Appendix

A. Heart-LoRA with More Granularity

As mentioned earlier, the original Heart-LoRA uses layer-by-layer accumulation to calculate responsiveness. This approach ensures that the responsiveness of each head is considered globally. However, directly calculating responsiveness at each layer and applying deactivation is also worth exploring. Therefore, we propose a variant of Heart-LoRA that directly calculate responsiveness at each layer and then perform the deactivation at layer level. We term this variant as Heart-LoRA† and present its results in Table 3.

From the results, it can be seen that the layer-by-layer implementation of Heart-LoRA† also performs well, outperforming LoRA by 0.6%. We argue that this is due to our proposed responsiveness calculation algorithm, where the responsiveness of the heads acts on the local layer, which could potentially yield better results in certain tasks. Overall, both Heart-LoRA† and the original Heart-LoRA significantly outperform LoRA and generally require fewer parameters to function.

B. On Hierarchical Transformers

We also apply our method on Swin Transformer [18], a widely adopted and representative design of hierarchical structures of transformers aiming to improve ViT's computational efficiency.

Given that the number of heads and dimension of token is different across layers, simply setting a ne hyper-parameter is not enough. Instead, we set a hyper-parameter $ratio$ indicating the percentage of heads to deactivate in the window attention. The accumulation of responsiveness R and thus the candidate C for swin transformer is restricted in layers with the same number of attention heads in this case, which makes sense since layers with the same number of attention heads generally are in the same hierarchy of Swin Transformer.

We use Swin-B as our backbone following [15]. It consists of four stages, with depths $\{2, 2, 18, 2\}$ and number of heads $\{4, 8, 16, 32\}$. In this case, take $ratio = 0.25$ as an example, we set the number of candidate of deactivation C for the four stages separately as $\{1, 2, 4, 8\}$. While in many tasks, the deeper the layer is, it learns more task specific information, setting a fixed number of heads also proves to be effective since it discard little information in deeper layer while discarding much information in shallower layer where more redundancy persists.

The results are shown in Table 4, using **swin-B** pre-trained on ImageNet-21K [4] as backbone for Heart-LoRA. Following [15] [14], we compare Heart-LoRA with several methods which could also be applied on swin transformer: Full, Linear, VPT [11] and Bi-LoRA [15]. From the results, we observe that Heart-LoRA achieve high performance in Swin Transformer even with only very few hyper-parameter settings, showcasing its versatility across different transformer designs. While the performance may be sub-optimal compared to ViT, there is potential for further improvement as our experiments were conducted in only a limited number of cases. Additionally, our method on Swin-B also demonstrates that Heart-LoRA is structure-agnostic and method-agnostic, indicating its potential to enhance performance across various architectures.

Method	Average	Natural	Specialized	Structured
Full	68.9	75.9	83.4	47.6
LoRA	76.4	81.5	85.2	62.6
Heart-LoRA†	**76.8** (↑ 0.4)	**81.7** (↑ 0.2)	**85.8** (↑ 0.6)	**63.0** (↑ 0.4)

Table 3. Results on VTAB-1K of Heart-LoRA†, using a layer by layer deployment of the calculation in section 3.

Method	Average	Natural	Specialized	Structured
Full	74.99	79.2	86.2	59.7
Linear	62.60	73.5	80.8	33.5
VPT-Deep	71.55	76.8	84.5	53.4
Bi-LoRA	76.67	82.0	86.8	61.2
Heart-LoRA	**76.93** (↑ 0.26)	**82.3** (↑ 0.3)	**87.1** (↑ 0.3)	**61.4** (↑ 0.2)

Table 4. Results on VTAB-1K using Swin-B as backbone, reported accuracy from average, natural, specialized and structured.

C. Full Ablation on Number of Heads

We provide a comprehensive analysis of the impact of ne settings on performance. We selected 16 datasets, which essentially encompass all types of tasks in the VTAB-1K benchmark. For each dataset, we varied the value of ne and tracked the corresponding performance. Since we are using ViT-B, which has 12 heads, the values for ne were set at $1, 3, 5, 7, 9, 11$ to provide more data points. The results are in Fig 8.

