Parameter Efficient Mamba Tuning via Projector-targeted Diagonal-centric Linear Transformation

Supplementary Material

Algorithm S1 ProDiaL: Projector-targeted Diagonal-centric Linear Transformation

```
Require: Pretrained Projector weight $W$, Downstream task data $X$, Learning rate $\eta$, Hyperparameter $r_b, r_\epsilon$ for ProDiaL.
Ensure: Fine-tuned Projector weight $W'$
 1: Set the hyperparameter $r_b, r_\epsilon$ and learning rates $\eta$
 2: Initialize $r_b$ small block matrices $x_1, ..., x_{r_b} \in \mathbb{R}^{(d_{in}/r_b)\times(d_{in}/r_b)}$ with identity matrix
 3: Construct $D_a = diag(x_1, ..., x_{r_b})$
 4: Construct $D_b = [\mathbb{I} - relu(\mathbb{I} * D_a)] + (\mathbf{1} - \mathbb{I}) * D_a$
 5: Initialize scaling factor $s$ with one vector
 6: Initialize low-rank matrices $A_\epsilon$ with random noise $\mathcal{N}(0, \sigma^2)$ and $B_\epsilon$ with zeros
 7: while not converged do
 8:     Sample a mini-batch $X_{\text{batch}}$ from the downstream task data
 9:     Compute the intermediate transformation: $W' = sWD_b + B_\epsilon A_\epsilon$
10:     Perform forward pass with $W'$ on $X_{\text{batch}}$
11:     Compute loss $\mathcal{L}(W')$ based on task objective
12:     Backpropagate gradients $\nabla_{D_b}\mathcal{L}, \nabla_{A_\epsilon}\mathcal{L}, \nabla_{B_\epsilon}\mathcal{L}$
13:     Update $D_b \leftarrow D_b - \eta \cdot \nabla_{D_b}\mathcal{L}$
14:     Update $A_\epsilon \leftarrow A_\epsilon - \eta \cdot \nabla_{A_\epsilon}\mathcal{L}$
15:     Update $B_\epsilon \leftarrow B_\epsilon - \eta \cdot \nabla_{B_\epsilon}\mathcal{L}$
16: end while
17: Compute final fine-tuned weight: $W' = sWD_b + B_\epsilon A_\epsilon$
18: return $W'$
```

A. Algorithms of ProDiaL

The ProDiaL (Projector-targeted Diagonal-centric Linear Transformation) is a parameter-efficient fine-tuning (PEFT) method specifically designed for Mamba architecture's Projectors. It efficiently and effectively adapts pretrained Projector weights W to downstream tasks via a diagonal-centric linear transformation, significantly minimizing the number of learnable parameters. Algorithm S1 presents the full detailed ProDiaL algorithm.

The algorithm begins with the initialization of key components and hyperparameters. Firstly, small block matrices $x_1, ..., x_{r_b}$ are initialized as identity matrices and used to construct the auxiliary block-diagonal transformation matrix D_a. The hyperparameter r_b controls the size and the number of these small block matrices. As r_b increases, the sizes of each small block decreases, and the number of small blocks increases, as illustrated in Fig. S1. When r_b equals the input dimension d_{in}, the small block matrices form a perfectly diagonal matrix. With these small block matrices, the block-diagonal matrix D_a is constructed. To facilitate faster convergence, ProDiaL trains the difference between identity matrix and diagonal entries. Then, the final diagonal-centric linear transformation matrix D_b is derived from D_a. Additionally, a scaling factor s is initialized as a vector of ones, and low-rank matrices A_ϵ and B_ϵ are initialized with random noise and zeros, respectively, to address off-diagonal matrices.

