Supplementary Material
Mini but Mighty: Finetuning ViTs with Mini Adapters

November 6, 2023

In this supplementary material, we provide (1) a visual representation of our method MiMi for clarity (2) additional experimental results to further analyze the proposed MiMi approach (Multi-task benchmark, and VTAB benchmark). (3) justify in more detail our choice for the design of the importance score (4) the effect of the parameters allocation in Adapters for ViTs (5) provide details regarding the datasets used in our experiments.

S1 Illustration of MiMi Design

In our work, we augment a pre-existing ViT model with an 'adapter' module. For each adapter, the input is denoted as $\boldsymbol{h}_i$ with dimension M_i. The adapter undergoes two primary transformations.

Firstly, it maps $\boldsymbol{h}_i$ to $\boldsymbol{z}_i \in \mathbb{R}^{N_i}$ through a fully-connected layer characterized by a parameter matrix $\boldsymbol{W}_i^{\text{down}}$. A non-linear activation function $\phi(\cdot)$ is also applied during this step. Subsequently, $\boldsymbol{z}_i$ is transformed back to an output $\boldsymbol{r}_i \in \mathbb{R}^{M_i}$ by another fully-connected layer, parameterized by $\boldsymbol{W}_i^{\text{up}}$.

A special feature of the adapter is its residual skip-connection. If $\boldsymbol{r}_i$ is near zero, the adapter essentially acts as an identity function, making minimal changes to the input.

Our design of MiMi is influenced by an observation: if an entire row in $\boldsymbol{W}_i^{\text{down}}$ and an entire column in $\boldsymbol{W}_i^{\text{up}}$ are zero, the adapter behaves as though its complexity is reduced, effectively acting as if it has a smaller dimension M_i, as illustrated in Fig.S1.

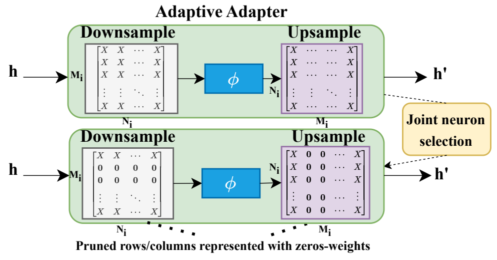


Figure S1: Illustration of the design of adapter after using MiMi.

S2 MiMi versus Vanilla training on CIFAR-100

Looking at Fig. S2), we observe *the significant performance gap between vanilla adapters compared to adapters trained with MiMi approach.* Our method outperforms vanilla adapters with a more than 10% accuracy gap at small sizes.

S3 Local versus Global Neurons Selection

Here below, we provide extra experiments for adapters with different sizes for VGG-Flowers, CIFAR-10, and CIFAR-100.

Tables S1 and S2 report the performance of MiMi algorithm with respect to vanilla training -*Baseline*- on VGG-Flowers, CIFAR-10 and CIFAR-100. MiMi outperforms vanilla adapters on all the adapters with a significant performance gap. Interestingly, this

Table S1: A comparative performance analysis of local and global neuron selection on VGG-Flowers.

Method	**Neurons Removal**	**Selection**	**Iterative**	**Scaling**	σ							
					32	64	128	256	512	1024	2048	4096
Vanilla adapters					94.80	90.12	89.42	88.85	86.03	86.09	85.14	85.14
Baselines	Local	✓			-	96.10	95.57	96.15	96.23	96.76	95.83	95.83
	Local	✓	✓		-	96.41	96.65	96.72	96.72	**96.81**	**96.83**	94.54
	Global	✓			-	94.88	95.28	95.66	95.45	95.56	96.03	96.03
	Global	✓		✓	-	96.10	95.82	96.34	96.50	96.15	96.03	96.03
MiMi	Global	✓	✓	✓	-	**96.59**	**96.92**	**96.73**	**96.81**	96.55	96.47	**96.17**

Table S2: A comparative performance analysis of local and global neuron selection on CIFAR-10 and CIFAR-100.

Method	**Neurons Removal**	**Selection**	**Iterative**	**Scaling**	σ							
					32	64	128	256	512	1024	2048	4096
					CIFAR-10							
Vanilla adapters	-				97.89	97.39	97.06	96.60	96.53	96.27	82.06	82.06
Baseline	Local	✓	✓		-	**98.06**	97.92	**97.64**	97.11	**96.91**	**96.44**	93.49
	Random	✓	✓		-	97.79	97.68	97.42	96.98	96.56	96.15	93.84
	Gradient	✓	✓		-	97.76	97.73	97.56	97.14	96.71	96.29	93.92
MiMi	Global	✓	✓	✓	-	97.98	97.92	97.49	**97.15**	96.71	96.04	**95.57**
					CIFAR-100							
Vanilla adapters	-				86.22	85.02	84.33	83.54	82.26	82.19	83.17	82.19
Baseline	Local	✓	✓		-	86.88	86.15	85.30	84.06	83.71	**83.35**	78.44
	Random	✓	✓		-	85.97	85.71	84.82	84.23	83.72	83.08	78.24
	Gradient	✓	✓		-	86.11	85.67	85.16	84.57	84.16	83.35	78.35
MiMi	Global	✓	✓	✓	-	**87.12**	**86.22**	**85.42**	**84.67**	**83.25**	82.67	**82.10**

gap in performance expands, as we compare smaller adapters $\sigma = 256, ..., 4096$ (higher compression ratio).