The results show a general trend where increasing ne leads to a decrease in performance. However, the degree of this decrease is not linear; in many datasets, we observed that the performance drop after increasing ne is not significant (like in Cifar100 and dSpr-Ori), and in some cases the performance even improves (EuroSAT and SVHN). We argue that this phenomenon is due to the inherent redundancy among all the heads in the multi-head self-attention mechanism, where some heads have minimal responsiveness (defined in section 1) and thus contribute little to the current task. Since Heart-LoRA can accurately identify these minimally responsive heads and deactivate them accordingly, the performance does not drop significantly, and in many datasets, it even improves.

However, when ne is large, such as $ne = 9$ or $ne = 11$,

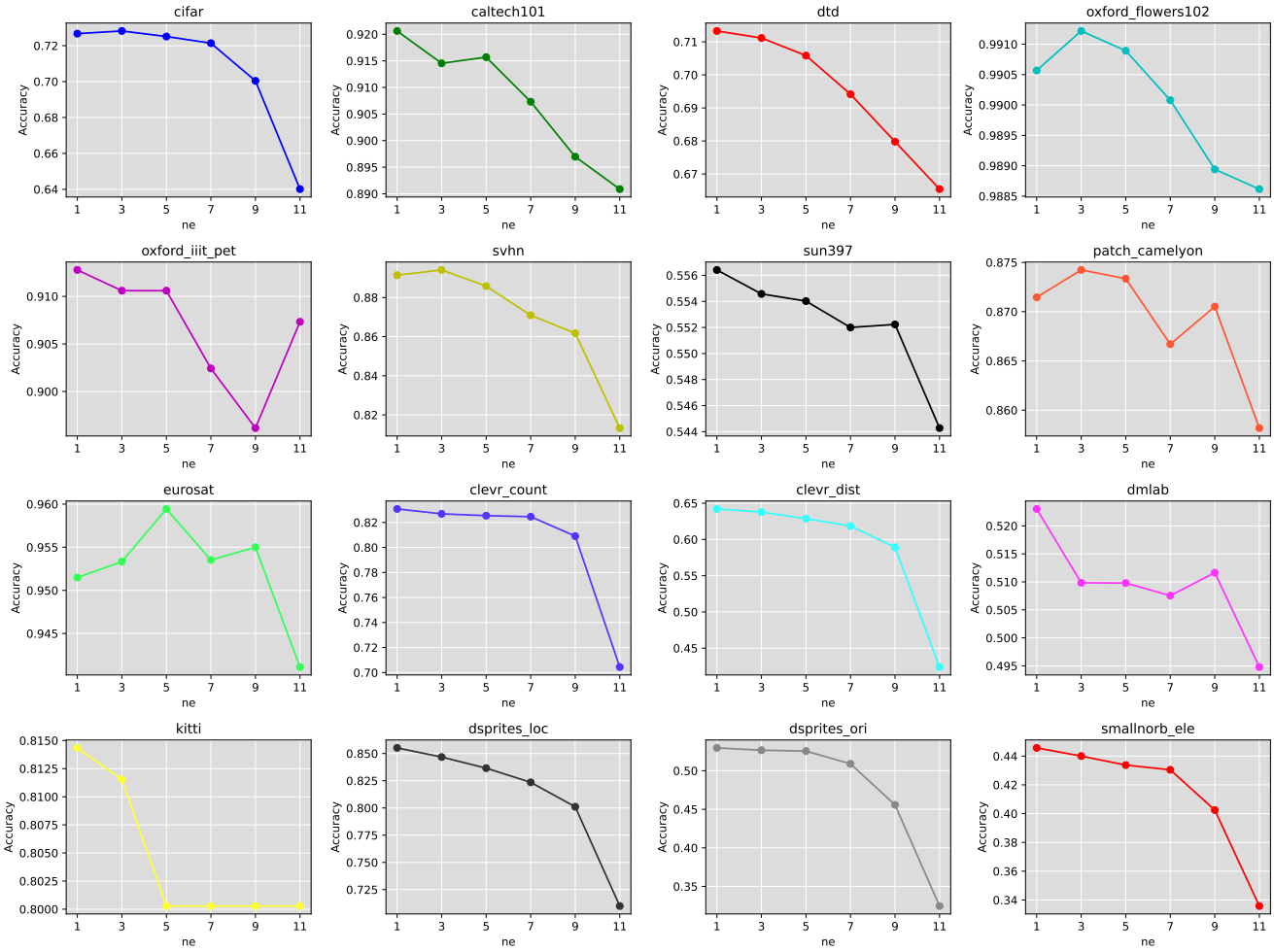

Figure 8. Full ablation results on the effect of ne on different datasets in VTAB-1K benchmark.

more than half of the heads are deactivated. At this point, performance generally experiences significant degradation. This is because deactivating too many heads also deactivates those with high responsiveness, thus discarding their contributions.

Finally, we can see that the ne value that yields the highest performance varies across different datasets, which validates that the degree of redundancy varies with the task, and thus the pattern of heads that Heart-LoRA should identify and deactivate is distinct across tasks. This confirms the ability of Heart-LoRA to uniquely discover task-specific patterns for different tasks.

D. Other Ways to Obtain Responsiveness

Although we utilized Taylor expansion in the section 3 to calculate responsiveness, it is feasible to use other simpler and more straightforward methods as well. Here, we present the results of calculating responsiveness using two criteria: the L2 norm of weight with one parameter or the L2 norm of gradient, as shown in Table 5.

From the results, we can observe that using Heart-LoRA†, we still manage to surpass LoRA and match Bi-LoRA. Considering this was achieved with a search between ne values of only 1 and 3, it is quite impressive. This outcome stems from the unique adaptation process of Heart-LoRA, which identify the redundant heads and deactivate them during fine-tuning. However, it is noted that this re-

