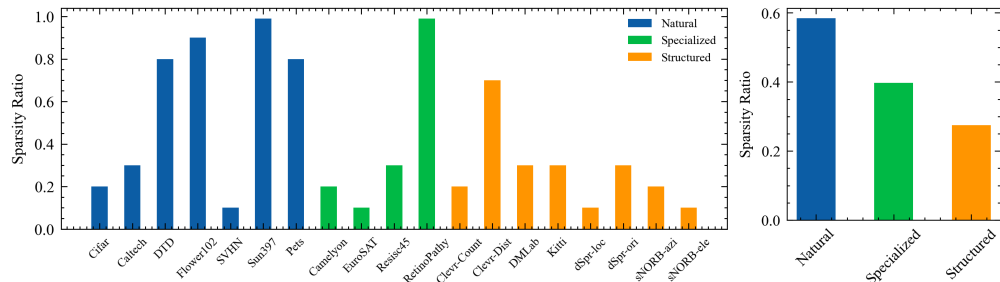

Figure 5: The optimal sparsity ratio of SNELL-8 on different downstream tasks (*left*) and the average optimal sparsity ratio within each group (*right*) in VTAB-1k benchmark. The pre-trained model is the ConvNeXt-B pre-trained on ImageNet-21k.

significantly across different downstream tasks within the same group (*e.g.*, Cifar *vs.* Sun397, dSpr-loc *vs.* Clevr-Dist). Furthermore, we can observe that the Natural task group exhibits a higher average optimal sparsity ratio compared to the Specialized group, while the Structured group demonstrates the lowest ratio. This observation aligns with the example illustrated in Figure A6, where cross-domain adaptation from a model pre-trained on natural images (ImageNet) to images of Specialized and Structured groups necessitates a larger number of tunable parameters.

5 Conclusion

In this work, we proposed a PEFT method named SNELL (**S**parse tuning with ker**NEL**ized **L**oRA) to conduct high-performance sparse tuning with low memory usage. To reduce memory usage, we sparsified the adaptation matrix merged with low-rank matrices rather than the pre-trained weight matrix to reduce the volume of learnable parameters stored in the optimizer. Then we designed a competition-based sparsification mechanism to avoid the additional memory usage of storing the tunable weight indexes. To reveal the effectiveness of sparse tuning, we utilize nonlinear kernel functions to merge the adaptation matrix, increasing the rank of the merged matrix to maintain a compact representation suitable for sparse tuning with low memory usage. Extensive experiments demonstrated the ability of SNELL to leverage the high performance of sparse tuning and the low memory usage of LoRA. For future work, we will apply SNELL on larger models such as LLMs and improve its training efficiency. For limitations discussion, please refer to Appendix E.2.

Acknowledgement

References

[1] Alan Ansell, Edoardo Ponti, Anna Korhonen, and Ivan Vulić. Composable sparse fine-tuning for cross-lingual transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1778–1796, Dublin, Ireland, 2022. Association for Computational Linguistics.

[2] Yutong Bai, Xinyang Geng, Karttikeya Mangalam, Amir Bar, Alan Yuille, Trevor Darrell, Jitendra Malik, and Alexei A Efros. Sequential modeling enables scalable learning for large vision models. *ArXiv preprint*, abs/2312.00785, 2023.

[3] Ankur Bapna and Orhan Firat. Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1538–1548, Hong Kong, China, 2019. Association for Computational Linguistics.

[4] Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert A. Legenstein. Deep rewiring: Training very sparse deep networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[5] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland, 2022. Association for Computational Linguistics.

[6] Alain Berlinet and Christine Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.

[7] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-shot video object segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5320–5329. IEEE Computer Society, 2017.

[8] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *Advances in Neural Information Processing Systems*, 35:16664–16678, 2022.

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 2020.

[10] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 9620–9629. IEEE, 2021.

[11] Zihang Dai, Hanxiao Liu, Quoc V. Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 3965–3977, 2021.

[12] E Dataset. Novel datasets for fine-grained image categorization. In *First Workshop on Fine Grained Visual Categorization, CVPR. Citeseer. Citeseer*. Citeseer, 2011.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.