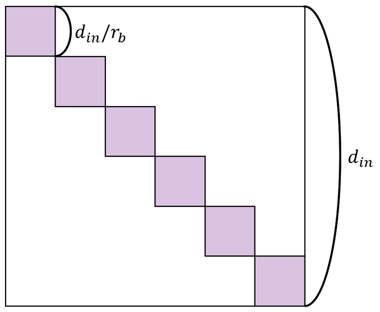

Figure S1. **Block-Diagonal Matrix Design for ProDiaL.** The diagram illustrates the block-diagonal structure of the transformation matrix D_b, with r_b controlling the size and number of small block matrices $(x_1, ..., x_{r_b})$. As r_b increases, the block size decreases.

After the initialization and settings, the fine-tuning process iteratively refines these components using mini-batches of downstream task data. For each mini-batch, the algorithm computes the updated Projector weights W' as a combination of the scaled diagonal transformation sWD_b and the low-rank adjustment for off-diagonal matrices $B_\epsilon A_\epsilon$. A forward pass is performed using W', followed by the computation of the task-specific loss $L(W')$. Next, the gradients obtained from $L(W')$ are backpropagated to update $D_b, A_\epsilon, \text{and} B_\epsilon$ via gradient descent. This process is repeated until convergence, ensuring that the learned transformations effectively adapt W to the downstream task.

Upon convergence, the final fine-tuned weights W' are returned as the output of this algorithm. By employing a block-diagonal structure in D_b and low-rank matrices for A_ϵ and B_ϵ, ProDiaL achieves strong downstream task performance with parameter efficiency and flexibility. This approach is particularly advantageous for large-scale Mamba-based models in both vision and language domains, where full fine-tuning is computationally challenging.

B. Trade-off between Performance and Number of Parameters by Controlling r_b and r_ϵ

Our ProDiaL method offers superior flexibility in determining learnable parameters by controlling the size of small block matrices $(x_1, ..., x_{r_b})$ in the block-diagonal matrix D_b using r_b and the low-rank value for LoRA using r_ϵ. To examine how performance and the number of parameters vary depending on r_b and r_ϵ, we conducted experiments using the Vim-tiny model [16] on the Caltech [9] and Flowers datasets [11]. The hyperparameter r_{b1} controls the small block size of the Input Projectors, while r_{b2} controls the small block size of the Output Projectors.

Table S1 shows the performance and the number of parameters for different values of r_b and r_ϵ. Firstly, we confirm that varying the small block sizes (the number of parameters) does not degrade performance. In other words, this demonstrates that it is possible to flexibly adjust the number of parameters by tuning r_b and r_ϵ, enabling a trade-off between performance and computational cost (parameter usage). Interestingly, even replacing the block-diagonal matrix with a diagonal matrix—represented by the case $(r_{b1}, r_{b2}) = (192, 384)$—yields comparable performance with the smallest number of parameters among the same r_ϵ values. Secondly, we observe that the optimal number of parameters for the best performance depends on the dataset. For the Caltech dataset, the highest accuracy is achieved with a relatively small number of parameters, whereas for the Flowers dataset, the best accuracy requires the largest number of parameters. This suggests that simpler datasets (those with higher baseline accuracy) can achieve high performance with fewer parameters, while more complex datasets (those with lower baseline accuracy) need a larger parameters for high performance.

r_{b1}	r_{b2}	r_ϵ	Params	Caltech (%)	Flowers (%)
192	384	16	0.65M	96.24	86.96
32	32	16	0.77M	95.93	89.12
16	32	16	0.80M	95.86	89.22
8	32	16	0.85M	96.01	89.51
32	16	16	0.88M	95.63	87.75
16	16	16	0.91M	95.63	89.80
8	16	16	0.96M	96.24	89.90
32	8	16	1.10M	95.63	87.75
16	8	16	1.13M	95.47	90.10
8	8	16	1.19M	95.70	**90.29**
192	384	8	0.35M	95.70	88.63
32	32	8	0.48M	96.09	87.94
16	32	8	0.50M	96.01	88.92
8	32	8	0.56M	**96.62**	89.12
32	16	8	0.59M	96.24	88.33
16	16	8	0.61M	96.32	89.22
8	16	8	0.67M	96.09	89.61
32	8	8	0.81M	95.70	89.80
16	8	8	0.84M	96.09	89.02
8	8	8	0.89M	95.93	90.10

Table S1. **Performance Across Varying Hyperparameters.** The table demonstrates the impact of varying block sizes (r_{b1}, r_{b2}) and low-rank value (r_ϵ) on the number of parameters and performance for the Caltech and Flowers datasets. Smaller parameter counts perform well on simpler datasets (e.g., Caltech), while larger parameter counts yield better performance on more complex datasets (e.g., Flowers).