Furthermore, these results emphasize the usefulness of each component of the importance score for MiMi. We notice that applying global neuron selection, normalization, and iterative training outperforms local neuron selection on almost all adapter sizes for VGG-Flowers (Table S1) and CIFAR-100 (Table S2). This indicates that each component of the importance score of MiMi is important to boost performance and reduce the parameters.

S4 Vanilla versus MiMi Training for Adapters

To validate the greater optimization performance of MiMi, in Figs. S3, S4, we show the training loss curves of vanilla, and MiMi training of adapters for CIFAR-100, and SVHN, respectively. At the end of the training, the two models (*i.e.* vanilla training and MiMi) have similar numbers of training parameters.

We notice that the loss of the training using MiMi algorithm is much smoother than vanilla training resulting in adapters that generalize well on the downstream task as previously shown in Fig. 4. Furthermore, we notice spikes in the training loss of MiMi, due to the removal of neurons after each cycle. Eventually, sequential training and neuron selection is a successful strategy to find small networks while maintaining good performance, since directly training small networks does not provide similar results [5].

Table S3: Effect of adapters compression rate σ_i on adapters performance in terms for top-1 accuracy (%). Compression rate $\sigma = \infty$ is equivalent to not adding an adapter. We vary the σ_i value for each ViT stage (I, II, III, IV).

$\boldsymbol{\sigma}_i$ in each Swin Stage				**Dataset**	
I	**II**	**III**	**IV**	**CIFAR-10**	**VGG-Flowers**
128	128	32	32	**97.91**	**89.90**
32	32	128	128	97.64	87.05
128	32	128	32	97.84	89.45
32	128	32	128	97.72	88.49
128	128	128	32	97.79	89.22
256	128	64	32	**97.91**	89.84
32	64	128	256	97.43	86.90
128	128	∞	∞	95.29	73.38
∞	∞	128	128	97.40	87.04
∞	128	128	∞	96.97	84.65
128	∞	∞	128	96.53	80.84

S5 Impact of the parameter allocation

In this ablation study, we validate the idea that every layer of a ViT needs to be adapted differently in order to learn a new task. The Swin-B vision transformer model that we use in all our experiments consists of 4 stages. Therefore, we propose to evaluate the performance when we vary the adapter size in each stage. The results are reported in table S3.

First, it shows that the size of adapters has a strong effect on the model performance. In general, the best performances are achieved when using adapters with a higher number of parameters. Furthermore, bigger sizes of adapters are not sufficient for better performance, but subject to which stage they are injected into.

We observe that adding adapters to late stages (*i.e.* III and IV) boosts the performance better than injecting them into early stages: adapters with $\sigma_i = 128$ added to (III, IV) stages rather than (I, II) improve the performance from 95.29%, 73.38% to 97.40%, 87.04% on CIFAR-10 and VGG-Flowers, respectively.

S6 Illustrations of Local versus Global Neuron Removal

Figures S5 and S6 show additional illustrations of the distribution of the removed and remaining neurons using MiMi on VGG-Flowers, and CIFAR-10. We show the learned adapters at different cycles for both local and global neuron selection methods. We also complete these visualizations (Figures S7 and S8) with histograms where we show the percentages of remaining neurons. Overall, these experiments show that our method is capable of obtaining different parameter allocations that are specific to every task.

S7 Magnitude assumption as importance score

In this section, we provide more insights on the importance score employed within MiMi. In particular, under Gaussian input assumption for adapters and imposing weight decay at training time, we will see that, towards a better choice of parameters to be removed, considering just $\boldsymbol{W}_i^{down}$ is sub-optimal, and $\boldsymbol{W}_i^{up}$ should be accounted as well. We drop the adapter index i for abuse of notation, as we will always refer to the same adapter.
Let us have an m-dimensional input $\boldsymbol{h}$, whose elements are distributed according to a Gaussian $\mathcal{N}(\mu_k, \Sigma_k)$. We assume the adapter has already been trained; hence we consider, in the down-sampling phase all the w_{jk}^{down} as constants. From the property of linear expectations, we know that, before reaching the non-linear activation, the post-synaptic potential is still a Gaussian random variable, having an average

$$\mu_j^{down} = \sum_{k=1}^{m} W_{jk}^{down} \cdot \mu_k \tag{1}$$

and variance

$$\Sigma_j^{down} = \sum_{k=1}^{m} W_{jk}^{down} \cdot \left[W_{jk}^{down} \Sigma_{kk} + 2 \sum_{k'<k} W_{jk'}^{down} \Sigma_{kk'} \right], \tag{2}$$

where Σ_{ab} indicates an element of the covariance matrix for the input of the adapter. For the sake of tractability, if we assume $\Sigma_{kk'} = 0\ \forall k \neq k'$, equation 2 simply reduces to

$$\Sigma_j^{down} = \sum_{k=1}^{m} \left(W_{jk}^{down}\right)^2 \Sigma_{kk}. \tag{3}$$