[14] Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Haitao Zheng, and Maosong Sun. OpenPrompt: An open-source framework for prompt-learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 105–113, Dublin, Ireland, 2022. Association for Computational Linguistics.

[15] Xin Dong, Shangyu Chen, and Sinno Jialin Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4857–4867, 2017.

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[17] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[18] Zihao Fu, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12799–12807, 2023.

[19] Timnit Gebru, Jonathan Krause, Yilun Wang, Duyun Chen, Jia Deng, and Li Fei-Fei. Fine-grained car detection for visual census estimation. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 4502–4508. AAAI Press, 2017.

[20] Demi Guo, Alexander Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4884–4896, Online, 2021. Association for Computational Linguistics.

[21] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: Efficient inference engine on compressed deep neural network. *ACM SIGARCH Computer Architecture News*, 44(3):243–254, 2016.

[22] Haoyu He, Jianfei Cai, Jing Zhang, Dacheng Tao, and Bohan Zhuang. Sensitivity-aware visual parameter-efficient fine-tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11825–11835, 2023.

[23] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 15979–15988. IEEE, 2022.

[24] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 9726–9735. IEEE, 2020.

[25] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge J. Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 595–604. IEEE Computer Society, 2015.

[26] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th*

International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 2019.

[27] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[28] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *ArXiv preprint*, abs/1607.03250, 2016.

[29] Piotr Indyk and Sandeep Silwal. Faster linear algebra for distance matrices. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 35576–35589. Curran Associates, Inc., 2022.

[30] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision*, pages 709–727. Springer, 2022.

[31] Chen Ju, Tengda Han, Kunhao Zheng, Ya Zhang, and Weidi Xie. Prompting visual-language models for efficient video understanding. In *European Conference on Computer Vision*, pages 105–124. Springer, 2022.

[32] Konstantinos Koutroumbas and Sergios Theodoridis. *Pattern recognition*. Academic Press, 2008.

[33] Wei-Hong Li, Xialei Liu, and Hakan Bilen. Cross-domain few-shot learning with task-specific adapters. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 7151–7160. IEEE, 2022.

[34] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online, 2021. Association for Computational Linguistics.

[35] Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Scaling & shifting your features: A new baseline for efficient model tuning. *Advances in Neural Information Processing Systems*, 35:109–123, 2022.

[36] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. *ArXiv preprint*, abs/2402.09353, 2024.

[37] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *ArXiv preprint*, abs/2110.07602, 2021.

[38] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 9992–10002. IEEE, 2021.

[39] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 11966–11976. IEEE, 2022.

[40] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. 2018.

[41] Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[42] Dmitry Molchanov, Arsenii Ashukha, and Dmitry P. Vetrov. Variational dropout sparsifies deep neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 2498–2507. PMLR, 2017.

[43] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 11264–11272. Computer Vision Foundation / IEEE, 2019.

[44] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pages 722–729. IEEE, 2008.

[45] Zhengqi Pei and Shuhui Wang. Dynamics-inspired neuromorphic visual representation learning. In *International Conference on Machine Learning*, pages 27521–27541. PMLR, 2023.

[46] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online, 2021. Association for Computational Linguistics.

[47] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[48] Suraj Srinivas and R. Venkatesh Babu. Data-free parameter pruning for deep neural networks. In Xianghua Xie, Mark W. Jones, and Gary K. L. Tam, editors, *Proceedings of the British Machine Vision Conference 2015, BMVC 2015, Swansea, UK, September 7-10, 2015*, pages 31.1–31.12. BMVA Press, 2015.

[49] Xue-Lian Sun, Zhen-Hua Chen, Xize Guo, Jingjing Wang, Mengmeng Ge, Samuel Zheng Hao Wong, Ting Wang, Si Li, Mingze Yao, Laura A Johnston, et al. Stem cell competition driven by the axin2-p53 axis controls brain size during murine development. *Developmental Cell*, 58(9):744–759, 2023.

[50] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5227–5237, 2022.

