Training Vision Transformers Faster: Understanding Token Dropping, Overfitting, and the Role of Regularization

Aumkesh Chaudhary
Indian Institute of Technology, Patna
aumkeshchaudhary@gmail.com

Abstract

Vision Transformers (ViTs) have demonstrated competitive performance on image recognition benchmarks but remain computationally expensive due to the uniform processing of all image tokens. In this report, we investigate Progressive Token Drop (PTD), a training strategy that progressively reduces token attendance based on attention saliency. Unlike pruning-based approaches that target inference-time acceleration, PTD reduces compute only during training through a token curriculum, leaving the model architecture and inference unchanged.

We train two hybrid models on CIFAR-100: a baseline using a convolutional stem followed by a 6-block ViT backbone (192-dim embedding, 6 heads), and a PTD variant that modifies one transformer block to drop low-saliency tokens according to an epoch-dependent curriculum. Our ablation reveals that PTD alone reduces validation accuracy from 71.59% to 68.83% due to increased overfitting despite faster training. However, when combined with strong data augmentation (Mixup, CutMix) and EMA (Exponential Moving Average) of weights, PTD achieves 74.12% validation accuracy—a 2.53pp improvement over baseline—while reducing training time by 25%.

Our results demonstrate that token reduction strategies require careful regularization to be effective, and that the combination yields both efficiency and accuracy gains. Importantly, the method leaves inference-time computation unchanged, since all tokens are retained at test time.

1 Introduction

Vision Transformers (ViTs) operate on sequences of image patches and apply self-attention uniformly across all tokens, regardless of their semantic contribution. On low-resolution datasets such as CIFAR-100, this can be redundant: many patches carry background or low-information content but are processed at the same cost as highly informative regions.

Token pruning and dynamic token routing methods attempt to address this by dropping or merging tokens [1, 2, 3, 4]. However, most of these approaches either alter the model architecture, introduce nontrivial routing modules, or tightly couple pruning with inference-time constraints. They also often require careful retraining and can suffer from significant accuracy degradation when aggressive token reduction is used.

In this work, we implement and study a training-time Progressive Token Drop (PTD) mechanism on top of a compact hybrid ViT baseline. Rather than permanently pruning tokens, PTD progressively drops low-saliency tokens only during training. Early epochs expose the full token set, while later epochs focus the model on a subset of tokens ranked by attention-based saliency. At test time, all tokens are retained and the forward pass is identical to the baseline.

Concretely, we compare:

- **Baseline ViT:** A convolutional patch embedding layer followed by a 6-layer ViT with dropout and stochastic depth, trained with standard data augmentation.
- **PTD ViT:** The same backbone, but with a single transformer block modified to perform token dropping using an epoch-dependent curriculum. This variant also uses Mixup, CutMix, and EMA of weights, with slightly softer pixel-level augmentations.

The analysis focuses on training dynamics, accuracy progression, and wall-clock training cost derived from logged metrics.

2 Method

2.1 Baseline Hybrid ViT Architecture

Both baseline and PTD models share the same backbone:

- **Conv patch embed:** A 3-layer convolutional stem maps input images $x \in \mathbb{R}^{3\times32\times32}$ to an embedding tensor $z \in \mathbb{R}^{E\times16\times16}$ with $E = 192$. This corresponds to $N = 16 \times 16 = 256$ patch tokens.
- **Tokenization:** The feature map is flattened into N tokens and concatenated with a learnable class token:
$$X_0 = [x_{\text{cls}}; z_1; \ldots; z_N] \in \mathbb{R}^{(N+1)\times E}.$$
- **Positional encoding:** A learnable positional embedding $P \in \mathbb{R}^{(N+1)\times E}$ is added to X_0, followed by dropout.
- **Transformer encoder:** A stack of $L = 6$ transformer blocks with embedding dimension $E = 192$, $H = 6$ attention heads, MLP expansion ratio 4.0, dropout $p = 0.1$, and stochastic depth with a linearly increasing drop-path rate up to 0.1.
- **Head:** The final class token is layer-normalized and passed through a linear classifier to produce 100-way logits.

Each baseline transformer block has the standard structure:

$$Y = X + \text{DropPath}(\text{MHA}(\text{LN}(X))), \tag{1}$$
$$Z = Y + \text{DropPath}(\text{MLP}(\text{LN}(Y))), \tag{2}$$

where MHA denotes multi-head self-attention, MLP is a 2-layer feed-forward network with GELU nonlinearity, and DropPath implements stochastic depth.