In transformers, the commonly-used activation function is the Gaussian error linear unit (GELU), whose analytical expression is

$$\phi(x) = x \cdot \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right] \tag{4}$$

where $\operatorname{erf}(\cdot)$ is the error function. For values close to zero, or larger than zero, it can be approximated to the identity function, while for values much lower than zero, it asymptotically tends to zero. Let us focus on the first scenario: we can approximate the post-synaptic potential to the output of the non-linearity, saying that the output

$$z_j \approx \mathcal{N}(\mu_j^{down}, \Sigma_j^{down}). \tag{5}$$

At this point, the signal undergoes an up-sampling: following up on the same approach adopted for the down-sampling, we find that the output $\boldsymbol{r}$ still follows a Gaussian distribution having an average

$$\mu_l^{up} = \sum_{j=1}^{n} W_{jl}^{up}, \mu_j^{down} = \sum_{j=1}^{n} W_{jl}^{up} \sum_{k=1}^{m} W_{jk}^{down} \cdot \mu_k \tag{6}$$

and variance

$$\Sigma_l^{up} = \sum_{j=1}^{n} \left(W_{jl}^{up}\right)^2 \tag{7}$$

$$\Sigma_j^{down} = \sum_{j=1}^{n} \left(W_{jl}^{up}\right)^2 \sum_{k=1}^{m} \left(W_{jk}^{down}\right)^2 \Sigma_{kk} \tag{8}$$

$$\begin{aligned} \mu_{l,\bar{a}}^{up} &= \mu_l^{up} - W_{al}^{up} \sum_{k=1}^{m} W_{ak}^{down} \cdot \mu_k \\ \Sigma_{l,\bar{a}}^{up} &= \Sigma_l^{up} - (W_{al}^{up})^2 \sum_{k=1}^{m} \left(W_{ak}^{down}\right)^2 \Sigma_{kk} \end{aligned} \tag{9}$$

In order to assess the impact of removing a whole neuron in the embedding space, we can write the KL-divergence of the distribution for r_l with and without the a-th neuron in the embedding space:

$$\begin{aligned} &D_{\mathrm{KL}}(r_l, r_{l,\bar{a}}) = \\ &\quad \log\left(\frac{\Sigma_l^{up} - (W_{al}^{up})^2 \sum_{k=1}^{m} \left(W_{ak}^{down}\right)^2 \Sigma_{kk}}{\Sigma_l^{up}}\right) \\ &+ \frac{(\Sigma_l^{up})^2 + \left(\mu_l^{up} - \mu_l^{up} + W_{al}^{up} \sum_{k=1}^{m} W_{ak}^{down} \cdot \mu_k\right)}{2 \cdot \left[\Sigma_l^{up} - (W_{al}^{up})^2 \sum_{k=1}^{m} \left(W_{ak}^{down}\right)^2 \Sigma_{kk}\right]^2} - \frac{1}{2}. \end{aligned} \tag{12}$$

According to equation 9, we rewrite equation 12 as 13.

$$\begin{aligned} &D_{\mathrm{KL}}(r_l, r_{l,\bar{a}}) = \\ &\quad \log\left(1 - \frac{(W_{al}^{up})^2 \sum_{k=1}^{m} \left(W_{ak}^{down}\right)^2 \Sigma_{kk}}{\Sigma_l^{up}}\right) \\ &+ \frac{(\Sigma_l^{up})^2 + \left(W_{al}^{up} \sum_{k=1}^{m} W_{ak}^{down} \cdot \mu_k\right)}{2 \cdot \left[\Sigma_l^{up} - (W_{al}^{up})^2 \sum_{k=1}^{m} \left(W_{ak}^{down}\right)^2 \Sigma_{kk}\right]^2} - \frac{1}{2}. \end{aligned} \tag{13}$$

Let us now investigate which is the a-th neuron which, when removed, causes the least perturbation at the output r_l (or in other words, such that $D_{\mathrm{KL}}(r_l, r_{l,\bar{a}})$ is as low as possible). Looking at the argument of the logarithm, we ask $\frac{(W_{al}^{up})^2 \sum_{k=1}^{m} (W_{ak}^{down})^2 \Sigma_{kk}}{\Sigma_l^{up}} = 0$ and, since we can safely assume Σ_l^{up}, we need to select a such that $(W_{al}^{up})^2 \sum_{k=1}^{m} \left(W_{ak}^{down}\right)^2 \Sigma_{kk} = 0$. Considering that also $\Sigma_{kk} > 0\ \forall k$, we satisfy the condition if either:

- $W_{ak}^{down} = 0\ \forall k$, namely the L^1 norm for $\boldsymbol{W}_{-a}^{down}$ is zero;
- $W_{al}^{up} = 0$. Considering though that this condition needs to be satisfied for all the l outputs of the adapters, we ask $W_{al}^{up} = 0\ \forall l$ or, in other words, the L^1 norm for $\boldsymbol{W}_{a-}^{up}$ is also zero.

We observe that, when either of the two conditions is met, the KL divergence is zero as

$$D_{\mathrm{KL}}(r_l, r_{l,\bar{a}}) = \log(1) + \frac{(\Sigma_l^{up})^2}{2 \cdot (\Sigma_l^{up})^2} - \frac{1}{2} = 0$$

We can also assume that if either $W_{ak}^{down} = 0\ \forall k$ or $W_{al}^{up} = 0\ \forall l$, the norm of the non-zero parameters associated with some neuron a are small when training with any weight penalty regularizer (as the contribution to the output is zero, the signal is either not forward or back-propagated, leaving the weight penalty term the only update for these parameters).

