A Implementation Details

Layer-level Merging Baselines. We used the official repositories for EViT (Liang et al., 2022a), ToMe (Bolya et al., 2022), and DTEM (Lee & Hong, 2024). Since implementations and experiments for ViT-L and ViT-H were not provided, we extended the code to include these two model configurations. Aside from adding the ViT-L and ViT-H variants, all experimental settings (training schedule, augmentations, optimizers, input resolutions, and other hyperparameters) were kept identical to the original baselines to ensure a fair comparison.

Image Classification. We implemented image classification models using the `timm` library Wightman (2019), leveraging its pretrained checkpoints. The ImageNet-1K dataset Deng et al. (2009) was used for training and evaluation, following prior works Havtorn et al. (2023); Ronen et al. (2023). For the full fine-tuning experiment, we follow the exact training recipe of MAE He et al. (2021), training VIT-B for 100 epochs and VIT-L for 50. We use a base learning rate of 1.5e-3 and use standard augmentations, namely RandAug Cubuk et al. (2020), Random Erasing Zhong et al. (2020), random flipping, and cropping. All training was done with 8 GPUs and used batch size 1024. We set layer decay to 0.75 during long fine-tuning. For short fine-tuning, we train the network for 1 epoch with layer decay set to 0.99, and learning rate set to 1e-6, and disable augmentations.

Visual QA. For our Visual Question Answering (VQA) experiments, we utilized the official LLaVA-1.5 Liu et al. (2024a) implementation and its pretrained checkpoints. Unlike the original approach, which collects data [cite] and fine-tunes the entire dataset for one epoch, we fine-tuned only 5% of the dataset, as we initialized from an already fine-tuned checkpoint. To adapt to this setting, we reduced the learning rate by a factor of 10 while following all other fine-tuning procedures recommended by LLaVA. The base image resolution was set to 336 with a patch size of 14, as specified in LLaVA's default configuration. A threshold of 5.75 was applied to determine a patch size of 28.

Object Detection. For object detection, we employed the official EVA-02 Fang et al. (2024) implementation along with its pretrained checkpoints, which utilize a window attention mechanism. Fine-tuning was conducted following the recommended procedures outlined in EVA-02. Consistent with our previous experiments, we fine-tuned for 5% of the total iterations while reducing the learning rate by a factor of 10. Following EVA-02's settings, the image resolution was 1536 with a patch size of 16. Patch sizes of 128, 64, and 32 were determined based on threshold values of 0.3, 2, and 2, respectively.

Semantic Segmentation. We also utilized the official EVA-02 implementation along with its pretrained checkpoints for semantic segmentation. The ADE20K dataset Zhou et al. (2019; 2017) was used for training and evaluation. Fine-tuning followed the recommended procedures outlined in EVA-02. In alignment with our previous experiments, we fine-tuned for 5% of the total iterations while reducing the learning rate by a factor of 10. According to EVA-02's settings, the image resolution was either 512 or 640, with a patch size of 16. A threshold of 5.75 was applied.

B Hardware Setup

All ImageNet experiments were conducted on a node of 8x NVIDIA A100s, and the experiments on object detection, segmentation and visual QA were conducted with 8xNVIDIA RTX A6000. The inference-time results were computed on a single GPU, along with the throughput and FLOPS analysis. We used a single node for all work on this paper.

C Additional Results

We provide additional visualizations to illustrate how APT (Adaptive Patch Token) prunes tokens and to analyze the qualitative effects of varying the difference threshold τ, augmentation, and scorers. All visualizations were conducted using images at a resolution of 336×336 and a patch size of 14×14.

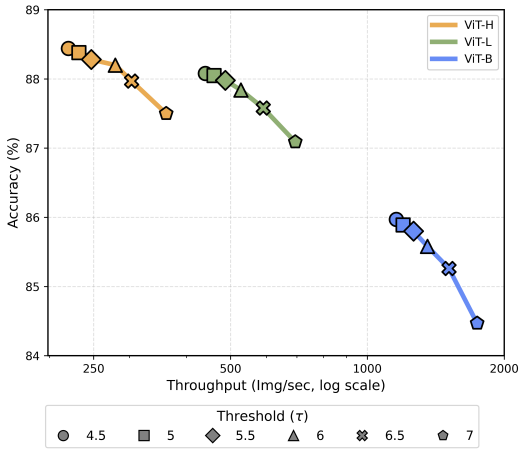

Figure 6: **Threshold Effect.** Increasing the threshold increases throughput significantly, but after approximately $\tau = 5.5$, the accuracy begins to severely drop off, and is not 'fixable' with fine-tuning.

