ViT-AdaLA: Adapting Vision Transformers with Linear Attention

Yifan Li[1*] **Seunghyun Yoon**[2] **Viet Dac Lai**[2] **Franck Dernoncourt**[2]
Jason Kuen[2] **Yu Kong**[1] **Trung Bui**[2]
[1]Michigan State University, {liyifa11, yukong}@msu.edu
[2]Adobe Research, {bui, dernoco, kuen, daclai}@adobe.com

Abstract

Vision Transformers (ViTs) based vision foundation models (VFMs) have achieved remarkable performance across diverse vision tasks, but suffer from quadratic complexity that limits scalability to long sequences. Existing linear attention approaches for ViTs are typically trained from scratch, requiring substantial computational resources, while linearization-based methods developed for large language model decoders do not transfer well to ViTs. To address these challenges, we propose **ViT-AdaLA**, a novel framework for effectively adapting and transferring prior knowledge from VFMs to linear attention ViTs. ViT-AdaLA consists of three stages: attention alignment, feature alignment, and supervised fine-tuning. In the attention alignment stage, we align vanilla linear attention with the original softmax-based attention in each block to approximate the behavior of softmax attention. However, residual approximation errors inevitably accumulate across layers. We mitigate this by fine-tuning the linearized ViT to align its final-layer features with a frozen softmax VFM teacher. Finally, the adapted prior knowledge is transferred to downstream tasks through supervised fine-tuning. Extensive experiments on classification and segmentation tasks demonstrate the effectiveness and generality of ViT-AdaLA over various state-of-the-art linear attention counterpart. Project page is available: https://jackyfl.github.io/vitadala-project-page.

1 Introduction

Vision Transformers (ViTs) [18] based vision foundation models (VFMs) such as DINOv2 [39] and CLIP [41] have been widely adopted across a broad range of computer vision tasks [33], including segmentation, detection, visual question answering (VQA), depth estimation, and image, video, and 3D point-cloud generation. However, the standard softmax-based self-attention in ViTs scales quadratically with the number of visual tokens, leading to substantial computational and memory overhead as the sequence length increases as shown in Fig. 1. This limitation becomes increasingly acute as modern vision applications demand processing long-sequence visual tokens.

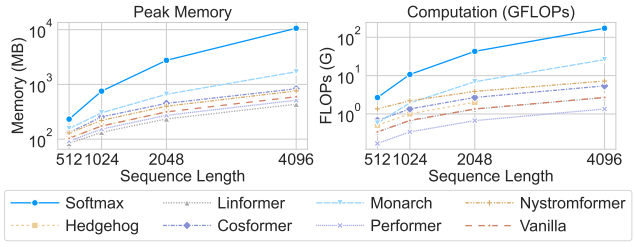
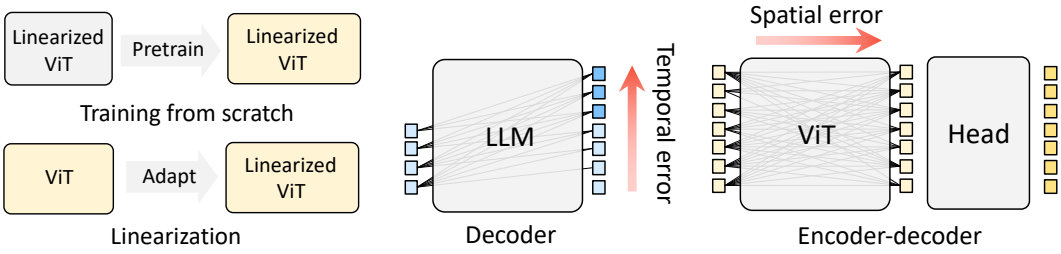


Figure 1: Efficiency comparison of different attentions, including peak memory and GFLOPS varying with sequence length. Only attention module is benchmarked in these experiments. "Vanilla" indicates the vanilla linear attention [28].

[*] *This work was done during an internship at Adobe Research.*

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Linearization paradigm.