[51] Johan AK Suykens and Joos Vandewalle. Chaos control using least-squares support vector machines. *International journal of circuit theory and applications*, 27(6):605–615, 1999.

[52] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *ArXiv preprint*, abs/2307.09288, 2023.

[53] Cheng-Hao Tu, Zheda Mai, and Wei-Lun Chao. Visual query tuning: Towards effective usage of intermediate representations for parameter and memory efficient transfer learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7725–7735, 2023.

[54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[55] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[56] Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. Raise a child in large language model: Towards effective and generalizable fine-tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9514–9528, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics.

[57] Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6071–6079. IEEE Computer Society, 2017.

[58] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12104–12113, 2022.

[59] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *ArXiv preprint*, abs/1910.04867, 2019.

[60] Qizhe Zhang, Bocheng Zou, Ruichuan An, Jiaming Liu, and Shanghang Zhang. Mosa: Mixture of sparse adapters for visual efficient tuning. *ArXiv preprint*, abs/2312.02923, 2023.

[61] Renrui Zhang, Rongyao Fang, Wei Zhang, Peng Gao, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. Tip-adapter: Training-free clip-adapter for better vision-language modeling. *ArXiv preprint*, abs/2111.03930, 2021.

[62] Yuanhan Zhang, Kaiyang Zhou, and Ziwei Liu. Neural prompt search. *ArXiv preprint*, abs/2206.04673, 2022.

[63] Zhi Zhang, Qizhe Zhang, Zijun Gao, Renrui Zhang, Ekaterina Shutova, Shiji Zhou, and Shanghang Zhang. Gradient-based parameter selection for efficient fine-tuning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 28566–28577. IEEE, 2024.

[64] Mengjie Zhao, Tao Lin, Fei Mi, Martin Jaggi, and Hinrich Schütze. Masking as an efficient alternative to finetuning for pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2226–2241, Online, 2020. Association for Computational Linguistics.
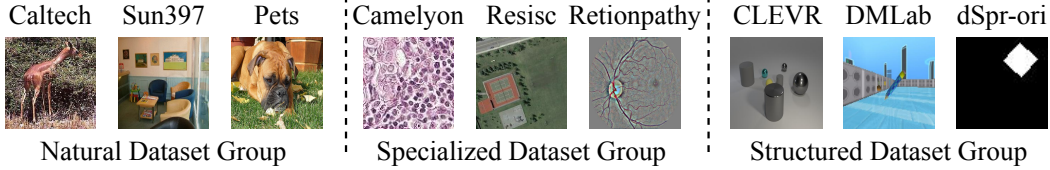
Figure A6: Samples of different downstream tasks in VTAB-1k.

A More Details of Experimental Setup

A.1 More Details of Contenders.

- Full: fully tunes all the model parameters (including backbone and classification head).
- Linear: freezes all the backbone parameters and only tunes the linear classification head.
- Bias [5]: freezes all model parameters except for the bias term and the linear classification head.
- Partial-1: freezes the backbone except for the last 1 layer and also tunes the classification head as described in [30].
- MLP-3: freezes the backbone and tunes the classification head implemented by a trainable 3-layer multi-layer perceptron as described in [30].
- VPT-Shallow [30]: freezes all the backbone parameters while introducing additional trainable prompts to the input space of the pretrained ViT.
- VPT-Deep [30]: freezes the backbone while appending additional trainable prompts to the sequence in the multi-head self-attention layer of each ViT block.
- Adapter-r [26]: freezes all the backbone parameters while adding a down projection, a ReLU non-linearity, and an up projection layer sequentially in the feed-forward network (FFN) of each visual Transformer block. We report the performance implemented by [22] for comparison.
- Lora-r [27]: freezes all the backbone parameters while adding a concurrent branch including two low-rank matrices to the weight matrices in the multi-head self-attention layers to approximate efficiently updating them. The low-rank matrices can be merged into the backbone weights after fine-tuning. We report the performance implemented by [22] for comparison.
- SPT [22]: identifies the tunable parameters for a given task in a data-dependent way, and utilizes LoRA (SPT-LoRA) or Adapter (SPT-Adapter) for weight matrices with a large number of tunable parameters and sparse tuning for weight matrices with a small number of tunable parameters.
- VQT [53]: introduces a handful of learnable query tokens to each layer for adaptation.
- DoRA [36]: decomposes the pre-trained weight into two components, *i.e.*, magnitude and direction, for fine-tuning. It specifically employs LoRA for directional updates to efficiently minimize the number of trainable parameters.
- GPS [63]: identifies task-dependent tunable weights and applies sparse tuning to these weights.