	Datasets	SIQA	OBQA
Both-Proj	LoRA	34.14 (2.36M)	38.00 (1.18M)
	DoRA	34.34 (2.45M)	38.20 (1.27M)
	ProDiaL	**34.80** (2.51M)	**39.60** (1.18M)
In-Proj	LoRA	34.80 (1.48M)	36.40 (0.74M)
	DoRA	34.39 (1.55M)	36.80 (0.81M)
	ProDiaL	**35.47** (1.03M)	**37.80** (0.89M)
Out-Proj	LoRA	33.16 (0.89M)	33.80 (0.44M)
	DoRA	33.27 (0.90M)	34.20 (0.46M)
	ProDiaL	**33.93** (0.90M)	**35.00** (0.46M)

Table S2. **Performance (%) on SIQA and OBQA Datasets.**

C. Additional Experiments

C.1. Evaluation on additional datasets

To further evaluate the effectiveness of our proposed ProDiaL method, we conducted additional experiments on SIQA [13] and OBQA [10] datasets. These datasets were selected to assess the wider reasoning capabilities of ProDiaL across diverse tasks and domains. As shown in Tab. S2, our ProDiaL method consistently outperforms both LoRA and DoRA, demonstrating superior performance in line with the results observed on other datasets in the main manuscript. This consistent improvement underscores the robustness of ProDiaL and its ability to effectively adapt across various data distributions and task complexities.

C.2. Evaluation on Mamba2 architecture

To further assess the effectiveness of our ProDiaL method within Mamba 2 architecture [2], we conducted additional experiments on LLMs based on the Mamba2-130M. Mamba 2 is an advanced model that addresses the limitations of Mamba 1 by incorporating selective state updates, efficient parallelization, and a lightweight attention mechanism, ensuring efficient and strong performance even with long sequences. Unlike Mamba 1, where Mamba 2 employs a Hybrid Attention Mechanism, the performance of ProDiaL is not fully explored due to its structural differences. As presented in Tab. S3, our ProDiaL method consistently achieves superior accuracy while maintaining a comparable or smaller number of parameters than other approaches. These results demonstrate that ProDiaL

	Method	HellaSwag	Winogrande	ARC-E	ARC-C	Avg
	Full-FT	38.23 (130.00M)	53.12 (130.00M)	53.54 (130.00M)	28.84 (130.00M)	43.43
Both-Proj	FT	38.76 (90.1M)	53.12 (90.1M)	50.67 (90.1M)	28.84 (90.1M)	42.84
	LoRA	38.50 (2.47M)	53.35 (2.47M)	52.36 (2.47M)	30.20 (2.47M)	43.60
	DoRA	35.24 (2.57M)	52.01 (2.57M)	47.18 (2.57M)	24.15 (2.57M)	39.65
	ProDiaL	**38.57** (2.44M)	**53.83** (2.00M)	**53.03** (2.33M)	**30.46** (2.22M)	**43.97**
In-Proj	FT	39.89 (61.8M)	53.35 (61.8M)	51.56 (61.8M)	27.22 (61.8M)	43.01
	LoRA	37.32 (1.58M)	53.43 (1.58M)	52.02 (1.58M)	30.03 (1.58M)	43.20
	DoRA	35.24 (1.66M)	52.01 (1.66M)	47.18 (1.66M)	24.15 (1.66M)	39.65
	ProDiaL	**37.91** (0.98M)	**53.75** (0.89M)	**53.03** (0.98M)	**30.29** (0.93M)	**43.75**
Out-Proj	FT	40.62 (28.3M)	53.43 (28.3M)	54.08(28.3M)	29.10 (28.3M)	44.31
	LoRA	37.44 (0.89M)	53.28 (0.89M)	52.65 (0.89M)	28.50 (0.89M)	42.97
	DoRA	37.44 (0.90M)	53.59 (0.90M)	52.69 (0.90M)	28.92 (0.90M)	43.16
	ProDiaL	**37.86** (0.90M)	**53.51** (0.50M)	**53.45** (0.68M)	**30.29** (0.90M)	**43.78**