Method	Average	Natural	Specialized	Structured
Full	68.9	75.9	83.4	47.6
LoRA	76.4	81.5	85.2	62.6
Heart-LoRA† (weight)	76.7	81.6	85.8	62.5
Heart-LoRA† (grad)	76.7	81.6	85.5	63.0

Table 5. Results on VTAB-1K of Heart-LoRA†, using L2 norm of weight (denoted as weight) or L2 norm of gradient (denoted as grad) to calculate the responsiveness.

sult does not outperform the effectiveness of Heart-LoRA† using Taylor expansion, which underscores the superiority of approach from the section 3. By considering both weight and gradient simultaneously, it allows for the identification of heads with truly minimal responsiveness, thereby reducing the risk of misidentification.

E. Comparison to Arbitrary Deactivation

We compare the results of the arbitrary deactivation approach adopted in section 4.3.1 with Heart-LoRA. The results are presented in Table 6. As we can see, although the arbitrary mask performs quite well, exceeding LoRA by roughly 0.5%, it still underperforms Heart-LoRA by 0.3%, which is notable. This indicates that the performance gain of Heart-LoRA is not only due to the structural advantage of head deactivation but also stems from the unique advantage of using responsiveness to cleverly identify unimportant heads.

While there is improvement when utilizing arbitrary deactivation, we did not utilize the property of head redundancy to the extreme as we were arbitrarily selecting heads to deactivate and ignore the fact that *head-level responsiveness varies across tasks*, as discussed in section 4.3.2. One option to maximize the performance is to study the attention patterns for each downstream task manually. However, manually studying the patterns is not feasible when dealing with a significant amount of downstream tasks for it's huge computational cost. This leads us to seek methods capable of efficiently identifying deactivation candidates in the study of PETL without the need for manual investigation, and this brings us to our proposed Heart-LoRA which utilize task-specific responsiveness to cleverly recognize unimportant heads to deactivate.

Method	Average	Natural	Specialized	Structured
Full	68.9	75.9	83.4	47.6
LoRA	76.4	81.5	85.2	62.6
Front	76.9	81.7	85.8	63.3
Heart-LoRA	77.2 (↑ 0.3)	81.8 (↑ 0.1)	86.2 (↑ 0.4)	63.5 (↑ 0.2)

Table 6. Comparison between using arbitrary deactivation and the original Heart-LoRA.