A.2 Dataset Samples for the Downstream Tasks

We visualize some sampled images from different downstream tasks of VTAB-1k [59]) in Figure A6. The VTAB-1k benchmark encompasses a diverse range of tasks, including natural images, remote sensing, medical images, etc. Notably, our SNELL has achieved state-of-the-art (SOTA) performance on these datasets, demonstrating its general effectiveness.

A.3 Implementation details of Figure 4(a)

Given a random matrix $\mathbf{W}^{(gt)} \in \mathbb{R}^{m \times n}$, we fit this matrix by merging two low-rank learnable matrices $\mathbf{B} \in \mathbb{R}^{m \times r}, \mathbf{A} \in \mathbb{R}^{n \times r}$ with different kernel functions κ,

$$\min_{\mathbf{A}, \mathbf{B}} \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (\mathbf{W}_{ij}^{(gt)} - \kappa(\mathbf{B}_{i,\cdot}, \mathbf{A}_{j,\cdot}))^2. \tag{A7}$$

Table A6: Expression of kernel function utilized in the main text.

Kernel Function	Expression
Linear	$\kappa(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$
Piecewise Linear	$\kappa(\mathbf{x}, \mathbf{x}') = \sum_{p=1}^{P} \alpha_p \|\mathbf{x}_{\lceil rp/P \rceil : \lceil r(p+1)/P \rceil} - \mathbf{x}'_{\lceil rp/P \rceil : \lceil r(p+1)/P \rceil}\|_2$
Sigmoid	$\kappa(\mathbf{x}, \mathbf{x}') = \alpha(1 + exp(-\beta\mathbf{x}^\top\mathbf{x}'))^{-1} + \gamma$
RBF	$\kappa(\mathbf{x}, \mathbf{x}') = \alpha(exp(-\beta\|\mathbf{x} - \mathbf{x}'\|_2^2)) + \gamma$

Table A7: Top-1 accuracy (%) on VTAB-1k benchmarks using ViT-B/16 backbone pre-trained on ImageNet-21k supervisedly. The best result is in **bold**, and the second-best result is underlined.