Threshold Analysis. The main tunable parameter in APT is the entropy threshold, which can differ per scale and controls how compressible a region must be in order to be retained. Lower values indicate higher sensitivity, and for the vast majority of experiments in this paper, we used $\tau_1 = 5.5, \tau_2 = 4.0$. In Figure 6, we vary τ_1 for 3 model scales with resolution 336 and patch size 14, measuring ImageNet accuracy. We observe that for threshold values larger than 6.0, accuracy drops off significantly, while throughput continues to increase. We find that 5.5 offers a good trade-off between acceleration and maintaining quality, and hypothesize that this is close to the 'true' threshold for compressibility; beyond this point, coarse-scale patches result in information loss. Figure 7 shows a diverse set of sample images and how our method prunes tokens with relatively lower amounts of information (e.g., background regions or uniform color patches). We fixed $\tau_2 = 4.0$, and changed τ_1 from 4.5 to 7. By observing various categories of images, one can see that patches containing high-frequency details or salient object features are consistently preserved. In contrast, less critical regions—such as large uniform areas—are pruned. This visualization confirms that the model potentially increasing efficiency by ignoring parts of the image that contribute less to the downstream task.

Augmentation Analysis. We compare how APT operates under different data augmentation techniques in Figure 8. Notably, random erasing removes parts of the image, causing the overall information to be reduced from the outset. As a result, the total number of retained tokens also decreases because many regions lose their distinguishing features. This phenomenon implies that the speed-up gain could be higher during training or fine-tuning—when augmentations are applied repeatedly—than during inference.

Scorer Analysis. Figure 9 contrasts the results of an entropy-based scorer with a Laplacian-based scorer. The entropy-based scorer measures how diverse or complex the pixel-value distribution is within a patch. If a patch has pixels with a wide range of intensities or colors, it scores higher and is more likely to be retained. This approach naturally favors regions with intricate textures, multiple color transitions, or high levels of detail. In comparison, the Laplacian-based scorer uses a second-derivative operator (or second-order difference) to detect edges or sharp transitions. Specifically, it looks at how abruptly the pixel intensity changes within a patch. As a result, if there is a strong boundary or a sharp difference in color or brightness, the Laplacian score becomes high, signaling that the patch likely contains important edge information and should be preserved. We generally found that the Laplacian scorer performed slightly worse than the entropy one, likely because it makes stronger assumptions about what is and is not important to the downstream model - the entropy scorer has less image related inductive bias. When controlling for fraction of reduced tokens, the Laplacian scorer consistently performed about 0.2-0.3% worse on ImageNet classification.

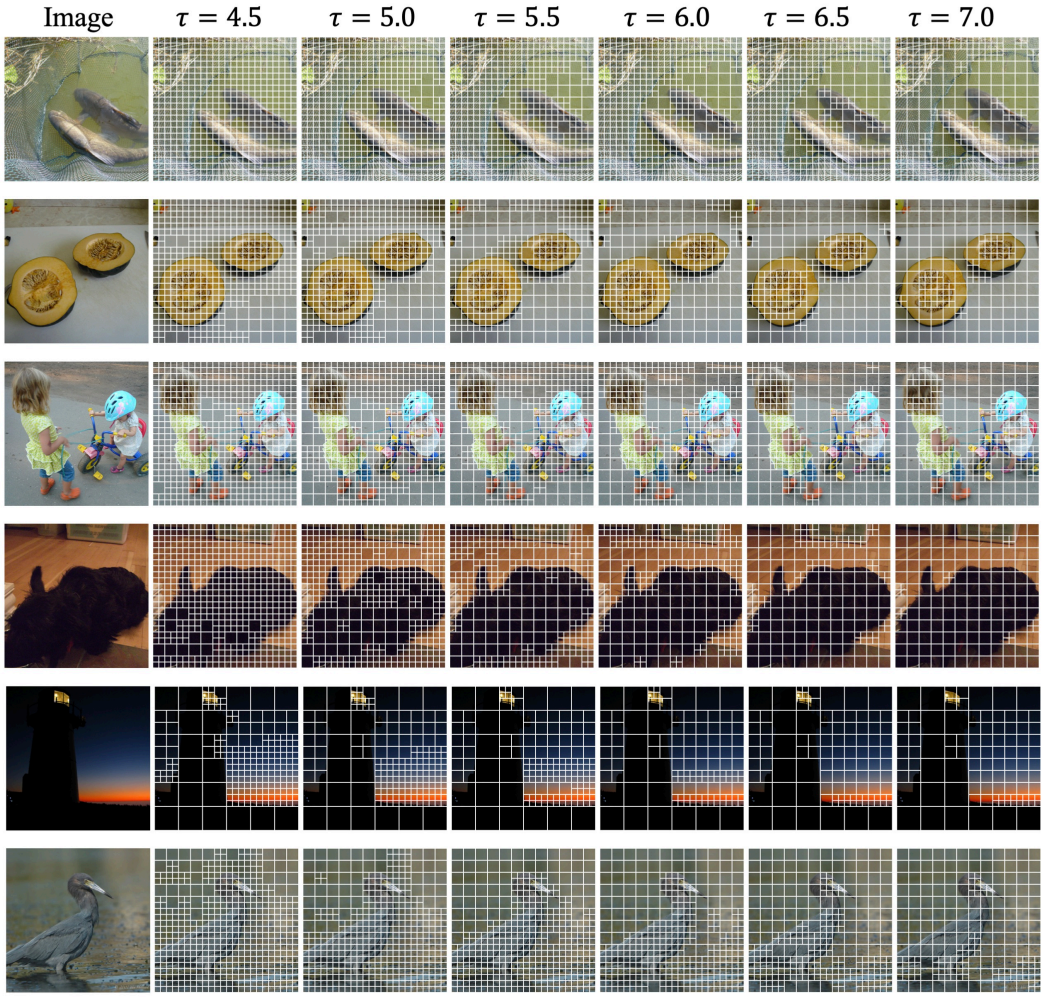

Figure 7: **Threshold visualization.** We can see that patches containing high-frequency details or salient object features are consistently preserved under various thresholds. We used $\tau = 5.5$ for most of the experiments. Zoom in for the best view.