Table S3. **Comparison of accuracy(%) as a downstream task performance of Mamba2 architecture based LLM across various datasets.** Consistent with Mamba1 architecture, ProDiaL methods (in Both-Proj, In-Proj, and Out-Proj) achieve superior accuracy with smaller or similar number of parameter compared to other methods.

effectively enhances downstream task performance in the Mamba 2 architecture, highlighting its strong generalization capability. Details of the hyperparameter configurations, including the block size settings for ProDiaL, are available in Tab. S9.

D. Experiment Details

D.1. Models & Datasets

First, Mamba LLM [6] is the first model to implement the Mamba architecture, achieving faster inference than transformer-based LLMs as input token sizes increase. For Mamba LLM, we adapt the model pretrained on the PILE dataset [5] to other reasoning task datasets: HellaSwag [15], Winogrande [12], ARC-Easy [1], and ARC-Challenge [1]. The HellaSwag dataset is a challenging benchmark for commonsense reasoning that requires contextual understanding to predict the most plausible continuation of a given scenario from multiple choices. The Winogrande dataset is a large-scale benchmark for commonsense reasoning, consisting of sentence pairs with subtle differences, requiring the model to determine the best sentence completion by resolving nuanced context clues and pronoun references. The ARC-Easy dataset, a subset of the AI2 Reasoning Challenge (ARC), contains straightforward science questions at elementary and middle school levels, designed to assess a model's basic factual and scientific reasoning abilities. The ARC-Challenge dataset, also part of the AI2 Reasoning Challenge, includes complex science questions that require advanced reasoning and domain knowledge.

Next, similar that transformer architecture was originally designed for language models but being used for vision tasks [4], the Mamba architecture, initially designed for language models, also has been adapted into Vision Mamba [16] by sequentially processing image tokens using both forward and backward State Space Models (SSMs) [7]. For the Vision Mamba model, we adapt the model pretrained on the ImageNet dataset [3] to other classification datasets, including Stanford Cars [8], Caltech [9], and Flowers [11]. The Stanford Cars dataset includes images of 196 car models spanning various makes and years, offering detailed visual information to support the development of models capable of distinguishing between different car types and designs. The Caltech101 dataset comprises images from 101 object categories with diverse shapes and appearances, providing a foundation for developing models capable of recognizing real-world objects. The Flowers102 dataset contains images of 102 different flower species, capturing a range of visual variations to help models learn fine-grained distinctions among flower types.

D.2. Universal Effectiveness of Experiments

All experiments and conclusions, including the effectiveness of fine-tuning only the projectors and our proposed ProDiaL approach, are universally applicable. This universal applicability has been empirically validated through

	Datasets	HellaSwag	ARC-E	StanfordCars	Flowers
Both-Proj	LoRA	38.54$_{\pm 0.18}$ (2.36M)	54.00$_{\pm 0.11}$ (2.36M)	85.32$_{\pm 0.25}$ (0.63M)	86.88$_{\pm 0.46}$ (0.61M)
	DoRA	38.37$_{\pm 0.22}$ (2.38M)	54.12$_{\pm 0.00}$ (2.45M)	85.34$_{\pm 0.16}$ (0.69M)	87.14$_{\pm 0.50}$ (0.65M)
	ProDiaL	**38.63**$_{\pm 0.25}$ (2.42M)	**55.09**$_{\pm 0.26}$ (2.38M)	**85.48**$_{\pm 0.13}$ (0.67M)	**87.91**$_{\pm 0.24}$ (0.65M)

Table S4. **Multiple Runs for Both Projectors.** Average performance across three runs with random seeds 30, 42, and 100.