Methods	Natural							Specialized				Structured								-
	Cifar100	Caltech101	DTD	Flower102	SVHN	Sun397	Pets	Camelyon	EuroSAT	Resisc45	Retinopathy	Clevr-Count	Clevr-Dist	DMLab	KITTI-Dist	dSpr-Loc	dSpr-Ori	sNORB-Azim	sNORB-Ele	Mean Acc.
Full	68.9	87.7	64.3	97.2	87.4	38.8	86.9	79.7	95.7	84.2	73.9	56.3	58.6	41.7	65.5	57.5	46.7	25.7	29.1	65.6
Additional-based methods																				
MLP-3	63.8	84.7	62.3	97.4	32.5	49.2	84.7	77.0	88.0	70.2	56.1	47.8	32.8	32.3	58.1	12.9	21.2	15.2	24.8	53.2
VPT-Shallow	77.7	86.9	62.6	97.5	74.5	51.2	87.3	78.2	92.0	75.6	72.9	50.5	58.6	40.5	67.1	68.7	36.1	20.2	34.1	64.9
VPT-Deep	**78.8**	90.8	65.8	98.0	78.1	49.6	88.3	81.8	96.1	83.4	68.4	68.5	60.0	46.5	72.8	73.6	47.9	32.9	37.8	69.4
Adapter-8	69.2	90.1	68.0	98.8	82.8	54.3	89.9	84.0	94.9	81.9	75.5	80.9	65.3	48.6	78.3	74.8	48.5	29.9	41.6	71.4
Adapter-32	68.7	92.2	69.8	98.9	84.2	53.0	90.3	83.2	95.4	83.2	74.3	81.9	63.9	48.7	80.6	76.2	47.6	30.8	36.4	71.5
NOAH	69.6	92.7	70.2	99.1	86.1	53.7	90.4	84.4	95.4	83.9	75.8	82.8	**68.9**	49.9	81.7	81.8	48.3	32.8	**44.2**	73.2
SPT-Adapter	72.9	93.2	72.5	**99.3**	88.8	55.8	91.4	**86.2**	96.1	85.5	75.5	83.0	68.0	51.9	81.2	82.4	**51.9**	31.7	41.2	74.1
Reparameterized-based methods																				
Linear	63.4	85.0	63.2	97.0	36.6	51.0	86.3	78.5	87.5	68.6	74.0	34.3	30.6	33.2	55.4	12.5	20.0	9.6	19.2	52.9
Partial-1	66.8	85.9	62.5	97.3	37.6	50.6	85.5	78.6	89.8	72.5	73.3	41.5	34.3	33.9	61.0	31.3	32.8	16.3	22.4	56.5
Bias	72.8	87.0	59.2	97.5	59.9	51.4	85.3	78.7	91.6	72.9	69.8	61.5	55.6	32.4	55.9	66.6	40.0	15.7	25.1	62.0
LoRA-8	67.1	91.4	69.4	98.8	85.3	54.0	90.4	84.9	95.3	84.4	73.6	82.9	69.2	49.8	78.5	75.7	47.1	31.0	44.0	72.3
LoRA-16	68.1	91.4	69.8	99.0	86.4	53.1	90.5	85.1	95.8	84.7	74.2	83.0	66.9	50.4	81.4	80.2	46.6	32.2	41.1	72.6
SPT-LoRA	73.5	93.3	72.5	**99.3**	87.9	55.5	**91.5**	85.7	**96.2**	85.9	75.9	**84.4**	67.6	52.5	**82.0**	81.0	51.1	30.2	41.3	74.1
SNELL-8	73.7	92.7	72.4	99.2	89.2	55.4	91.4	84.9	96.1	86.4	75.2	84.0	68.5	**53.5**	81.0	82.7	49.9	33.9	39.2	74.2
SNELL-16	74.2	**93.4**	72.5	**99.3**	90.2	55.7	91.4	85.7	95.8	86.5	**76.3**	**84.4**	68.2	53.0	**82.0**	82.2	49.6	33.3	40.6	74.4
SNELL-32	74.5	**93.4**	73.1	**99.3**	91.1	55.9	**91.5**	85.5	96.1	86.5	76.2	83.4	68.6	52.2	81.3	**83.2**	50.7	35.9	39.0	**74.6**

We use gradient descent for $1e5$ optimization steps, employing the Adam optimizer with a learning rate of $1e-4$. We fit 10 randomly generated matrices for each kernel function presented in Table A6 and report the average MSE Loss in Figure 4(a).

B Introduction of Utilized Kernel Functions

Kernel Function Definition (positive semi-definite). Consider a vector space $\mathbb{R}^r$, a kernel function $\kappa : \mathbb{R}^r \times \mathbb{R}^r \to \mathbb{R}$ is called a positive semi-definite kernel on $\mathbb{R}^r$ if

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \geq 0 \tag{A8}$$

holds for all $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathbb{R}^r, c_1, ..., c_n \in \mathbb{R}, n \in \mathbb{N}$.

Given two vectors $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^r$, we show the utilized kernel functions in Table A6. We introduce additional learnable parameters (the *e.g.* α for Sigmoid and RBF kernel, α_p for piecewise linear kernel) that enable the merged adaptation matrix $\Delta\mathbf{W}$ to accommodate both positive and negative values. The additional parameters select certain elements in the matrix and assign them negative values, without compromising the high-rank property of the merged adaptation matrix $\Delta\mathbf{W}$. We set $P = 2$ for the piecewise linear kernel.

Table A8: (a) Performance comparisons on FGVC between SNELL and GPS. (b) Memory usage comparison between SNELL and GPS (batchsize=8) on different pre-trained models.