(b) Comparison of decoder and encoder-decoder architecture.

Figure 2: (a) Comparison between training-from-scratch and linearization paradigms for ViTs with linear attention. Training-from-scratch linear attention paradigms focus on designing accurate attention approximation methods, which typically require large-scale pretraining to acquire strong prior knowledge. In contrast, ViT linearization leverages an off-the-shelf pretrained ViT, substantially reducing the need for extensive pretraining. (b) Comparison of decoder and encoder–decoder architectures. In decoder-based LLMs, the LLM serves as both the feature extractor and the target generator, where the error propagates temporally. In contrast, in vision models, ViTs function solely as feature extractors, while a separate task-specific head is responsible for target generation. The ViT error propagates spatially and hierarchically.

To address the computational and memory bottlenecks of ViTs, extensive efforts have been devoted to improving the efficiency of softmax-based self-attention, including attention matrix optimization [15], token reduction methods [42], distillation [45], sliding-window mechanisms [36], sequence modeling approaches [22], and linear attention variants [28]. Among these, linear attention methods are particularly attractive, as they reduce the quadratic complexity ($\mathcal{O}(N^2D)$), to linear complexity ($\mathcal{O}(ND^2)$), where N and D indicate the sequence length and the feature dimension, respectively. Existing linear attention approaches can be categorized into two types (Fig. 2a): training from scratch [57, 53] and linearization [60, 61, 38, 30, 21]. The former focuses on designing accurate softmax-approximation methods, and trains a linearized ViT entirely from scratch, typically requiring large-scale pretraining before fine-tuning on downstream tasks, especially for the VFMs designed as general-purpose feature extractors. Without such extensive pretraining, these approaches often suffer from severe performance degradation when directly adapted to downstream scenarios (see Tab. 1, 2, 3), limiting their practicality under realistic data and compute constraints. The latter, in contrast, inherits prior knowledge from the softmax-based VFMs and therefore requires substantially fewer pretraining steps than training-from-scratch methods.

However, existing work on linearization stream such as LoLCATS [61] has primarily focused on large language models (LLMs), which are decoder-based transformers and differ fundamentally from encoder–decoder-based vision models (see Fig. 2b). In decoder-only LLMs, the model acts as both a feature extractor and a target generator, whereas in vision models, the ViT primarily serves as a feature extractor and a separate prediction head functions as the generator. Consequently, directly transferring linear attention adaptation paradigms from LLMs to ViTs leads to a substantial performance drop. We attribute this to divergent error propagation: while LLM errors accumulate temporally, ViT errors accumulate spatially and hierarchically. This distorts the global semantic manifold essential for dense prediction, making feature alignment non-negotiable to preserve the spatial consistency vision tasks require. More analyses are provided in Appendix A.2.2, A.2.3.

To address these challenges, we introduct **ViT-AdaLA** (*Adapting Vision Transformers with Linear Attention*). ViT-AdaLA consists of three stages designed to inherit knowledge from a pretrained softmax-based ViT and transfer it to downstream tasks: attention alignment, feature alignment, and supervised fine-tuning. To effectively adapt the prior knowledge from the VFMs, we first align the linear attention module with the original softmax attention in each transformer block. We find that *tuning the vanilla linear attention module yields a strong approximation to the original softmax attention*, outperforming other linear attention variants (Fig. 4 and 6). Although the linear attention modules are aligned independently in each block during Stage 1, the residual approximation error accumulates across layers. To mitigate this accumulated error, we introduce a feature alignment stage that finetunes the entire linearized model. Specifically, we replace the original softmax attention with the aligned linear attention in Stage 1, and finetune the full linearized ViT to align its final-layer