S8 Detailed evaluation of MiMi on MultiTask/DomainNet benchmarks

Table S4 reports the number of trained parameters and the average accuracy across datasets in the DomainNet benchmark. For both, the number of trained parameters is reported in millions, and the average top-1 accuracy on the datasets is reported in the rightest column.

We observe that *full fine-tuning* has generally the highest accuracy, but it requires a huge number of parameters to be finetuned for each dataset. Among the vanilla fine-tuning baselines, we observe that tuning the parameters of the *attention/MLP* layer turns out to be surprisingly effective. Nevertheless, it still requires a high number of task-specific parameters, compared to other PET approaches. *Linear probing* does not perform well illustrating the need to change the feature representations of the model when adapting to new tasks.

PHM, and *Compacter* are effective methods to get on-par performance with *full-model finetune* while adjusting less than 2% of the parameters. Contrarily to what is observed for NLP tasks [9], PETs on visual tasks do not reach *full fine-tuning* performance on any dataset with a low number of trainable parameters (smaller than 2%). VPT does not perform well, indicating that injecting tokens into the embedding space does not help much if the pre-training dataset is different from the downstream task. Generally speaking, all PET methods maintain similar performance rankings on all tasks. This suggests that the choice of the best adaptation strategy does not depend on the downstream task.
Adapters outperform all PET methods in terms of accuracy (69.39% for DomainNet, 92.91% for Multi-task) but just with a higher number of trainable parameters (1.37M, 4.90% of the total) for $\sigma = 32$.

Adapters outperform *AdaptFormer* with fewer parameters (92.91% with 1.37M parameters, versus 92.27% with 2.98M parameters). This result indicates that adapting the representations after both MSA and MLP blocks, as done in *Adapters* (see Fig. S9), allows better adaptation than acting only on the MLP block via a parallel branch (as done in *AdaptFormer* [2]).

When comparing *adapter* with uniform and proportional parameter distribution, we observe that allocating parameters proportionally to the layer dimension performs better. Indeed, adapters with $\sigma = 32$ outperform adapters with $n_i = 47 \forall i$ (70.65% vs 69.39% in DomainNet, 93.53% vs 92.91% in Multi-task). This suggests that the last layers, which have higher dimensionality, are more task-specific, and consequently require more adaptation. We also show that reducing the size of adapters ($n_i = 23$) hurts the performance with a drop which, despite being marginal for Multi-task (0.23%) is more consistent in DomainNet (1.01%). This emphasizes that training tiny adapters in a vanilla fashion leads to unsatisfying performance and motivates our specific training procedure.

S8.1 Impact of ρ

In this section, we investigate the effect of the hyper-parameter ρ (amount of neuron removal) in our method MiMi.

In Fig. S10. We notice that higher values of ρ hurt the performance because we remove many parameters after each cycle, but we reduce the size of adapters significantly. On the other hand, if ρ is small (i.e 25%), we maintain good performance on the VGG-Flowers dataset, but it requires higher training cycles C to reach the target compression rate σ_{target}.

We have a trade-off between the performance, and training budget in order to reach the σ_{target}. Re-

Table S4: Results on the DomainNet benchmark [20]. C, I, P, Q, R, and S stand for Clipart, Infograph, Painting, Quickdraw, Real, and Sketch respectively. † is equivalent to Adapters with $\sigma = 8$. Methods are grouped according to the relative number of trainable parameters ($\leq 2\%$, $\in]2, 10[\%$, $\geq 10\%$)

Method	# **Params (M)** $\downarrow$	**Trained (%)** $\downarrow$	**C.**	**I.**	**P.**	**Q.**	**R.**	**S.**	**Mean** $\uparrow$
Full fine-tuning	27.8	100	79.16	48.29	74.64	75.88	86.21	73.26	**72.90**
Att-blocks	8.93	32.14	48.36	75.38	73.28	86.13	72.81	72.57	72.57
MLP-blocks	17.54	63.12	79.23	48.11	75.02	74.82	86.35	73.29	72.80
MiMi (0 cycle)†	4.44	16.15	78.11	46.93	74.38	72.19	86.09	71.97	71.61
AdaptFormer-256	2.54	9.24	75.32	43.74	72.16	66.00	83.88	67.68	68.13
AdaptFormer-64	0.84	3.06	73.76	42.38	71.11	63.41	83.23	66.14	66.67
VPT (100 tokens)	0.71	2.57	64.33	18.65	63.20	59.40	79.65	56.41	56.94
Adapters ($\sigma = 32$)	1.37	4.90	77.42	46.51	74.06	69.81	85.30	70.84	**70.65**
Adapters ($n_i = 47$)	1.37	4.90	76.15	45.28	73.04	67.86	84.83	69.17	69.39
Adapters ($n_i = 23$)	0.68	2.47	75.28	44.17	72.41	66.44	83.98	68.02	68.38
MiMi (1 cycle)	0.80	2.89	76.83	46.00	73.76	67.93	85.05	69.61	69.86
Linear prob	**0.27**	**0.95**	62.89	33.96	64.93	42.95	81.69	54.24	56.77
PHM-Adapter	0.47	1.72	75.79	44.62	72.49	66.62	83.73	68.51	68.62
Compacter	0.41	1.44	75.25	44.19	72.09	66.01	83.42	67.99	68.16
BitFit	0.34	1.22	72.14	41.07	70.00	60.39	82.43	64.43	65.08
VPT (10 tokens)	0.32	1.15	61.91	24.87	57.05	57.09	76.94	55.12	55.49
SSF	0.28	0.96	74.45	42.64	71.72	64.70	82.95	65.90	67.06
Fact-TK$_{32}$	0.33	1.18	72.46	41.14	70.32	61.26	80.23	63.87	64.88
Adapters ($n_i = 1$)	0.30	1.07	72.12	41.30	69.93	59.95	82.49	64.18	65.00
MiMi (2 cycles)	0.53	1.92	76.83	45.45	73.11	66.67	84.42	69.05	**69.26**
MiMi (3 cycles)	0.40	1.43	75.44	44.60	72.59	64.73	83.87	68.05	68.21
MiMi (4 cycles)	0.30	1.07	74.38	43.52	71.50	63.41	83.12	67.46	67.23