(a)

Method	CUB	NABirds	Oxford Flowers	Stanford Dogs	Stanford Cars	Mean
GPS	**89.9**	86.7	**99.7**	**92.2**	90.4	**91.8**
SNELL-32	**89.9**	**87.0**	99.4	92.0	**90.5**	**91.8**

(b)

Method	Memory Usage (Mb)		
	ViT-B	ViT-L	ViT-H
GPS	2428	7522	16119
SNELL-32	**1673**	**4519**	**9692**

Table A9: Performance comparisons on VTAB-1k between SNELL and VQT with ViT-B/16 pre-trained on ImageNet-21K. The best result is in **bold**.

Method	Natural	Specialized	Structured	Mean Acc.
VQT	72.7	84.5	49.3	68.8
SNELL-8	82.0	85.7	61.6	76.4
SNELL-16	82.4	**86.1**	61.7	76.7
SNELL-32	**82.7**	**86.1**	**61.8**	**76.9**

C   Additional Experiments

C.1   More Comparisons with Existing Methods

Given that some methods do not provide performance or implementation details on both FGVC and VTAB benchmarks, we present a comparison between SNELL and these methods in the appendix rather than in Table 1. First, we provide comparisons with GPS [63] on the FGVC benchmark in terms of performance and memory usage in Table A8. With comparable performance, SNELL has a significant advantage over GPS in terms of memory usage. Then, we compare SNELL with VQT [53] on VTAB-1k dataset in Table A9. SNELL significantly outperforms VQT (76.9% *vs.* 68.8%).

C.2   Per-task results on the VTAB-1k benchmark

We provide the per-tasks results on the VTAB-1k benchmark using ViT-B/16 supervised pre-trained on ImageNet21K in Table A7. Our SNELL has demonstrated superior performance by achieving SOTA performance on 13 downstream tasks. Additionally, SNELL achieves SOTA performance on the mean accuracy across all tasks (74.6% *vs.* 74.1%), indicating its effectiveness in various domains.

C.3   More ablation studies.

C.3.1   Comparison between Competition-based Sparsification and Pre-defined Weight Mask

To verify the effectiveness of the proposed competition-based sparsification mechanism, we compare the performance on FGVC datasets between kernelized LoRA (KLoRA-8-Fixed) with pre-defined fixed masks, and SNELL in Table A10. The weight masks are generated by SPT [22]. For a fair comparison, we utilize the same data augmentation as SPT. Compared to our dynamic masking strategy, pre-defined fixed masking can hardly identify and adjust the most task-relevant weights in an end-to-end fashion, which leads to performance degradation (89.4 vs. 90.3).

Table A10: Performance comparisons on FGVC benchmark between kernelized LoRA with fixed weight masks (KLoRA-8-Fixed) and our dynamical masks (SNELL-8).

Method	CUB-200	NABirds	Oxford Flowers	Stanford Dogs	Stanford Cars	Mean
KLoRA-8-Fixed	88.0	82.1	99.0	89.4	88.4	89.4
SNELL-8	89.0	83.9	99.3	90.6	88.6	**90.3**

Table A11: Comparisons between LoRA and kernelized LoRA (KLoRA) on VTAB-1k using ViT-B/16 pre-trained on ImageNet21k supervisedly. Better performance for the same rank is in **bold**.

Method	Natural	Specialized	Structured	Mean Acc.
LoRA-8	**80.8**	84.9	59.6	75.1
KLoRA-8	**80.8**	**85.5**	**60.5**	**75.6**
LoRA-16	80.6	85.6	58.5	74.9
KLoRA-16	**80.9**	**85.7**	**59.7**	**75.4**
LoRA-32	79.4	**85.4**	57.8	74.2
KLoRA-32	**80.8**	**85.4**	**59.4**	**75.2**

Table A12: Memory usage comparison between SNELL and LoRA. Δ Mem. denotes the incremental memory usage of SNELL in comparison to LoRA.