2.2 PTD Block with Token Drop

The PTD model modifies one transformer block to enable optional dropping of low-saliency patch tokens during training. The key components are:

Attention-based saliency. Given normalized attention weights from that block

$$A^{(h)} \in \mathbb{R}^{(N+1)\times(N+1)} \quad \text{for head } h = 1, \ldots, H,$$

we compute a scalar saliency for each patch token $i \in \{1, \ldots, N\}$ as the mean attention it receives from the class token, averaged over heads:

$$s_i = \frac{1}{H} \sum_{h=1}^{H} A^{(h)}_{0,i}.$$

Here index 0 denotes the class token and indices $1, \ldots, N$ correspond to patch tokens.

Progressive token drop schedule. Let t be the current epoch, and let t_s and t_e be the start and end epochs of the curriculum. The drop ratio $d(t)$ is defined by:

$$d(t) = \begin{cases} d_{\text{init}}, & t < t_s, \\ d_{\text{final}}, & t \geq t_e, \\ d_{\text{init}} + \left(\dfrac{t - t_s}{t_e - t_s}\right)^{\gamma} (d_{\text{final}} - d_{\text{init}}), & \text{otherwise}, \end{cases}$$

where:

- $d_{\text{init}} = 0.0$ is the initial drop rate,
- $d_{\text{final}} = 0.35$ is the final drop rate,
- $t_s = 40$ and $t_e = 140$,
- $\gamma = 2.0$ controls the shape of the curriculum (slower increase at the beginning, more aggressive later).

At epoch t, we keep the top-k tokens according to s_i, where

$$k = \lfloor (1 - d(t)) \cdot N \rfloor .$$

Forward path with PTD. Within the PTD block, the forward pass behaves as follows during training when PTD is enabled:

1. Compute self-attention outputs and weights with the modified attention module that can return A.
2. If $d(t) \approx 0$ or we are outside the curriculum window, fall back to standard block behaviour.
3. Otherwise, compute saliency scores $\{s_i\}$, select top-k patch tokens per sample, and reassemble a reduced sequence:

   $$X' = [x_{\text{cls}}; x_{i_1}; \ldots; x_{i_k}],$$

   where $\{i_j\}$ are the indices of retained tokens.
4. Apply the residual update with this reduced sequence and continue with the MLP sublayer.

Importantly, PTD operates *only* during training and only in a single chosen block. During evaluation, the model runs with the full token set and behaves identically to the baseline backbone.

2.3 Regularization: Mixup, CutMix and EMA

The PTD variant supplements token dropping with label-level and weight-level regularization:

- **Mixup & CutMix:** For each mini-batch, the code samples either Mixup or CutMix. Mixup blends images and labels with $\alpha_{\text{mixup}} = 0.8$, while CutMix uses patched image regions with $\alpha_{\text{cutmix}} = 1.0$ and probability $p_{\text{cutmix}} = 0.5$. The loss is computed as a convex combination between two labels.
- **EMA:** An Exponential Moving Average of the model parameters is maintained with decay $\beta = 0.9999$. Validation metrics are reported both for the raw weights and EMA weights, and the best model checkpoint is selected based on EMA validation accuracy.

3 Experimental Setup

3.1 Dataset and Preprocessing

All experiments are conducted on CIFAR-100:

- 50,000 training images and 10,000 test images.
- Image size: 32×32, 3 channels.

Pixel values are normalized using the standard CIFAR-100 statistics:

$$\text{mean} = (0.5071, 0.4867, 0.4408), \quad \text{std} = (0.2675, 0.2565, 0.2761).$$

Baseline data augmentation. The baseline setup applies:

- Random crop with padding 4 (on 32×32),
- Random horizontal flip,
- `RandAugment` with 2 operations and magnitude 10,
- Random erasing with probability 0.25 and random filling,
- Normalization with the above mean and std.

PTD data augmentation. The PTD setup adopts slightly softer pixel-level augmentation:

- Same random crop and flip,
- `RandAugment` with magnitude 7,
- Random erasing with probability 0.1,
- The same normalization,
- Additional label-level Mixup/CutMix as described earlier.