moving too many parameters at each cycle hurts performance. Maintaining good performance requires a higher number of training cycles C.

S8.2 Evaluation on VTAB Benchmark

We experiment on *VTAB* [26] is a collection of 19 diverse visual classification tasks, which are organized into three groups: *Natural* - tasks that contain natural images captured using standard cameras; *Specialized* - tasks that contain images captured via specialized equipment, such as medical and satellite imagery; and *Structured* - tasks that require geometric comprehension like object counting. Each task of VTAB contains 1000 training examples. Following [26], we use the provided 800-200 split of the train set to determine hyperparameters and run the final evaluation using the full training data. We report the average accuracy score on the test set within three runs.

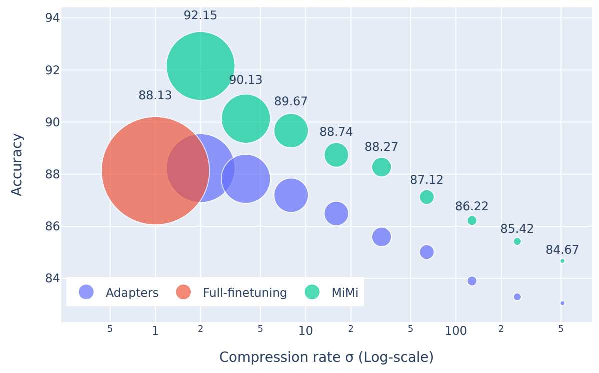


Figure S2: Comparison of top-1 accuracy of vanilla adapters, and MiMi with respect to compression rate σ on CIFAR-100 dataset. All MiMi results originate from the same MiMi run. Adapters are trained for the exact same number of epochs as their MiMi counterparts. The size of blob markers represents the number of trainable parameters. We notice that at $\sigma = 2, 4, 8$, MiMi outperforms full finetuning.

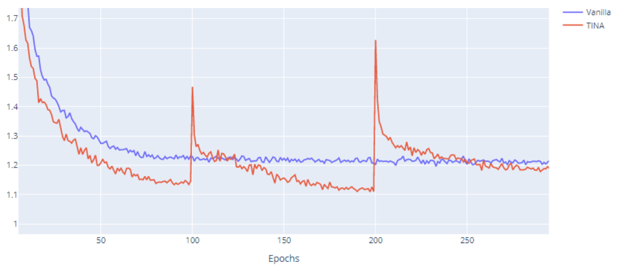


Figure S3: Training loss curves of finetuning adapters with vanilla, and MiMi training on CIFAR-100 dataset.

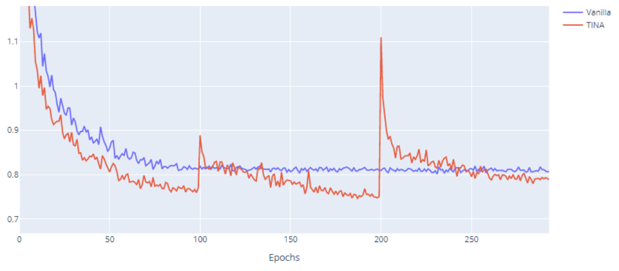
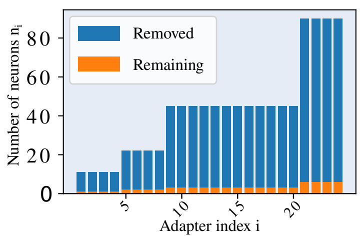
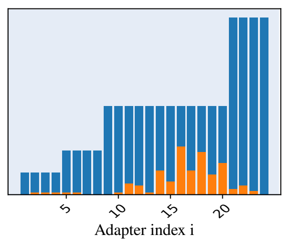


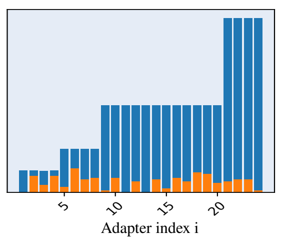
Figure S4: Training loss curves of finetuning adapters with vanilla, and MiMi training on SVHN dataset.



(a) Local pruning: all datasets.

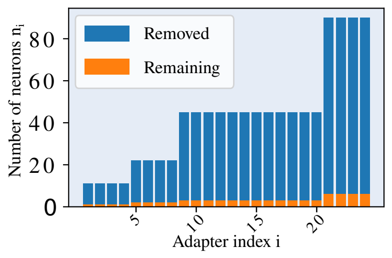


(b) Global pruning: VGG-Flowers .