Pre-trained Model	LoRA-8 Mem. (MB)	SNELL-8 Mem. (MB)	Δ Mem. / SNELL-8 Mem.
ViT-B/16	1546	1673	0.076
ViT-L/16	4325	4519	0.043
ViT-H/16	9325	9692	0.038

C.3.2 Comparison between Kernelized LoRA and LoRA

We compare the performance of LoRA and kernelized LoRA on the VTAB-1k benchmark, where all weight matrices of the pre-trained models are fine-tuned to ensure a fair comparison. The experimental results are presented in Table A11. Through experiments with different ranks, we observed that kernelized consistently outperforms LoRA across various task groups. The replacement of the inner product with nonlinear kernel functions leads to stronger expressive ability, which in turn contributes to improved performance on downstream tasks.

C.3.3 Additional Memory Usage from Nonlinear Kernel Functions

In Figure 3(a), we observe that SNELL requires additional memory usage compared to LoRA due to the incorporation of nonlinear kernel functions. To explore whether the impact of this additional usage hinders the usability of SNELL on large models, we compare the memory usage between SNELL and LoRA on models as the model size grows (in Table A12). As the model size expands, the incremental memory usage of SNELL becomes negligible.

C.4 Experiments on Large Language Models

We apply SNELL on LLaMA2-7B [52] to adapt to the commonsense reasoning benchmark. As Table A13 shows, SNELL achieves a better performance than LoRA. This shows the applicability of SNELL to NLP tasks. Many other vision PEFT approaches lack this capability, as they necessitate a full level of memory usage for fine-tuning as Figure 3(a) shows.

C.5 Training Time Analysis

Table A14 provides a comparison of training time costs between SNELL and other PEFT methods using NVIDIA GeForce RTX 4090 GPU. The training time of SNELL-8 is slightly higher than LoRA-8 (0.557 *vs.* 0.443). By further comparing the training time of SNELL-8 and SNELL-8 (saving $\Delta \mathbf{W}$), it becomes apparent that the increase in time cost primarily stems from the recomputation

Table A13: Performance on commonsense reasoning benchmark with LLaMA-2-7B.

Model	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Average
LoRA-32	69.8	79.9	79.5	83.6	82.6	79.8	64.7	81.0	77.6
SNELL-32	71.4	82.9	80.7	82.1	80.9	82.6	68.0	80.8	**78.7**

Table A14: Training time cost on ViT-B/16 of different PEFT methods.

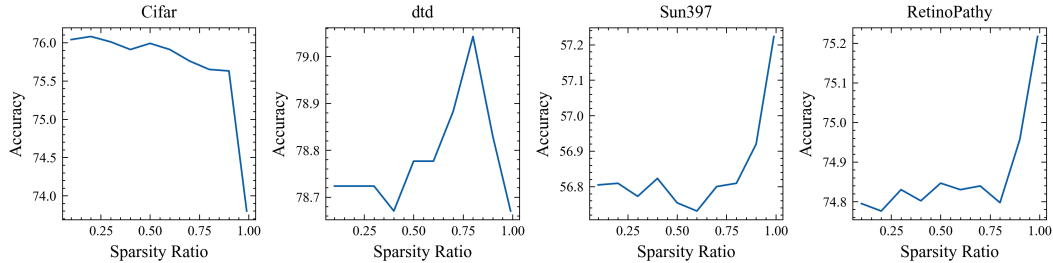| Method	LoRA-8	KLoRA-8	KLoRA-8 (saveing $\Delta \mathbf{W}$)	SNELL-8	SNELL-8 (saving $\Delta \mathbf{W}$)
Training time (s/img)	0.443	0.522	0.446	0.557	0.448

Figure A7: Accuracy of different sparsity ratios using SNELL-8. The pre-trained model is the ConvNeXt-B pre-trained on ImageNet-21k.

of the merged adaptation matrix $\Delta \mathbf{W}$. Conversely, the time cost associated with sparsification (KLoRA-8 *vs.* SNELL-8) and kernelization (LoRA-8 *vs.* KLoRA-8 (saving $\Delta \mathbf{W}$)) is relatively small. Indeed, despite the slight increase in time cost due to the recomputation, one significant advantage is the significant performance improvement and memory efficiency shown in Figure 3.

