p-MoD: Building Mixture-of-Depths MLLMs via Progressive Ratio Decay

Supplementary Material

This supplementary material includes the following sections:

- In Section A, we provide more experiment results.
- In Section B, we show visualization results of ***p-MoD*** token selection decisions across different layers.
- In Section C, we make some further discussions about our work.
- In Section D, we provide more implementation details.

A. More Experiments

A.1. Experiments on Larger Model Size

In Table 8, we compare our method against other token compression methods on LLaVA-NeXT-13B baseline model. The results demonstrate that our method significantly outperforms other methods on larger model size, validating the scalability of our approach.

Method	DocVQA	ChartQA	SEED	GQA	AVG over **15** Tasks*
LLaVA-NeXT-13B	73.5	67.2	71.4	65.2	66.5
+ MQT	58.6	54.6	68.8	64.2	63.0
+ FastV	70.2	64.7	70.8	64.7	65.7
+ LLaVolta	70.0	61.7	70.5	64.7	65.0
+ ***p-MoD***	**72.3**	**66.2**	**71.6**	**65.0**	**66.0**

Table 8. **Experiments on LLaVA-NeXT-13B**. Our method significantly outperforms other methods on larger baseline model, validating the scalability of our approach. *Average is computed on all **15** benchmarks used in Table 1 and 2.

A.2. Experiments on Visual Grounding

In Table 9, we compare ***p-MoD*** with other token compression methods on visual grounding benchmarks RefCOCO, RefCOCO+ and RefCOCOg [20, 52]. Our method significantly outperforms other methods, but it still exhibits some performance drop compared to the baseline model. This suggests that current token pruning methods still exhibit limitations on visual grounding, which might be one of the major challenges that future works in this field should aim to address.

Method	RefCOCO Val	RefCOCO+ Val	RefCOCOg Val
LLaVA-NeXT-7B	82.67	73.73	79.41
+MQT	68.77	59.07	63.52
+LLaVolta	79.96	70.89	76.79
+FastV	73.83	64.90	69.78
+p-MoD	**80.23**	**70.94**	**76.96**

Table 9. **Experiments on Visual Grounding**. Our method significantly outperforms other methods on visual grounding benchmarks RefCOCO, RefCOCO+ and RefCOCOg. We report ACC@0.5 metric on the validation sets of these benchmarks.

A.3. Compairson with More Relate Works

In addition to comparing ***p-MoD*** against three strong token compression methods in Table 5, we provide comparison with more methods in Table 10. Under the same token compression ratio, ***p-MoD*** demonstrates the best overall performance on our comprehensive evaluation suite covering 15 benchmarks.

Model	DocVQA	ChartQA	SEED	GQA	AVG over **15** tasks*
LLaVA-NeXT-7B	70.1	61.6	68.9	63.5	63.5
+ SparseVLM [57]	67.2	52.8	68.1	62.6	62.3
+ VisionZip [49]	65.5	50.3	67.4	61.1	61.2
+ PyramidDrop [48]	66.5	53.3	67.5	61.9	61.9
+ FasterVLM [56]	65.9	47.7	68.0	61.4	61.4
+ FreeVideoLLM [12]	55.5	36.0	68.9	59.3	57.6
+ iLLaVA [15]	64.5	56.6	65.5	61.5	59.3
+ LLaVA-PruMerge [42]	58.0	44.6	67.4	61.2	60.0
+ ***p-MoD***	**70.0**	**61.8**	**69.0**	**63.3**	**63.4**

Table 10. **Comparison with more vision token compression methods.** In addition to the comparison in Table 5, we make a fair comparison between ***p-MoD*** and more vision token compression methods by controlling the average token retention ratio. Our methods achieves the best overall performance across 15 benchmarks. *Average is computed on all **15** benchmarks used in Table 1 and 2.

A.4. Comparison with γ-MoD

Concurrent to our work, γ-MoD [33] also propose to integrate Mixture-of-Depths mechanism into MLLMs. In this section, we first analyze the difference between our approach and theirs. Then we conduct experiments to show that our ***p-MoD*** approach outperforms γ-MoD.

The core design of γ-MoD is computing attention map (ARank) on some samples to identify which layers MoD can be applied to. In contrast, ***p-MoD*** can be effectively applied to **every** layer thanks to our TanhNorm and STRing modules. Furthermore, we propose **PRD** strategy to progressively reduce the token retention ratio layer by layer, which significantly boosts performance and efficiency.