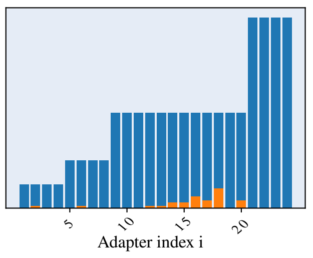


(c) Global pruning: CIFAR-10.

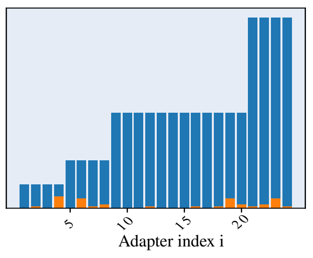
Figure S5: Layer-wise analysis of adapter's neurons distribution at 3^{rd} cycle. Bar plots represent a number of neurons n_i at each adapter i using local, and global neuron selection for VGG-Flowers and CIFAR-10, respectively.



(a) Local pruning: all datasets.

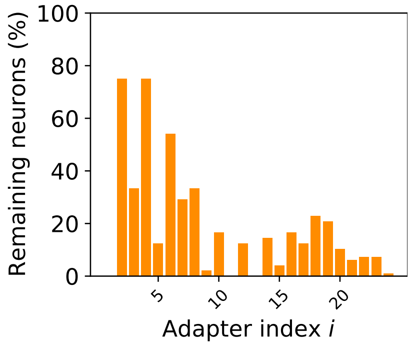


(b) Global pruning: VGG-Flowers.

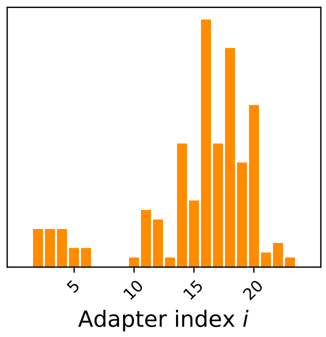


(c) Global pruning: CIFAR-10.

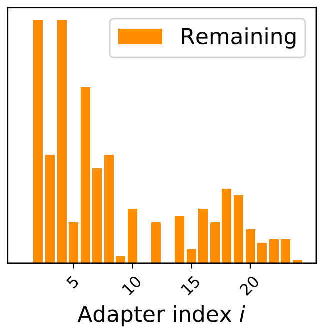
Figure S6: Layer-wise analysis of adapter's neurons distribution at 5^{th} cycle. Bar plots represent the number of neurons n_i at each adapter i using local and global neuron selection for VGG-Flowers and CIFAR-10, respectively.



(a) Local pruning: all datasets.

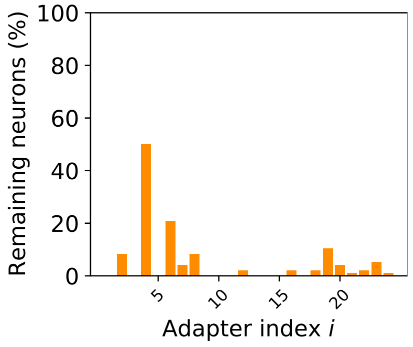


(b) Global pruning: VGG-Flowers .

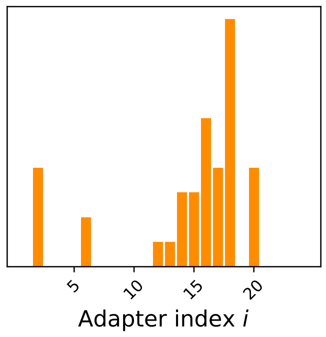


(c) Global pruning: CIFAR-10.

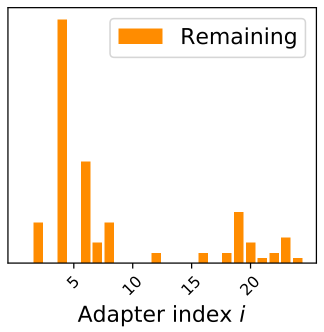
Figure S7: Layer-wise analysis of adapter's neurons distribution at 3^{rd} cycle. **Normalized** bar plots represent **percentage (%) of remaining neurons** n_i at each adapter i using local, global neuron selection for VGG-Flowers and CIFAR-10, respectively.



(a) Local pruning: all datasets.



(b) Global pruning: VGG-Flowers .



(c) Global pruning: CIFAR-10.

Figure S8: Layer-wise analysis of adapter's neurons distribution at 5^{th} cycle. **Normalized** bar plots represent **percentage (%) of remaining neurons** n_i at each adapter i using local, global neuron selection for VGG-Flowers and CIFAR-10, respectively.

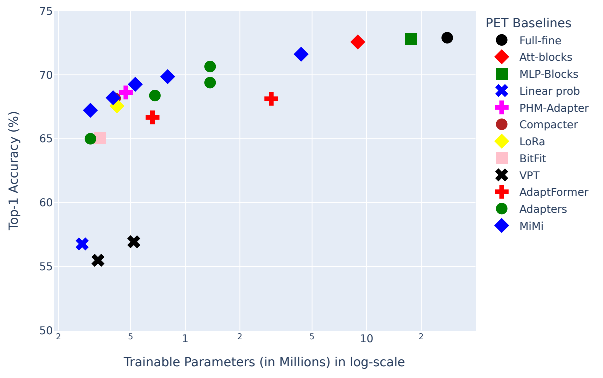


Figure S9: Evaluation of PET baselines mean top-1 accuracy on **DomainNet** benchmark. Reducing MiMi (◆) parameters does not significantly reduce performance compared to other PET baselines.