F. Experimental details

F.1. Actual Implementation

The core of Heart-LoRA lies in obtaining a task-specific mask, with the actual code implementation in Alg. 1:

F.2. Backbones

F.3. Implementation Environment

On a single NVIDIA RTX 4090 GPU with OS being Ubuntu 22.04.3 LTS x86_64, we conduct all experiments using *PyTorch* and *timm* library.

Algorithm 1: Implementation of obtaining the mask using PyTorch. The obtained mask is then used to perform the deativation of heads.

```
def get_scores(args, adapter, num_of_heads=12):
    weight = get_weight(adapter)
    grad = get_grad(adapter)
    step = weight.shape[1] // 12
    all_weights = [weight[:, i:i + step] for i in range(num_of_heads)]
    all_grads = [grad[:, i:i + step] for i in range(num_of_heads)]
    scores = [get_score(all_weights[i], all_grads[i]) for i in range(num_of_heads)]
    return scores

def get_mask(args, adapter, mask, ne):
    scores = get_scores(args, adapter, 12)
    scores_tensor = torch.tensor(scores)
    _, indices = torch.topk(scores_tensor, ne, largest=False)
    indices_list = indices.tolist()
    for i in indices_list:
        mask[:, :, i, :, :] = 0
    return mask
```

Model	Pre-Training Dataset	Size (M)	Pre-Trained Weights
ViT-B/16 [5]	ImageNet-21K	85.8	checkpoint
Swin-B [18]	ImageNet-21K	86.7	checkpoint

Table 7. **Pre-Trained backbones.**

F.4. Data Augmentation

F.4.1 VTAB-1K

Following [15], we simply adjust the image dimensions to 224×224.

F.4.2 Few-shot Learning

Also following [34] and [15], we apply color-jitter and RandAugmentation to the training samples. For the validation/test samples, we first resize them to 256×256, then center-crop them to 224×224, and finally normalize them using the mean and standard deviation of ImageNet.

F.5. Hyper-parameter

Following [15], we continue to employ a scaling parameter s. Specifically, our search for s is conducted within the set $\{0.01, 0.1, 1.0, 10, 100\}$. Other hyper-parameters are in Table 9, which is basically the same as those in [15]

and [34]. More information about the datasets used is provided in Table 8. For parameters ne, when the head-level deactivation of Heart-LoRA result in performance degradation, we simply opt for a hyper-parameter setting of $ne = 0$. In this scenario, no information is discarded and P is simply all ones.

	Dataset	#Classes	Train	Val	Test
VTAB-1k	CIFAR100	100	800/1,000	200	10,000
	Caltech101	102			6,084
	DTD	47			1,880
	Oxford-Flowers102	102			6,149
	Oxford-Pets	37			3,669
	SVHN	10			26,032
	Sun397	397			21,750
	Patch Camelyon	2			32,768
	EuroSAT	10			5,400
	Resisc45	45			6,300
	Retinopathy	5			42,670
	Clevr/count	8			15,000
	Clevr/distance	6			15,000
	DMLab	6			22,735
	KITTI-Dist	4			711
	dSprites/location	16			73,728
	dSprites/orientation	16			73,728
	SmallNORB/azimuth	18			12,150
	SmallNORB/elevation	18			12,150
Few-shot	Food-101	101	(1/2/4/8/16)*(#Classes)	20,200	30,300
	Stanford Cars	196		1,635	8,041
	Oxford-Flowers102	102		1,633	2,463
	FGVC-Aircraft	100		3,333	3,333
	Oxford-Pets	37		736	3,669

Table 8. **Dataset Details.**

	optimizer	batch size	learning rate	weight decay	# epochs	lr decay	# warm-up epochs
VTAB-1K	AdamW	64	1e-3	1e-4	100	cosine	10
Few-shot learning	AdamW	64	5e-3	1e-4	100	cosine	10

Table 9. **Other Hyper-parameters.**