In Table 11, we compare our method with γ-MoD on LLaVA-v1.5-7B baseline. γ-MoD utilizes higher token retention ratio than ***p-MoD*** (60+% vs 53%), but ***p-MoD*** still outperforms it by a large margin. Note that we are unable to compare both methods on LLaVA-NeXT, as γ-MoD's code implementation does not support LLaVA-NeXT.

Model	Keep Ratio↓	Doc VQA	Info VQA	Chart QA	Text VQA	RW QA	SE ED	PO PE	MM MU	AI 2D	VQA v2	OK VQA	MM E	G QA	S QA	MM B	AVG
LLaVA-v1.5	100%	28.1	25.8	18.2	46.0	55.6	66.2	85.9	36.6	55.2	76.6	53.4	1506.8	61.9	69.7	64.1	54.6
+ γ-MoD	$>60\%$	20.4	21.5	**18.1**	**47.5**	53.7	66.3	**86.6**	35.0	55.1	**77.1**	51.3	1,377.0	62.1	67.2	59.9	52.7
+ ***p-MoD***	**53.7%**	**27.6**	**26.8**	16.8	44.8	**55.7**	**66.5**	85.5	**36.3**	**56.2**	76.9	**56.0**	**1482.8**	**62.2**	**69.3**	**65.4**	**54.7**

Table 11. **Comparison with concurrent work γ-MoD. All models are of 7B parameter scale.**

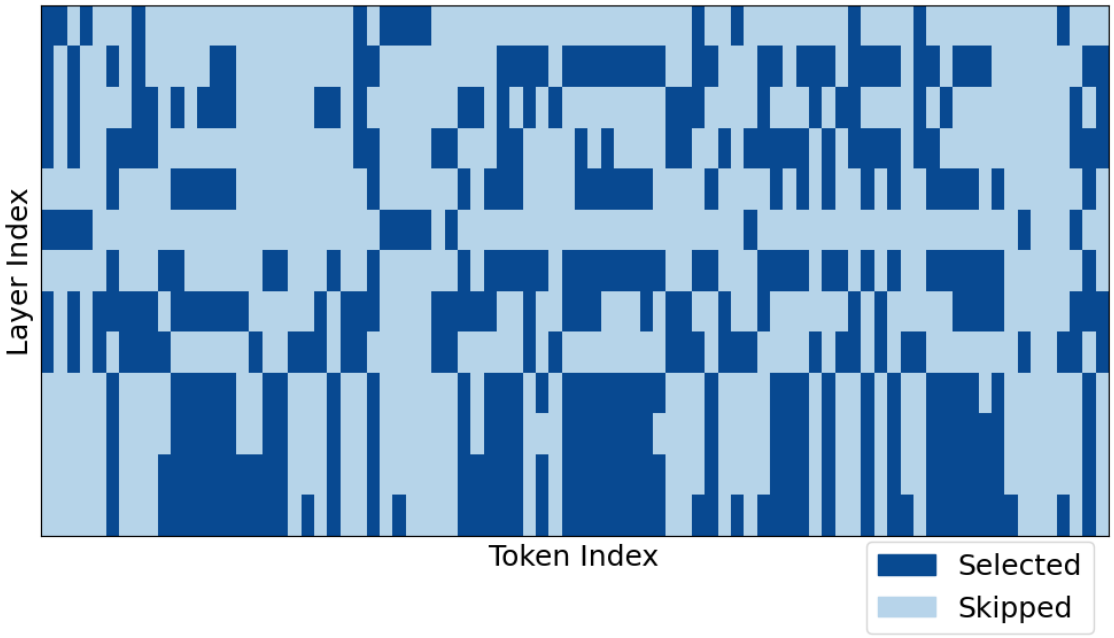

Figure 6. **Visualization of token selection decisions across different *p-MoD* layers.** The horizontal axis denotes the token indexes, and the vertical axis denotes the layer indexes. It can be observed that every ***p-MoD*** layer independently selects important and informative tokens.

B. Visualization of Token Selection Decisions

Figure 6 visualizes the token selection decisions of ***p-MoD*** across different layers. The horizontal axis denotes the token indexes, and the vertical axis denotes the layer indexes. It can be observed that every layer select different tokens to process, and every token is selected by different ***p-MoD*** layers. This demonstrates that every ***p-MoD*** layer independently selects important and informative tokens, without degrading into selecting a same set of tokens across different layers, which is identical to dropping tokens instead of layer-wise selection.

C. Further Discussions

C.1. Discussion on OCR Performance

In our experiments, we found that *all* token compression methods inevitably cause a significant performance drop on OCR-related benchmarks compared to other benchmarks. The main challenge we aim to address throughout the progress of this work is to mitigate this issue as much as possible. Experiment results in this paper show that our method achieves the *least* performance drop on OCR tasks compared to others.