3.2 Training Configuration

For both experiments:

- **Backbone:** Conv stem + 6-layer ViT with $E = 192$, $H = 6$, MLP ratio 4.0.
- **Optimizer:** AdamW with weight decay 0.05.
- **Learning rate:** Initial LR 4×10^{-4} with 5-epoch warmup.
- **Schedule:** Cosine decay implemented as a LambdaLR scheduler.
- **Label smoothing:** 0.05 for the baseline, 0.03 for PTD.
- **Batch size:** 128 for training, 256 for evaluation.
- **Epochs:** 200 for both models.
- **Gradient clipping:** Global norm clipping at 1.0.
- **Hardware:** Apple M2 Pro GPU via PyTorch's MPS backend.

The baseline uses standard dropout $p = 0.1$ and drop-path up to 0.1, whereas the PTD variant uses no dropout ($p = 0.0$) and a milder drop-path of 0.05, relying more on Mixup/CutMix and EMA for regularization.

3.3 Timing Measurement

Training time is computed by aggregating per-epoch durations recorded in the logs. For the baseline, epoch durations are extracted from the `tqdm` progress output (e.g., `[MM:SS<00:00, it/s]`) and converted to seconds before summation across all epochs. For the PTD model, total training time is obtained using the same aggregation procedure. All reported values correspond to wall-clock time measured on the same hardware under otherwise identical settings.

4 Results

4.1 Quantitative Comparison

Table 1 summarizes the main metrics aggregated from the training logs and the CSV files generated by the notebooks.

Method	Train Acc	Val Acc	Time	Speedup
Baseline (no PTD, no aug)	92.4%	71.59%	13h 40m	1.0×
PTD only (no aug, no EMA)	97.9%	68.83%	10h 20m	1.3×
PTD + Mixup/CutMix + EMA	53.5%	**74.12%**	10h 20m	**1.3**×

Table 1: Performance and compute comparison on CIFAR-100. The training accuracy column reveals that PTD alone increases overfitting (97.9% train vs 68.83% val), while PTD combined with strong regularization achieves the best validation accuracy despite low training accuracy (53.5%) due to soft labels from Mixup/CutMix.

4.2 Ablation Study: PTD Alone Increases Overfitting

To isolate the effect of Progressive Token Drop (PTD), we trained a variant that applied only the token curriculum while keeping all other settings identical to the baseline—no Mixup, no CutMix, and no EMA of weights.

Table 1 shows that this PTD-only configuration reaches a training accuracy of 97.9% but only 68.83% validation accuracy, which is a **2.76-percentage-point drop** relative to the baseline (71.59%). The train-validation gap increases from −20.81% (baseline) to −29.07% (PTD only), indicating severe overfitting.

This result reveals a counterintuitive finding: PTD alone not only fails to improve performance but actually decreases validation accuracy despite reducing training time. Why does reducing token count lead to worse overfitting? We hypothesize several mechanisms:

Token compression and memorization. When PTD removes low-saliency tokens, the remaining tokens must encode all necessary information for classification. Rather than learning distributed, generalizable representations, the model may memorize training-specific patterns in the retained high-saliency regions. This is analogous to extreme bottleneck layers: too much compression can lead to memorization rather than abstraction.

Attention feedback loops. Our saliency metric is based on class-to-patch attention weights. Tokens that receive high attention are retained, while low-attention tokens are dropped. This creates a positive feedback mechanism: retained tokens become increasingly important to the model, receive even more attention in subsequent epochs, and dominate the learned representation. Dropped tokens never recover, preventing the model from exploring alternative feature combinations that might generalize better.

Capacity-regularization mismatch. Classical wisdom suggests that reducing capacity acts as implicit regularization. However, PTD does not uniformly reduce capacity—it selectively prunes the token dimension while keeping all other parameters fixed. This asymmetric reduction may fail to provide regularization benefits and instead create a pathological training dynamic where the model overfits to the token selection strategy itself.

This ablation demonstrates that **PTD requires strong regularization to be effective**. When paired with Mixup, CutMix, and EMA, the model transitions from harmful overspecialization to improved generalization. The extreme training accuracy drop to 53.5% (due to soft labels from data augmentation) is the price paid for forcing the model to learn robust features despite reduced token capacity.