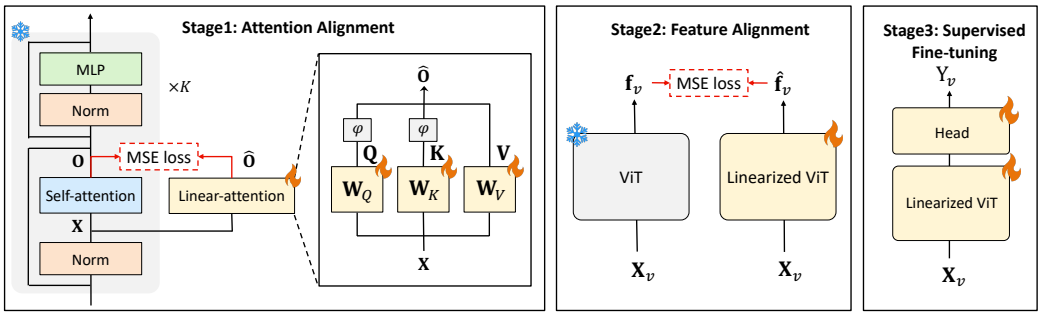


Figure 3: ViT-AdaLA consists of three stages: attention alignment, feature alignment, and supervised fine-tuning. First, softmax attention is approximated by tuning only the linear attention modules. Second, to mitigate residual approximation errors that accumulate across layers, the feature alignment stage finetunes the entire linearized model by aligning its final-layer representations with those of the original softmax-based teacher. Finally, supervised fine-tuning is performed to transfer the adapted prior knowledge to downstream tasks.

features with those of the frozen softmax-based teacher model. Interestingly, we observe that *the attention alignment in Stage 1 can accelerate convergence during this feature alignment process*. Finally, we perform supervised fine-tuning to transfer the adapted prior knowledge to downstream tasks. Our contributions are three-fold:

- We introduce a new paradigm for ViTs with linear attention that shifts the focus from designing more accurate attention approximations to adapting prior knowledge from pretrained ViTs. Our paradigm enables linearized ViTs to inherit the power of existing VFMs, eliminating the need for expensive training from scratch.
- We introduce *ViT-AdaLA*, which adapts VFMs via attention alignment, feature alignment, and supervised fine-tuning. This progressive alignment allows linear attention models to inherit the strong priors of softmax-based ViTs. Furthermore, our framework is architecture-agnostic and compatible with other linear attention methods.
- We perform extensive experiments on classification and segmentation tasks across multiple VFMs, and compare against a wide range of state-of-the-art linear attention baselines. Experimental results validate the effectiveness, efficiency, and scalability in resolution of ViT-AdaLA across different VFMs and downstream tasks.

2 Related Work

Efficient Attention. The Transformer architecture [47] has been widely adopted in both natural language processing and vision tasks due to its scalability. However, the quadratic complexity of standard attention limits long-context understanding, leading to numerous approaches to reduce *memory* and *computation* overhead.

FlashAttention [15, 13, 43] improves memory efficiency by employing tile-based computation instead of explicitly materializing the full attention matrix. To further reduce the number of visual tokens and improve computational efficiency, some methods either select informative tokens [42] or merge redundant ones [4, 58, 31]. Others propose to distill knowledge from a large ViT to a smaller one [54, 45] or a more efficient model [2, 51]. Swin Transformer [36, 37] introduces a shifted-window mechanism to restrict dense attention computation within local regions. More recently, Mamba-based architectures [22, 35, 64, 48] have drawn significant attention due to their linear complexity, achieved through selective state-space modeling. Notably, Mamba can be seen as a variant of linear attention with specialized linear attention and modified block design [25].