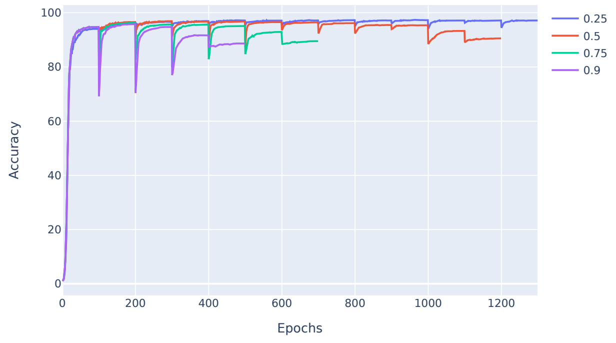


Figure S10: Analysis of MiMi performance on VGG-Flowers dataset with different values of ρ. If ρ is very high, the drop in performance is significant, but it requires less C training cycles to reach σ_{traget}.

Table S5: Per-task fine-tuning results for VTAB with a pre-trained ViT-B/16. Results for other baselines are reported from [10].

		CIFAR-100	**Caltech101**	**DTD**	**Flowers102**	**Pets**	**SVHN**	**Sun397**	**Mean**	**Patch Camelyon**	**EuroSAT**	**Resisc45**	**Retinopathy**	**Mean**	**Clevr/count**	**Clevr/distance**	**DMLab**	**KITTI/distance**	**dSprites/location**	**dSprites/orientation**	**SmallNORB/azimuth**	**SmallNORB/elevation**	**Mean**
(a)	Full	68.9	87.7	64.3	97.2	86.9	87.4	38.8	75.88	79.7	95.7	84.2	73.9	83.36	56.3	58.6	41.7	65.5	57.5	46.7	25.7	29.1	47.64
Head-oriented																							
(a)	Linear	63.4	85.0	63.2	97.0	86.3	36.6	51.0	68.93 (1)	78.5	87.5	68.6	74.0	77.16 (1)	34.3	30.6	33.2	55.4	12.5	20.0	9.6	19.2	26.84 (0)
	Partial-1	66.8	85.9	62.5	97.3	85.5	37.6	50.6	69.44 (2)	78.6	89.8	72.5	73.3	78.53 (0)	41.5	34.3	33.9	61.0	31.3	32.8	16.3	22.4	34.17 (0)
	Mlp-2	63.2	84.8	60.5	97.6	85.9	34.1	47.8	67.70 (2)	74.3	88.8	67.1	73.2	75.86 (0)	45.2	31.6	31.8	55.7	30.9	24.6	16.6	23.3	32.47 (0)
	Mlp-3	63.8	84.7	62.3	97.4	84.7	32.5	49.2	67.80 (2)	77.0	88.0	70.2	56.1	72.83 (0)	47.8	32.8	32.3	58.1	12.9	21.2	15.2	24.8	30.62 (0)
	Mlp-5	59.3	84.4	59.9	96.1	84.4	30.9	46.8	65.98 (1)	73.7	87.2	64.8	71.5	74.31 (0)	50.8	32.3	31.5	56.4	7.5	20.8	14.4	20.4	29.23 (0)
	Mlp-9	53.1	80.5	53.9	95.1	82.6	24.4	43.7	61.90 (1)	78.5	83.0	60.2	72.3	73.49 (0)	47.5	27.9	28.9	54.0	6.2	17.7	10.8	16.2	26.15 (0)
Backbone-oriented																							
(b)	Sidetune	60.7	60.8	53.6	95.5	66.7	34.9	35.3	58.21 (0)	58.5	87.7	65.2	61.0	68.12 (0)	27.6	22.6	31.3	51.7	8.2	14.4	9.8	21.8	23.41 (0)
	Bias	72.8	87.0	59.2	97.5	85.3	59.9	51.4	73.30 (3)	78.7	91.6	72.9	69.8	78.25 (0)	61.5	55.6	32.4	55.9	66.6	40.0	15.7	25.1	44.09 (2)
	Adapter-256	74.1	86.1	63.2	97.7	87.0	34.6	50.8	70.50 (4)	76.3	88.0	73.1	70.5	76.98 (0)	45.7	37.4	31.2	53.2	30.3	25.4	13.8	22.1	32.39 (0)
	Adapter-64	74.2	85.8	62.7	97.6	87.2	36.3	50.9	70.65 (4)	76.3	87.5	73.7	70.9	77.10 (0)	42.9	39.9	30.4	54.5	31.9	25.6	13.5	21.4	32.51 (0)
	Adapter-8	74.2	85.7	62.7	97.8	87.2	36.4	50.7	70.67 (4)	76.9	89.2	73.5	71.6	77.80 (0)	45.2	41.8	31.1	56.4	30.4	24.6	13.2	22.0	33.09 (0)
Visual-Prompt Tuning																							
(c)	shallow-VPT	77.7	86.9	62.6	97.5	87.3	74.5	51.2	76.81 (4)	78.2	92.0	75.6	72.9	79.66 (0)	50.5	58.6	40.5	67.1	68.7	36.1	20.2	34.1	46.98 (4)
	Prompt length (p)	100	5	1	200	50	200	1	79.4	5	50	50	10	28.7	100	200	100	100	100	100	200	200	137.5
	Tuned / Total (%)	0.18	0.10	0.04	0.27	0.08	0.19	0.36	0.17	0.01	0.05	0.09	0.01	0.04	0.10	0.18	0.09	0.09	0.10	0.10	0.19	0.19	0.13
	deep-VPT	78.8	90.8	65.8	98.0	88.3	78.1	49.6	78.48	81.8	96.1	83.4	68.4	82.43	68.5	60.0	46.5	72.8	73.6	47.9	32.9	37.8	54.98
	Prompt length (p)	10	10	10	1	1	50	5	12.4	100	100	10	1	52.8	50	200	100	50	10	50	200	200	107.5
	Tuned / Total (%)	0.20	0.20	0.15	0.10	0.04	0.54	0.41	0.23	1.06	1.07	0.15	0.02	0.57	0.54	2.11	1.07	0.54	0.12	0.55	2.12	2.11	1.14
(Ours)	MiMi	61.39	86.77	66.65	96.13	90.98	79.3	53.4	76.37	83.06	95.77	85.9	75.51	85.06	62.57	65.77	46.43	74.91	76.58	53.57	24.59	35.91	55.04