Learning rate	1e-5	5e-5	**1e-4**	5e-4	1e-3
LoRA	36.15	37.86	**38.33**	38.31	37.47
DoRA	36.15	37.66	**38.13**	37.33	36.09
ProDiaL	36.17	37.8	**38.92**	37.64	36.46

Table S5. **Optimal Learning Rates for LoRA, DoRA, and ProDiaL.** This experiment is conducted on HellaSwag dataset.

multiple trials conducted under various random seeds. As demonstrated in Tab. S4, the performance remains consistently robust across various random seeds, reinforcing the reliability of our method. This consistency highlights the stability of our approach, suggesting that it is not highly dependent on specific initialization conditions or random factors.

D.3. Optimal Learning Rates Selection for Fair Comparison

To ensure a fair comparison between LoRA, DoRA, and our proposed ProDiaL method, we conducted a comprehensive search to identify optimal learning rates for each approach. As shown in Tab. S5, LoRA, DoRA, and ProDiaL achieve their highest performance at the **same learning rate**. We believe this convergence is due to the inherent design similarities between the methods, as both DoRA and ProDiaL build upon the LoRA framework. Consequently, the reported results are based on these optimized settings to ensure that each method performs at its best.

D.4. Training and Evaluation

Our experiments in main manuscript are mainly conducted on Vim [16] based on Mamba 2 architecture [2] and Mamba LLM [6] based on Mamba 1 architecture. Below, we detail the experimental settings for each dataset.

D.4.1. Mamba LLM Experiments

In the experiments presented in Table 3 of the main manuscript, we use the Mamba-130M, configured with 24 layers and a maximum sequence length of 512. The dimensions for the input projectors are set as follows: input projectors have $d_{in} = 768$ and $d_{out} = 3072$, while output projectors are configured with $d_{in} = 1536$ and $d_{out} = 768$.

For the results in Table 4 of the main manuscript, we employ the larger Mamba-370M and Mamba-1.4B models, both configured with 48 layers and a maximum sequence length of 512. Mamba-370M uses input projectors with $d_{in} = 1024$ and $d_{out} = 4096$ and output projectors with $d_{in} = 2048$ and $d_{out} = 1024$. For Mamba-1.4B, the input projectors have $d_{in} = 2048$ and $d_{out} = 8192$, while the output projectors have $d_{in} = 4096$ and $d_{out} = 2048$.

For training Mamba LLM models, we use the AdamW optimizer with a batch size of 4 and a constant learning rate scheduler. Additional hyperparameters including ProDiaL's settings, which control the number of learnable parameters, are provided in Tabs. S6 and S7.

For evaluation, we use Language Model Evaluation Harness framework[1], following [6][2].

D.4.2. Vision Mamba Experiments

In the experiments in Table 3 of the main manuscript, we use the Vim-tiny, which has 24 layers, a patch size of 16, and an input image size of 224. The input projectors are set with $d_{in} = 192$ and $d_{out} = 768$, and output projectors have $d_{in} = 384$ and $d_{out} = 192$. We train the Vim-tiny model for 300 epochs with a batch size of 128, using the AdamW optimizer with a learning rate of $5e-4$ and weight decay of 0.1. A cosine learning rate scheduler with 5 warm-up epochs starting from $1e-6$ is applied. Hyperparameters for ProDiaL, determining the number of learnable parameters, are detailed in Tab. S8.