Linear Attention. Existing linearized Transformers can be broadly categorized into two streams: training-from-scratch-based and linearization-based approaches. Training-from-scratch-based approach targets at designing accurate attention approximation methods and trains from scratch to obtain prior knowledge. One stream designs alternative activation functions after queries and keys for better approximation [24, 28, 23, 44, 40, 3, 29, 1, 3]. Another family of methods employs low-rank

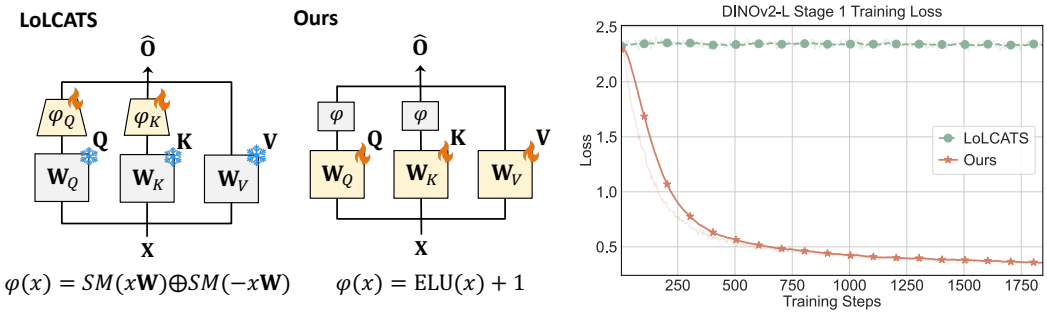


Figure 4: Linear attention architecture and Stage 1 training loss comparison with LoLCATS (SM: softmax; $\oplus$: concatenation). LoLCATS approximates the attention output based on Hedgehog [60] by tuning only two additional mapping modules applied to the queries and keys individually. In contrast, we tune all query, key, and value weights to approximate the attention output, which is both more efficient and more effective than the original LoLCATS approach.

decomposition to approximate, treating the softmax operation over queries and keys as a whole and decomposing it to derive more effective feature maps [53, 26, 52, 57, 55]. Recent work [19] observes that rank augmentation is beneficial for improving the performance. Another stream tries to combine convolution kernel with linear attention, preserving local and global information [63, 5]. However, these methods typically require large-scale pretraining before fine-tuning on downstream tasks, which is computationally and resource intensive.

In contrast, linearization-based approaches aim to adapt existing softmax-based Transformers to linearized one. Hedgehog [60] approximates the attention matrix using the Hedgehog linear-attention module. LoLCATS [61] introduces attention transfer to approximate attention outputs and employs low-rank linearization based on LoRA [27] for decoder-based LLMs. Building upon LoLCATS, Lizard [46], a hybrid attention paradigm, combines global attention via GLA [56] with local attention. Nevertheless, these methods cannot be directly applied to vision tasks due to architectural differences, as illustrated in Fig. 2b. To address this challenge, we propose **ViT-AdaLA**, a novel method to extend the linearization paradigm to ViTs.

3 Method

3.1 Preliminary

First, we briefly review the fundamentals of softmax and linear attention.

Softmax attention. Softmax attention is the fundamental module of the original Transformer, responsible for computing the pairwise attention among all input tokens. Let $\mathbf{X} \in \mathbb{R}^{N\times D}$ denote a sequence of N tokens, each with dimension D. The output $\mathbf{O} \in \mathbb{R}^{N\times D}$ is given by

$$\mathbf{O}_i = \sum_{j=1}^{N} \frac{\exp(\mathbf{Q}_i \mathbf{K}_j^T)}{\sum_{k=1}^{N} \exp(\mathbf{Q}_i \mathbf{K}_k^T)} \mathbf{V}_j, \tag{1}$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ denote the query, key, and value representations of the input tokens, obtained by multiplying $\mathbf{X}$ with the corresponding projection matrices $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$, respectively. Here, $\exp(\cdot)$ denotes the exponential function, and we omit the common scaling factor $1/\sqrt{D}$ for simplicity. The computational complexity of softmax attention is $\mathcal{O}(N^2 D)$.