D   Additional Visualization

D.1   Performance of Different Sparsity Ratio

Figure A7 depicts the accuracy of different sparsity ratios on datasets in VTAB-1k. Different downstream tasks exhibit diverse preferences for the sparsity ratio. For instance, CIFAR-100 tends to favor a smaller sparsity ratio (0.2), while DTD prefers a larger sparsity ratio (0.8). Both Sun397 and Retinopathy tasks also lean towards a larger sparsity ratio (0.99). This highlights the need to consider the specific characteristics of each task when determining the optimal sparsity ratio.

D.2   Analysis of Tunable Parameters

We analyze the tunable weights of SNELL for different downstream tasks. In Figure A8, we compute the number of weights in the weight matrix $\mathbf{W}_Q$ of self-attention [54] selected by multiple tasks. We find that most of the weights are only selected by a single downstream task (Tuned Times=1). Moreover, we find that in blocks of different depths, there will be a small part of weights that are selected by multiple downstream tasks, which indicates that there exists a small number of crucial parameters to improve the model's performance on downstream tasks.
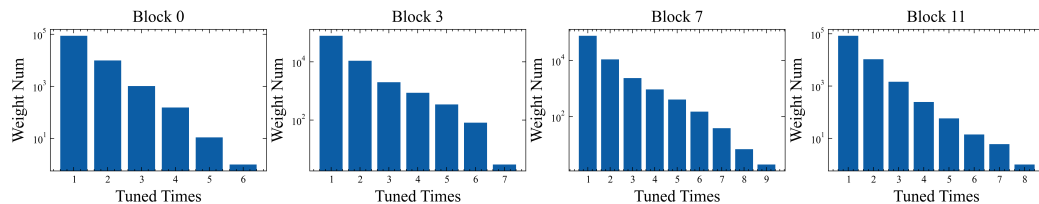

Figure A8: Number of weights in $\mathbf{W}_Q$ selected by multiple tasks with sparsity ratio $s = 0.99$.

E Discussion

E.1 Tunable Parameter Volume Computing

We justify our choice to omit to report the volume of learnable parameters. First, computing the volume of tunable parameters in SNELL is difficult. In the case of LoRA, the volume corresponds to the shape of the learnable low-rank matrices. Conversely, for sparse tuning, the volume is determined by the number of updated weights. However, SNELL employs low-rank matrices as learnable parameters and achieves additional updated weight reduction by sparsifying the merged matrices. When using the parameter volume computation method of LoRA, calculating the reduction in parameters due to sparsification becomes challenging. Conversely, applying the parameter volume computation method of sparse tuning would be inherently unfair, given that SNELL is specifically optimized using low-rank matrices. Second, the parameter efficiency is a pathway to achieve high performance and low memory usage rather than an objective for model improvement, because performance improvement and memory usage reduction hold practical value. In our experiments, SNELL has demonstrated its advantages in terms of high performance and low memory usage, which we consider more valuable than the pursuit of fewer learnable parameters.

E.2 Limitation Discussion

Despite achieving state-of-the-art performance with low memory usage, SNELL requires more training time than LoRA. The additional training time cost comes from the recomputation of the merged matrix $\Delta \mathbf{W}$ in the backpropagation process presented in Appendix C.5. However, it is crucial to note that this limitation can be solved. Firstly, given the unique characteristics of the kernel matrix, more efficient methods [29] can be employed to calculate the merged adaptation matrix $\Delta \mathbf{W}$. Secondly, by designing appropriate GPU operators, it is possible to avoid explicitly calculating $\Delta \mathbf{W}$ [45] during the fine-tuning process like LoRA and reparameterize the learnable low-rank matrices into the pre-trained weight matrices after the fine-tuning process.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitation in Appendix E.2

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: This paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Please refer to Section 4.1, Appendix A and the released codes.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Please refer to the released codes.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Please refer to Section 4.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: This paper does not report error bars following the practice of previous studies [30, 22, 53].

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please refer to Figure 3 and Table A14.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: This paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: This paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Please refer to Section 4.1 and Appendix A.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Please refer to the released codes.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.