4.3 Full Model Performance: Accuracy and Loss Curves

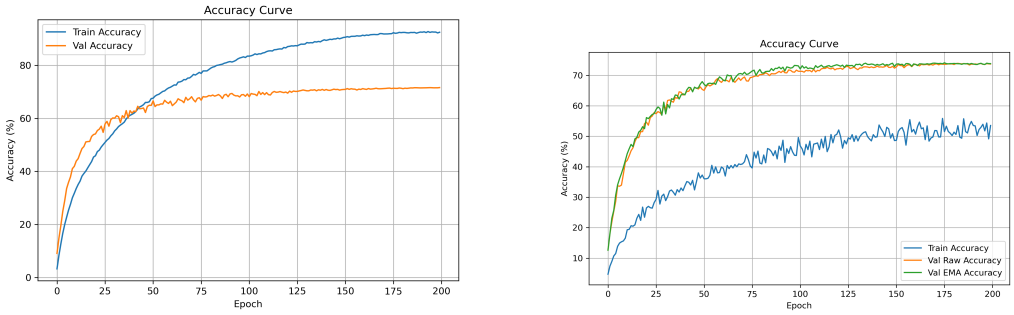

Figure 1: Training and validation accuracy curves on CIFAR-100 for the baseline ViT (left) and PTD+Mixup+EMA ViT (right). The PTD model shows substantially lower training accuracy (53.5%) due to heavy regularization (Mixup, CutMix, and EMA creating soft labels), yet reaches a higher peak validation accuracy (74.12%) than the baseline, indicating stronger generalization.

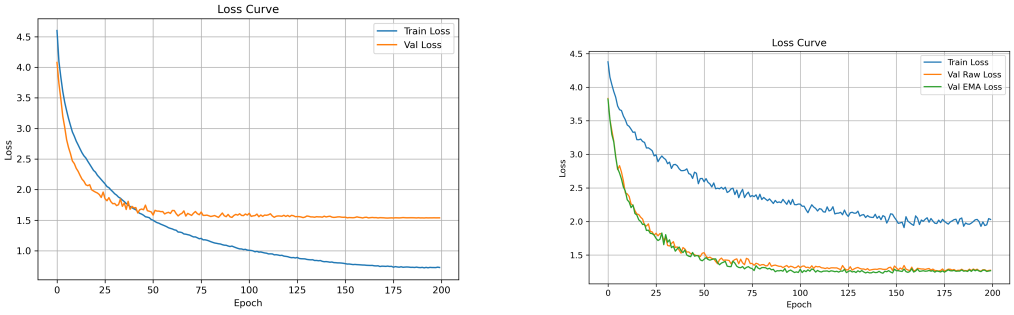

Figure 2: Training and validation loss curves for the baseline (left) and PTD model (right). PTD increases the train–validation gap due to strong regularization but ultimately achieves a lower final validation loss, indicating better generalization.

4.4 Learning Rate Schedules

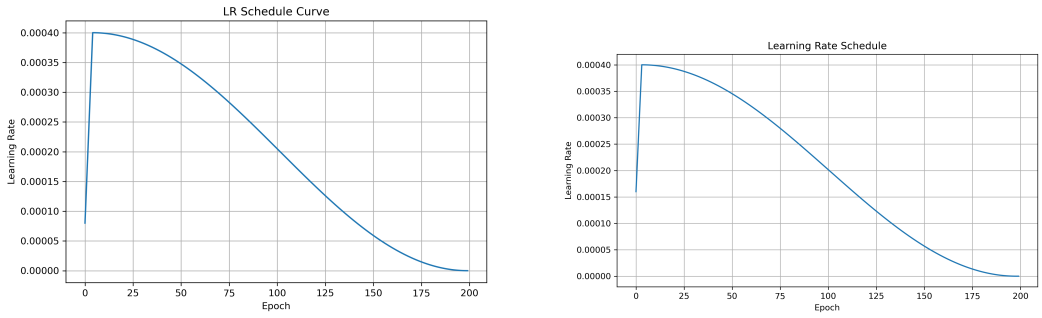

Figure 3: Cosine learning rate schedules used in both experiments. The PTD notebook applies the schedule via an epoch-based `LambdaLR`, while the baseline uses a step-based `LambdaLR`; both approximate the same warmup+cosine policy over 200 epochs.