In Table 4 of the main manuscript, we use the Vim-small, which also has 24 layers, a patch size of 16, and an input image size of 224. Whereas, input projectors are configured with $d_{in} = 384$ and $d_{out} = 1536$, while output projectors have $d_{in} = 768$ and $d_{out} = 384$. Training for Vim-small spans 300 epochs with a batch size of 64, using the AdamW optimizer at a learning rate of $1e-3$, weight decay of 0.05, and dropout rate of 0.05. The cosine learning rate scheduler is also used with a 5-epoch warm-up starting from $1e-6$. For the Caltech dataset, ProDiaL hyperparameters are set as follows: $r_{b1} = 192$, $r_{b2} = 384$, and $r_{\epsilon} = 16$.

D.5. Details for Each Analysis

In this section, we describe the detailed settings for each analysis experiment in the tables and figures of the main manuscript. The training and evaluation details for all models and datasets follow the guidelines provided in Section D.4

Table 1 and **Table 2** in main manuscript present the ablation studies of Vision Mamba and Mamba LLM, respec-

[1]https://github.com/EleutherAI/lm-evaluation-harness
[2]https://github.com/state-spaces/mamba

Method	Settings	HellaSwag	Winogrande	ARC-E	ARC-C
Default	Learning Rate	1e-4	5e-6	5e-6	1e-5
	Total Training Iter (M)	300K	100K	100K	100K
	Sampling Period (N)	10K	5K	5K	5K
ProDiaL	Both-Proj (r_{b1}, r_{b2})	(768, 1536)	(768, 1536)	(64, 64)	(64, 64)
	In-Proj r_{b1}	768	768	32	32
	Out-Proj r_{b2}	1536	1536	128	128
	Off-diagonal r_ϵ	16	8	8	8
LoRA	Low Rank r	16	8	16	16
	Scaling factor α	16	8	16	16
DoRA	Low Rank r	16	8	16	16
	Scaling factor α	16	8	16	16

Table S6. **Fine-Tuning Settings for Mamba-130M.** This table summarizes the hyperparameters and configurations used for training and fine-tuning the Mamba-130M model across reasoning tasks (HellaSwag, Winogrande, ARC-E, and ARC-C). The Default settings include learning rates, total training iterations (in millions), and checkpoint sampling periods (N). For ProDiaL, specific configurations for the block-diagonal matrix (r_{b1}, r_{b2}) in input and output projectors, as well as the low-rank value (r_ϵ) for off-diagonal matrices, are provided. Baseline methods (LoRA and DoRA) use a consistent low-rank value (r) and scaling factor (α) for comparison.

Method	Settings	Mamba-370M	Mamba-1.4B
Default	Learning Rate	5e-7	5e-7
	Total Training Iter (M)	30K	30K
	Sampling Period (N)	1K	1K
ProDiaL	Both-Proj (r_{b1}, r_{b2})	(1024, 2048)	(2048, 4096)
	In-Proj r_{b1}	1024	2048
	Out-Proj r_{b2}	2048	4096
	Off-diagonal r_ϵ	16	64
LoRA	Low Rank r	16	64
	Scaling factor α	16	64
DoRA	Low Rank r	16	64
	Scaling factor α	16	64

Table S7. **Fine-Tuning Settings for Larger Mamba Models.** This table outlines the hyperparameters and configurations used for training and fine-tuning Mamba-370M and Mamba-1.4B on the Winogrande dataset.

	Settings	StanfordCars	Caltech	Flowers
ProDiaL	(r_{b1}, r_{b2})	(192,384)	(192,384)	(16,16)
	r_{b1}	192	192	8
	r_{b2}	384	384	32
	r_ϵ	16	16	8
LoRA	r	16	16	16
	α	16	16	16
DoRA	r	16	16	16
	α	16	16	16

Table S8. **Hyperparameters for Fine-Tuning Vim-tiny on Classification Tasks.** The table presents the hyperparameters used for fine-tuning Vim-tiny on the StanfordCars, Caltech, and Flowers datasets. r_{b1} and r_{b2} represent the low ranks for block-diagonal matrices in input and output projectors, respectively. For LoRA and DoRA methods, r indicates the low rank, while α denotes the scaling factor.

tively. For these experiments, the components of the Vim-tiny model were trained on the SUN dataset [14], and the components of the Mamba-130M model were trained on the HellaSwag dataset. The results from both tables consistently show that the Projectors contribute more significantly to capturing knowledge for downstream tasks compared to the State-Space Model (SSM).