Linear Attention. The kernel trick, expressed as $\exp(\mathbf{Q}\mathbf{K}^T) = \phi(\mathbf{Q})\phi(\mathbf{K}^T)$, is employed to decompose the multiplication of $\mathbf{Q}$ and $\mathbf{K}$ and to reorder the computation:

$$\mathbf{O}_i = \sum_{j=1}^{N} \frac{\phi(\mathbf{Q}_i)\phi(\mathbf{K}_j^T)}{\sum_{k=1}^{N} \phi(\mathbf{Q}_i)\phi(\mathbf{K}_k^T)} \mathbf{V}_j = \frac{\phi(\mathbf{Q}_i)\sum_{j=1}^{N} \phi(\mathbf{K}_j^T)\mathbf{V}_j}{\phi(\mathbf{Q}_i)\sum_{k=1}^{N} \phi(\mathbf{K}_k^T)}, \tag{2}$$

where $\phi(\cdot) = \mathrm{ELU}(\cdot) + 1$, $\mathrm{ELU}(\cdot)$ indicates the exponential linear unit [11]. We also compare with other types in the Appendix A.2.4. By reordering the multiplication computation of $\mathbf{Q}$ and $\mathbf{K}$, linear attention can achieve a computational complexity of $\mathcal{O}(ND^2)$.

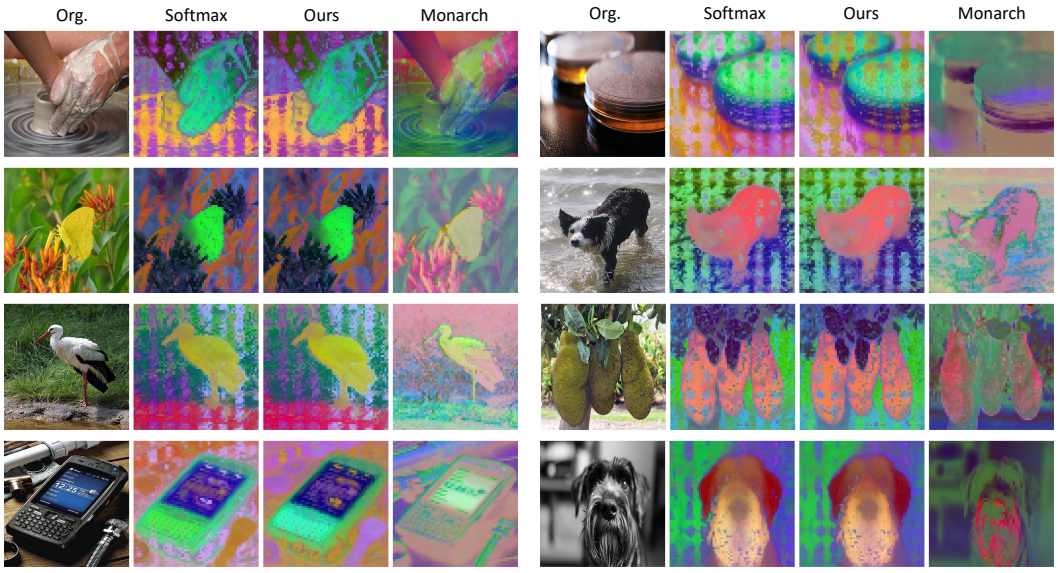


Figure 5: Visualization of PCA-projected features from the final layer of DINOv2-L. We compare the softmax-based features with those produced by ViT-AdaLA and Monarch attention [57] by projecting them to three channels using PCA. The results indicate that ViT-AdaLA better preserves the prior feature knowledge of the VFM. We provide more visualization results in the Appendix B.

3.2 ViT-AdaLA

ViT-AdaLA consists of three stages (see Fig. 3): attention alignment, feature alignment, and supervised fine-tuning.

Stage 1: Attention Alignment. To preserve the original attention quality while approximating softmax attention, we introduce an additional linear attention module and align it with the corresponding softmax attention module. Rather than adapting the linear attention module from scratch, we adapt it from an existing softmax attention module (based on Eq. 2) by simply modifying the computation order of queries, keys, and values using kernel trick. All components of the original model are frozen, except for the added linear attention module, where we only update the three projection matrices $\mathbf{W}_Q, \mathbf{W}_K$, and $\mathbf{W}_V$.