4.5 Key Observations

Analysis of the training and validation behavior reveals the following observations:

- PTD with Mixup/CutMix and EMA outperforms the baseline, improving top-1 validation accuracy from 71.59% to 74.12% despite dropping up to 35% of tokens in one transformer block during mid-to-late training.
- The PTD model trains substantially faster in wall-clock time (around 1.3× speedup), reducing total training time from 13h 40m to 10h 20m on the same hardware.
- PTD alone (without augmentation) harms performance, decreasing validation accuracy to 68.83% and increasing overfitting (97.9% train accuracy). This demonstrates that token reduction without proper regularization is detrimental.
- Due to strong regularization (Mixup, CutMix, and EMA), the training accuracy of the full PTD model remains far lower (53.5%) than its validation accuracy, resulting in a larger apparent train–validation gap—yet this correlates with improved generalization rather than underfitting.
- EMA consistently boosts validation accuracy over raw weights; the best PTD result reported (74.12%) uses EMA validation metrics.

5 Discussion

5.1 Accuracy vs Efficiency Trade-off

PTD in this implementation delivers both improved training efficiency and higher model accuracy when combined with appropriate regularization. The baseline hybrid ViT reaches 71.59% validation accuracy, while the PTD+Mixup+CutMix+EMA variant achieves 74.12%, corresponding to a 2.53-percentage-point gain. Thus, the full system does not merely trade accuracy for speed—it provides a favorable accuracy–efficiency Pareto improvement.

However, this improvement is not due to PTD alone. Our ablation shows that PTD without augmentation decreases accuracy to 68.83%, indicating that the gains come from the synergistic combination of token reduction with heavy data augmentation and weight averaging.

Unlike pruning-oriented methods, PTD does not alter the inference-time architecture: all tokens are retained at test time. As a result, inference latency and FLOPs remain effectively

identical between the baseline and PTD variants, with savings realized exclusively during training.

5.2 Effect of Augmentation and EMA

The PTD experiment combines multiple factors beyond token dropping:

- Softer pixel-level augmentation (lighter RandAugment, lower random erasing probability),
- Stronger label-level augmentation (Mixup and CutMix),
- EMA of weights used for evaluation.

Our ablation reveals that PTD alone significantly harms validation accuracy (68.83% vs 71.59% baseline), demonstrating that token reduction without proper regularization increases overfitting. The 74.12% accuracy of the full system is primarily due to Mixup/CutMix/EMA, with PTD contributing training efficiency.

Future work should investigate whether augmentation can be applied to the baseline for a more controlled comparison, though we note that the extreme training accuracy drop (53.5%) suggests PTD may interact uniquely with heavy augmentation—the reduced token capacity may force the model to learn more robust features when trained on heavily augmented data.

5.3 Why PTD Alone Increases Overfitting

A central finding of this work is that Progressive Token Drop, despite reducing model capacity by dropping up to 35% of tokens, paradoxically increases overfitting rather than acting as a regularizer. We identify several mechanisms that may explain this counterintuitive behavior:

Token compression and memorization. When PTD removes low-saliency tokens, the remaining tokens must encode all necessary information for classification. Rather than learning distributed, generalizable representations, the model may memorize training-specific patterns in the retained high-saliency regions. This is analogous to extreme bottleneck layers in autoencoders: too much compression can lead to memorization rather than abstraction.

Attention feedback loops. Our saliency metric is based on class-to-patch attention weights. Tokens that receive high attention are retained, while low-attention tokens are dropped. This creates a positive feedback mechanism: retained tokens become increasingly important to the model, receive even more attention in subsequent epochs, and dominate the learned representation. Dropped tokens never recover, preventing the model from exploring alternative feature combinations that might generalize better.

Capacity-regularization mismatch. Classical wisdom suggests that reducing capacity (e.g., fewer parameters, smaller models) acts as implicit regularization. However, PTD does not uniformly reduce capacity—it selectively prunes the token dimension while keeping all other parameters fixed. This asymmetric reduction may fail to provide regularization benefits and instead create a pathological training dynamic where the model overfits to the token selection strategy itself.

Information bottleneck collapse. From an information-theoretic perspective, PTD creates a progressive information bottleneck. If this bottleneck is too aggressive without compensating regularization, the model may collapse to a degenerate solution that perfectly fits the training data's high-saliency regions but fails to capture the diversity needed for generalization.