C.2. Explanation on Training Overflow Problem

In Section 3.2.1, we mention that **TanhNorm** ensures training and inference stability. Acoordingly, we observe overflow issues in the ablation study on **TanhNorm** in Table 3. In this Section, we give a detailed explanation on the causes of overflow.

As illustrated in Equation 2, scaling a token by excessively large weights w_i across multiple LLM layers may cause **floating-point overflow**. For TanhNorm with $\alpha = 1$ (Table 3 Row 4), large w_i makes $X_i' = \alpha \tanh(w_i) T(X_i) + X_i$ approximate scaling the token X_i by 2. Repeating this for the same token across multiple layers causes overflow. The same problem arises for vanilla MoD (Table 3 Row 1). In this case, **no normalization** is applied to constrain the range of the weights, making it easier to produce extreme values and result in numerical overflow.

Name	Resolution	Train Stage	Trainable Module& Learning Rate	Data	Batch Size
p-MoD-LLaVA-v1.5	(336,336)	PT	Connector: 1e-3	558K	256
		SFT	Connector+LLM+MoD: 2e-5	665K	128
p-MoD-LLaVA-NeXT	336 × [(2,2), (1,2), (2,1), (1,3), (3,1)]	PT	Connector: 1e-3	558K	256
		SFT	ViT: 2e-6 (baseline) Connector+LLM+MoD: 2e-5	779K	128

Table 12. **Detailed training configuration**. PT stands for pre-training. SFT stands for supervised fine-tuning. During fine-tuning, the vision encoder of the baseline LLaVA-NeXT model is updated, consistent with the original LLaVA-NeXT model[28]. We freeze the vision encoder of our ***p-MoD***-LLaVA-NeXT model for all experiments to reduce training cost.

C.3. Limitations

One limitation of our work is that ***p-MoD*** is only experimented on LLaVA-1.5 and LLaVA-NeXT models, which focus on single-image understanding tasks. We believe that our approach has the potential to achieve more remarkable results when applied on tasks that handle a larger number of vision tokens, such as multi-image and long video understanding. We leave the exploration of ***p-MoD*** on other vision tasks to future research.

Another limitation is that ***p-MoD*** is a *trainable* method, designed for training a new MLLM from scratch with reduced training and inference costs. When being applied to trained MLLMs, it requires continual fine-tuning the model. It is not suitable for training-free scenarios.

D. More Implementation Details

D.1. Detailed Training Recipe

Our detailed training recipe of ***p-MoD*** models is shown in Table 12. LLaVA-1.5 employs a fixed input resolution of 336×336, while LLaVA-NeXT supports a pre-defined set of different resolutions (up to 672×672). Both the LLaVA-v1.5 models and the LLaVA-NeXT models go through the same pre-training stage, where the MLP connector module is trained on 558K image caption data [29] with a learning rate of 1e-3 and a batch size of 256.

During supervised fine-tuning, LLaVA-1.5 is trained on 665K instruction-tuning data [27], while LLaVA-NeXT is trained on a larger set of 779K data[*]. The vision encoder of the LLaVA-NeXT baseline model is updated for fine-tuning. For our ***p-MoD***-LLaVA-NeXT model, we freeze the vision encoder to save training time, as the available GPU resources are limited. When measuring the training GPU hours reported in Table 7, we freeze the vision encoder for both baseline and ***p-MoD*** models to ensure fair comparison.

[*]https://huggingface.co/datasets/lmms-lab/LLaVA-NeXT-Data

D.2. Hardware and Hyperparameters

We train all our models on 8 NVIDIA RTX A6000 GPUs. The inference efficiency metrics reported in Section 4.5 are measured on a single A6000 GPU. By default, we set the gating factor α in **TanhNorm** to 0.2, and the shift factor β in **PRD** to 0.5.

Due to computational constraints, we mainly use 7B models in our experiments. Experiments on 13B models are show in Table 8 and Section A.1.

D.3. Details on the PRD schedule.

We find it challenging for MoD layers to learn to predict meaningful weights when the token retention ratio is set to an extremely large or small value. To address this issue, we constrain the range of the token retention ratio R_l within a predefined maximum and minimum threshold. If R_l exceeds the maximum threshold, we set the token retention ratio to 1 so that the MoD module is not applied to the l-th layer. If R_l falls below the minimum threshold, the token retention ratio for the l-th layer is set to the minimum threshold:

$$R'_l = \begin{cases} 1, & \text{if} \quad R_l \geq max \\ R_l, & \text{if} \quad min < R_l < max \\ min, & \text{if} \quad R_l \leq min \end{cases} . \tag{6}$$