S9 Details about Datasets

In Table S6, we report different statistics that capture the diversity of the datasets we use in our experiments.

	Dataset	Train Size	Test Size	Classes
Multi-task	CIFAR-10	50000	10000	10
	CIFAR-100	50000	10000	100
	Oxford Flowers	2040	6149	102
	SVHN	73257	26032	10
DomainNet	Clipart	33525	14604	345
	Infograph	36023	15582	345
	Painting	50416	21850	345
	Quickdraw	120750	51750	345
	Real	120906	52041	345
	Sketch	48212	20916	345

Table S6: Datasets used in our empirical analysis

References

[1] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016. 13

[2] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition, 2022. 5

[3] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 2017. 13

[4] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *CVPR*, 2014. 13

[5] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018. 2

[6] Timnit Gebru, Jonathan Krause, Yilun Wang, Duyun Chen, Jia Deng, and Li Fei-Fei. Fine-grained car detection for visual census estimation. In *AAAI*, 2017. 13

[7] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research*, 2013. 13

[8] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019. 13

[9] 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 *International Conference on Machine Learning*. PMLR, 2019. 5

[10] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning, 2022. 11

[11] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017. 13

Table S7: Specifications of the various datasets evaluated. $\star$: we randomly sampled the `train` and `val` sets since there are no public splits available

Dataset	**Description**	**# Classes**	**Train**	**Val**	**Test**
Fine-grained visual recognition tasks (FGVC)					
CUB-200-2011 [24]	Fine-grained bird species recognition	200	5,394*	600*	5,794
NABirds [22]	Fine-grained bird species recognition	55	21,536*	2,393*	24,633
Oxford Flowers [19]	Fine-grained flower species recognition	102	1,020	1,020	6,149
Stanford Dogs [13]	Fine-grained dog species recognition	120	10,800*	1,200*	8,580
Stanford Cars [6]	Fine-grained car classification	196	7,329*	815*	8,041
Visual Task Adaptation Benchmark (VTAB-1k) [26]					
CIFAR-100 [14]	Natural	100	800/1000	200	10,000
Caltech101 [16]		102			6,084
DTD [4]		47			1,880
Flowers102 [19]		102			6,149
Pets [21]		37			3,669
SVHN [18]		10			26,032
Sun397 [25]		397			21,750
Patch Camelyon [23]	Specialized	2	800/1000	200	32,768
EuroSAT [8]		10			5,400
Resisc45 [3]		45			6,300
Retinopathy [12]		5			42,670
Clevr/count [11]	Structured	8	800/1000	200	15,000
Clevr/distance [11]		6			15,000
DMLab [1]		6			22,735
KITTI/distance [7]		4			711
dSprites/location [17]		16			73,728
dSprites/orientation [17]		16			73,728
SmallNORB/azimuth [15]		18			12,150
SmallNORB/elevation [15]		9			12,150

[12] Kaggle and EyePacs. Kaggle diabetic retinopathy detection, July 2015. 13

[13] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011. 13

[14] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009. 13

[15] Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR*, 2004. 13

[16] Fei-Fei Li, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE TPAMI*, 2006. 13

[17] Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017. 13

[18] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. 13

[19] Maria-Elena Nilsback and Andrew Zisserman. Au-

tomated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*. IEEE, 2008. 13

[20] Yingwei Pan, Yehao Li, Qi Cai, Yang Chen, and Ting Yao. Multi-source domain adaptation and semi-supervised domain adaptation with focus on visual domain adaptation challenge 2019, 2019. 6

[21] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Cats and dogs. In *CVPR*, 2012. 13

[22] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *CVPR*, 2015. 13

[23] Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant cnns for digital pathology. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2018. 13

[24] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 13

[25] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 13

[26] 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 arXiv:1910.04867*, 2019. 6, 13