Implications for token reduction methods. Our findings suggest that token reduction strategies—whether via dropping, merging, or pruning—cannot be treated as drop-in efficiency improvements. They fundamentally alter the model's learning dynamics and must be paired with strong regularization techniques (data augmentation, label smoothing, weight averaging) to prevent overfitting. Future work on efficient transformers should carefully ablate token reduction mechanisms in isolation to verify their impact on generalization before combining them with other training improvements.

5.4 Saliency Metric and Curriculum Design

The current saliency metric uses averaged class-to-patch attention. While intuitive and easy to implement, attention weights do not always correlate with semantic importance. Alternative token importance measures (e.g. gradient-based saliency, information gain, or learned importance predictors) remain unexplored.

Similarly, the schedule parameters $(t_s, t_e, d_{\text{final}}, \gamma)$ are set manually and are not tuned via systematic ablation. A more thorough study could explore different curriculum shapes, multiple PTD blocks, or token reinstatement strategies.

5.5 Limitations

This implementation has several limitations:

- Results are limited to a single dataset (CIFAR-100) and a single compact ViT backbone.
- No multi-run statistics (e.g. mean $\pm$ std over multiple seeds) are reported due to computational constraints.
- Inference-time efficiency is unchanged; the method is not a compression or deployment optimization technique.
- PTD is applied in only one block; more aggressive designs may yield larger speedups but could impair stability.
- The baseline was not tested with Mixup/CutMix/EMA to provide a fully controlled comparison isolating PTD's contribution independent of augmentation.

6 Conclusion

This work investigated Progressive Token Drop (PTD) as a training strategy for Vision Transformers on CIFAR-100. Our ablation study reveals that PTD alone significantly harms generalization, reducing validation accuracy from 71.59% to 68.83% due to increased overfitting. However, when combined with strong data augmentation (Mixup and CutMix) and Exponential Moving Average (EMA) of weights, PTD achieves both improved validation accuracy (74.12%, a 2.53pp gain over baseline) and faster training (10h 20m vs 13h 40m, a 1.3× speedup).

Our key finding is that token reduction strategies require careful regularization to be effective. PTD's training-time efficiency gains come at the cost of reduced model capacity, which must be compensated by preventing overfitting through heavy augmentation. Because PTD modifies only the training phase and leaves the architecture and token count unchanged at inference, it offers a practical way to make ViT training more compute-efficient when paired with appropriate regularization, without sacrificing performance or deployment simplicity.

The combination of PTD with strong data augmentation demonstrates that efficiency and accuracy need not be traded off—with the right training recipe, both can be improved simultaneously.

6.1 Future Work

Our results open several research directions:

- **Controlled ablation:** Apply Mixup/CutMix/EMA to the baseline (without PTD) to isolate PTD's contribution independent of augmentation effects.
- **Augmentation sensitivity:** Systematically vary mixup/cutmix strength to identify the boundary where PTD helps versus hurts performance, and determine the minimal regularization needed for PTD to be beneficial.
- **Alternative regularization:** Test whether other regularization techniques (stronger dropout, increased label smoothing, or different augmentation strategies) can replace Mixup/CutMix while preserving PTD benefits.
- **Saliency feedback analysis:** Investigate whether the attention-based saliency metric creates pathological token selection patterns or feedback loops that contribute to overfitting.
- **Multi-block PTD:** Our single-block design is conservative; studying multi-block PTD with appropriate regularization may yield larger speedups, though stability and convergence would need careful analysis.
- **Larger-scale validation:** Validate PTD with strong augmentation on larger datasets (ImageNet-1K, ImageNet-21K) and different ViT architectures to assess generalizability beyond CIFAR-100.
- **Inference-time efficiency:** Explore combining PTD's training-time curriculum with inference-time pruning or token merging to obtain end-to-end speedups without the need for retraining.

Overall, this work demonstrates that token curricula can be integrated into standard PyTorch ViT pipelines with minimal changes, but they require careful pairing with strong regularization to deliver both accuracy and efficiency improvements during training.

References

[1] Rao, Y. et al. DynamicViT: Efficient Vision Transformers with Dynamic Token Sparsification. In *ICCV*, 2021.

[2] Xu, Z. et al. Evo-ViT: Slow-Fast Token Evolution for Dynamic Vision Transformer. In *NeurIPS*, 2021.

[3] Bolya, D. et al. Token Merging: Your ViT But Faster. In *CVPR*, 2023.

[4] Liu, Y., et al. PatchDropout: Economizing Vision Transformers Using Patch Dropout. In *WACV*, 2023.