Formally, let the input to the i-th block after the first layer normalization be denoted as $\mathbf{O}_i \in \mathbb{R}^{N\times D}$. The output of the original self-attention module is $\mathbf{O}_i = f_\theta^{SA}(\mathbf{X}_i) \in \mathbb{R}^{N\times D}$, where $f_\theta^{SA}(\cdot)$ denotes the softmax-based self-attention. The attention alignment loss $\mathcal{L}_{att}$ is then defined as:

$$\mathcal{L}_{att} = \frac{1}{N \cdot D} \sum_{i=1}^{K} \sum_{n=1}^{N} \sum_{d=1}^{D} \left(\mathbf{O}_i^{nd} - \hat{\mathbf{O}}_i^{nd} \right)^2, \tag{3}$$

where $\hat{\mathbf{O}}_i = f_\theta^{LA}(\mathbf{X}_i)$ denotes the output of the linear attention module, n and d index the token and feature dimension, respectively, and K is the number of layers. The alignment loss is defined using the mean-square-error (MSE), which measures the discrepancy between the feature maps produced by the self-attention and linear-attention modules in each block. Importantly, the original features remain unchanged. We only adjust the linear-attention module to better approximate the behavior of the softmax self-attention.

Unlike the attention transfer strategy in LoLCATS [61], which tunes only two additional mapping modules applied to the queries and keys (*i.e.*, Hedgehog linear attention), we adopt a vanilla linear attention formulation that relies on a simple activation function and directly tunes the query, key, and value projection matrices. We posit that the vanilla linear attention is highly malleable. Unlike sophisticated approximations whose rigid structural priors can "fight" the teacher during distillation, the vanilla attention's unconstrained nature avoids optimization bottlenecks, allowing it to flexibly learn necessary approximation patterns. As shown in Fig. 4, this design offers two key advantages

Table 1: Top-1 fine-tuning accuracy comparison on ImageNet-1K under different vision foundation models with multiple linear attention baselines. We reproduce all baselines to ensure a fair comparison. The classification head is a single linear layer for all the methods.

	Backbone	Res.	Params (M)	FLOPS (G)	Peak Mem. (GB)	Throughput (imgs/s)	Top-1 Acc. (%)
Softmax [39]	DINOv2-L	512^2	304.20	310.60	1.3181	36.52	86.8
Hedgehog [60]	DINOv2-L	512^2	305.77	265.41$_{\downarrow 14.5\%}$	1.2221$_{\downarrow 7.3\%}$	37.44$_{\uparrow 2.5\%}$	58.8
LoLCATS [61]	DINOv2-L	512^2	305.77	265.41$_{\downarrow 14.5\%}$	1.2221$_{\downarrow 7.3\%}$	37.44$_{\uparrow 2.5\%}$	61.6
Linformer [50]	DINOv2-L	512^2	305.77	262.19$_{\downarrow 15.6\%}$	1.2221$_{\downarrow 7.3\%}$	45.41$_{\uparrow 25.3\%}$	68.9
Performer [10]	DINOv2-L	512^2	304.20	265.41$_{\downarrow 14.6\%}$	1.2164$_{\downarrow 7.7\%}$	36.48$_{\uparrow 0.1\%}$	75.2
Cosformer [40]	DINOv2-L	512^2	304.20	265.41$_{\downarrow 14.6\%}$	1.2226$_{\downarrow 7.2\%}$	39.55$_{\uparrow 8.2\%}$	75.1
Nyströmformer [53]	DINOv2-L	512^2	304.20	265.20$_{\downarrow 14.6\%}$	1.2163$_{\downarrow 7.7\%}$	18.29$_{\downarrow 50.8\%}$	82.4
Monarch [57]	DINOv2-L	512^2	304.20	269.44$_{\downarrow 13.3\%}$	1.2304$_{\downarrow 6.7\%}$	18.31$_{\downarrow 48.9\%}$	82.7
ViT-AdaLA (Stage 2)	DINOv2-L	512^2	304.20	262.19$_{\downarrow 15.6\%}$	1.2163$_{\downarrow 7.7\%}$	41.56$_{\uparrow 16.1\%}$	84.5
ViT-AdaLA (Ours)	DINOv2-L	512^2	304.20	262.19$_{\downarrow 15.6\%}$	1.2163$_{\downarrow 7.7\%}$	41.56$_{\uparrow 16.1\%}$	86.0
Softmax [41]	CLIP-L	512^2	304.15	310.60	1.3179	36.82	86.4
ViT-AdaLA (Stage 2)	CLIP-L	512^2	304.15	262.19$_{\downarrow 15.6\%}$	1.2161$_{\downarrow 7.4\%}$	42.14$_{\uparrow 15.7\%}$	83.4
ViT-AdaLA (Ours)	CLIP-L	512^2	304.15	262.19$_{\downarrow 15.6\%}$	1.2161$_{\downarrow 7.4\%}$	42.14$_{\uparrow 15.7\%}$	85.5
Softmax [59]	SigLIP-L	512^2	316.74	312.46	1.3636	40.86	86.9
ViT-AdaLA (Stage 2)	SigLIP-L	512^2	316.74	264.04$_{\downarrow 15.5\%}$	1.2620$_{\downarrow 7.4\%}$	46.91$_{\uparrow 13.9\%}$	84.9
ViT-AdaLA (Ours)	SigLIP-L	512^2	316.74	264.04$_{\downarrow 15.5\%}$	1.2620$_{\downarrow 7.4\%}$	46.91$_{\uparrow 13.9\%}$	86.4