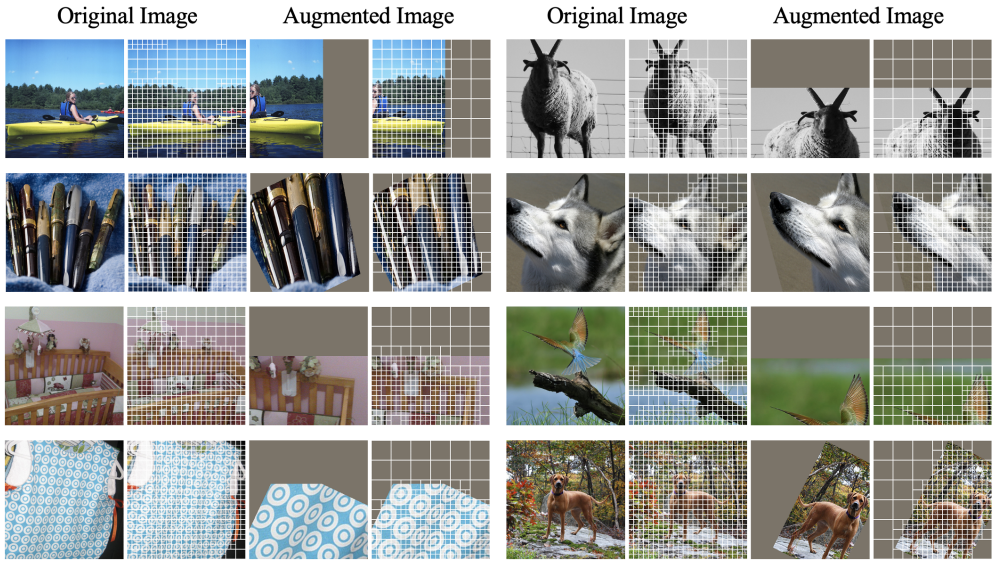

Figure 8: **Augmentation visualization.** We observe that augmentations generally lead to *fewer* tokens. In particular, Random Erasing Zhong et al. (2020), leads to regions that can be tokenized with the large patch sizes, significantly increasing throughput compared to inference time.

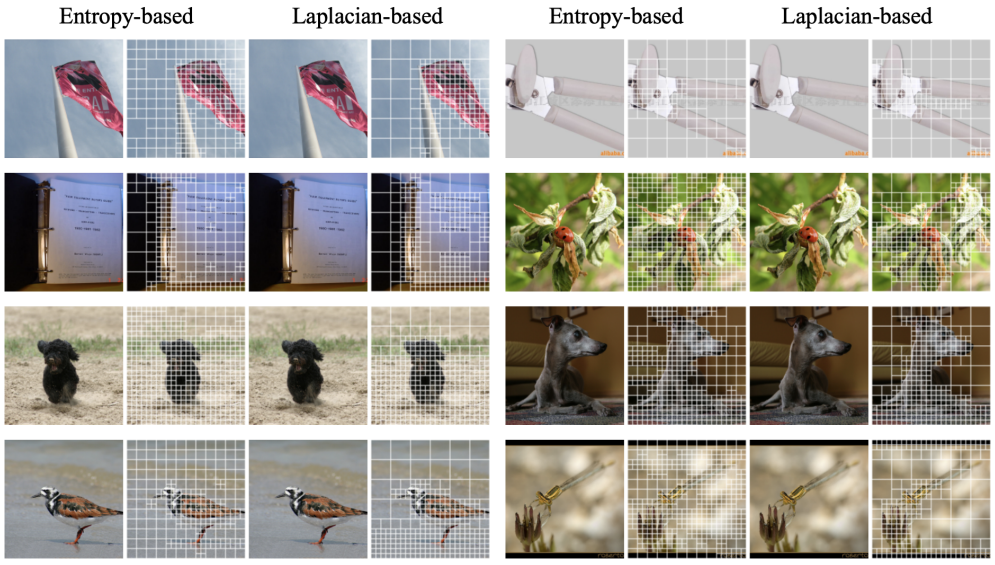

Figure 9: **Scorer visualization.** Using a Laplacian scorer tends to place larger patches outside object boundaries and smaller ones nearer edges and high frequency details, as it takes an approximation of the image structure into account. On the other hand, the entropy scorer simply uses the distribution of intensities in the patch to measure its compressibility.