Figure 2 in main manuscript visualizes the importance of training diagonal entries in the linear transformation matrix T for learning an effective linear transformation. This analysis was conducted using the Vim-tiny model trained on the Caltech dataset. The visualization illustrates the input projector at the fourth layer, which was randomly selected. Similar characteristics were observed in other layers and the output projectors, reinforcing the generalizability of this observation.

Figure 4 in main manuscript empirically demonstrates the effectiveness of training diagonal entries in the linear

Method	Settings	HellaSwag	Winogrande	ARC-E	ARC-C
Default	Learning Rate	1e-4	1e-6	1e-5	5e-6
	Total Training Iter (M)	200K	30K	100K	100K
	Sampling Period (N)	10K	1K	5K	5K
ProDiaL	Both-Proj (r_{b1}, r_{b2})	(64, 64)	(64, 128)	(128, 64)	(32, 128)
	In-Proj r_{b1}	32	768	128	256
	Out-Proj r_{b2}	128	1536	256	128
	Off-diagonal r_ϵ	8	8	8	8
LoRA	Low Rank r	16	16	16	16
	Scaling factor α	16	16	16	16
DoRA	Low Rank r	16	16	16	16
	Scaling factor α	16	16	16	16

Table S9. **Fine-Tuning Settings for Mamba2-130M.** This table details the hyperparameters and configurations used for fine-tuning Mamba2-130M on reasoning tasks, including learning rates, training steps, checkpoint intervals, and settings for ProDiaL, LoRA, and DoRA methods.

transformation matrix T. This experiment, conducted on the Vim-tiny model with the Caltech dataset, explores four distinct configurations for fine-tuning T. In the first configuration, the entire T matrix is directly fine-tuned, initialized as an identity matrix to ensure it starts as a standard linear transformation. The second configuration fine-tunes a block-diagonal matrix, where each block is initialized as an identity matrix. In the third configuration, only a diagonal vector was fine-tuned, initialized as a vector of ones. The fourth configuration involved fine-tuning only the off-diagonal matrix, starting from a zero matrix with the diagonal entries masked.

Table 5 in main manuscript investigates whether the diagonal entries in the linear transformation matrix T are effective only for Projectors or across all linear layers in the Mamba architecture. This experiment was also conducted using the Vim-tiny model on the Caltech dataset.

Table 6 in main manuscript explores the role of non-attention modules in the Transformer architecture. For this analysis, the Vision Transformer (ViT-B/16) model pre-trained on ImageNet was fine-tuned on the CIFAR-100 dataset. The model was trained using the Adam optimizer with a batch size of 128 and a learning rate of 0.001 for 5000 iterations. A cosine learning rate scheduler with a warmup period of 500 steps was used. For LoRA adaptation, a low rank of 8 was applied to both the Attention and FFN modules.

Table 7 in main manuscript provides an ablation studies of ProDiaL components, including Block-diagonal matrix D_b, Off-diagonal matrix ϵ, and scaling factor s. This experiment was conducted on the Vim-tiny model fine-tuned on the Caltech dataset.

References

[1] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. 3

[2] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024. 2, 4

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 3

[4] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 3

[5] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020. 3

[6] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 3, 4

[7] Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2021. 3

[8] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013. 3

[9] Fei-Fei Li, Marco Andreeto, Marc'Aurelio Ranzato, and Pietro Perona. Caltech 101, 2022. 2, 3

[10] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018. 2

[11] 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. 2, 3

[12] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64 (9):99–106, 2021. 3

[13] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019. 2

[14] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3485–3492. IEEE, 2010. 5

[15] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019. 3

[16] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024. 2, 3, 4