7. Supplemental Material

This supplemental data presents details from Tab. 7 in Sec. 4.3, and explaining limitations of our method mentioned in Sec. 5. First, in Sec. 7.1 we ablate over various pruning locations for our method. Second, in Sec. 7.2 we list the hyperparameters of Top-K and ToMe pruning methods used in evaluation with our method, in case others want to reproduce our work. Third, in Sec. 7.3 we show an example in which the accuracy-latency tradeoffs of our method become less significant at larger workload sizes.

7.1. Pruning location ablation study

In 3.2 we decide at which layer our pruning mechanism should be applied. To provide insight into the potential pruning locations, we performed an ablation study. Tab. 9 illustrates latency and accuracy tradeoffs for various pruning locations of DinoV2-G.

Batch Size	Pruning Layer	↓Acc. Loss	↓Median Latency (ms)
1	1/40	3.19	68.4
	10/40	1.07	81.3
	20/40	0.58	93.4
	30/40	0.49	104.4
2	1/40	7.08	79.1
	10/40	2.04	104.7
	20/40	0.97	133.0
	30/40	0.83	160.8

Table 9. Latency/accuracy tradeoffs by pruning location. Configuration: M = DinoV2-G on AGX Orin.

As expected, pruning earlier yields lower latency but greater accuracy degradation. For the batch size 1, our method pruned ∼54% of input tokens at the first layer degraded accuracy by 3.19% but yielded a 40% overall latency reduction. Across both batch size 1 and 2 in this ablation study, pruning after the first 25% of layers (layer 10) results in a good balance between latency reduction and accuracy degradation.

Pruning later in the network will reduce accuracy degradation, however we prioritize yielding latency benefits with our method. Therefore, we perform pruning at the layer 25% of the way into the network for all models evaluated in this work, as stated in Sec. 3.2.

7.2. Pruning Hyperparameters Used for Comparison with Other Work

In Tab. 7 we perform an experiment where we show the differences in accuracy of our method and others across models and devices. In Tab. 10 we present the same data annotated with an extra column for the hyperparameters of each method. Note that in both tables hyperparameters are chosen such that all methods achieve similar latency to our method.

Device	Batch Size	Model & Method	Acc. Loss	Median Latency (ms)
TX2	2	DeiT-S		68.92
		Top-K r=15	4.30	(-27.0%) 50.32
		ToMe r=17	2.47	(-23.1%) 52.98
		DyViT K=0.70	† 0.46	(-26.2%) 50.88
		Ours R=77	1.24	(-28.3%) 49.44
TX2	2	DeiT-B		215.0
		Top-K r=13	2.44	(-33.5%) 143.0
		ToMe r=12	1.63	(-33.7%) 142.7
		DyViT K=0.68	0.57	(-33.2%) 143.7
		Ours R=70	1.16	(-32.1%) 146.0
TX2	2	ViT-L		1327.0
		Top-K r=10	38.2	(-57.2%) 568.0
		ToMe r=10	17.5	(-57.8%) 559.3
		Ours R=133	8.4	(-55.2%) 594.9
Orin	4	ViT-L		70.29
		Top-K r=10	52.70	(-32.2%) 47.63
		ToMe r=15	17.51	(-22.9%) 54.18
		Ours R=101	2.35	(-33.0%) 47.08
Orin	2	DinoV2-G		155.5
		Top-K r=9	45.66	(-32.9%) 104.4
		ToMe r=7	6.96	(-32.9%) 104.4
		Ours R=166	2.04	(-33.1%) 104.1
A100	4	DinoV2-G		40.53
		Top-K r=8	13.31	(-16.7%) 33.76
		ToMe r=7	6.96	(-16.6%) 33.79
		Ours R=166	2.04	(-20.1%) 32.37

Table 10. Companion table to Tab. 7 with hyperparameters annotated for each entry of the original table. Top-K [12] and ToMe [1] remove r tokens each layer. R refers to the number of tokens our method prunes, and K is DynamicViT's keep ratio [34] (which they refer to as r in their work).

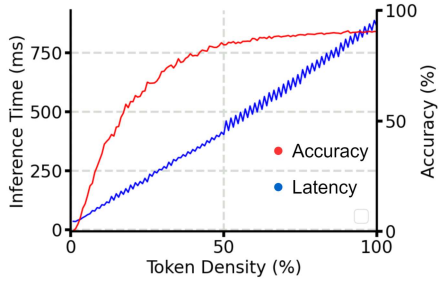

Figure 5. GPU Tail Effect has less impact on large batch size (here, the AGX Orin on DeIT-B with batch size of 128).

7.3. Large Workload Size Tradeoffs

In Sec. 5, we hypothesize that our method may achieve worse tradeoffs for larger workload sizes. Our method prioritizes pruning a number of tokens for which large latency changes occur. However, at larger workload sizes the latency-workload relationship becomes more linear. Fig. 5 depicts this phenomena for DeiT-B on the AGX Orin with batch size 128. It can be seen there are no large changes in latency to exploit, which is how our method is able to outperform other techniques like ToMe for small workload sizes.