compared to Hedgehog-based methods: (i) higher computational efficiency, and (ii) improved approximation quality.

Stage 2: Feature Alignment. Although we align the linear attention module with the softmax-based self-attention in each transformer block, the original features remain untuned, and replacing self-attention with linear attention introduces residual approximation errors to accumulate across blocks (see Fig. 14 in Appendix B). To ensure that the final output features of the linearized ViT remain consistent with the original model, we directly align the final features of the two models. Benefiting from the attention alignment in Stage 1, the linearized ViT converges faster and more effectively transfers prior knowledge from VFMs (see Sec. 4.3.1).

Specifically, we replace all softmax-based self-attention modules with the linear-attention modules obtained in Stage 1, resulting in a linearized ViT. This linearized ViT is then aligned with the frozen original ViT. Given the same input image $\mathbf{X}_v \in \mathbb{R}^{N\times D}$ for both models, we define the feature alignment loss $\mathcal{L}_{fa}$ as follows:

$$\mathcal{L}_{fa} = \frac{\lambda}{N \cdot D} \sum_{n=1}^{N} \sum_{d=1}^{D} (\mathbf{f}_v^{nd} - \hat{\mathbf{f}}_v^{nd})^2, \tag{4}$$

where $\mathbf{f}_v = \mathcal{F}_{\theta_0}(\mathbf{X}_v)$ and $\hat{\mathbf{f}}_v = \mathcal{F}_{\theta}(\mathbf{X}_v)$ denote the final representations produced by the original ViT and the linearized ViT, respectively. λ controls the scale of the output loss and is set to different values for different VFMs. We utilize MSE loss to align two final features. During this stage, $\mathcal{F}_{\theta_0}$ is kept frozen, while only $\mathcal{F}_{\theta}$ is updated.

Stage 3: Supervised Fine-tuning. After feature alignment, we transfer the linearized ViT, enriched with prior knowledge from VFMs, to downstream tasks by fine-tuning it on task-specific datasets. In this stage, a task-specific head is appended to the linearized ViT and both the backbone and the task head are updated during this process.

4 Experiment

We pretrain linearized VFMs using our ViT-AdaLA pipeline through Stages 1 and 2. Specifically, we train four linearized VFMs within the PyTorch Lightning framework using 8 $\times$ H100 GPUs. Stage 1 training is conducted on COCO [34] for 4 epochs with batch size 32 per GPU, while Stage 2 training is performed on ImageNet-22K [16] for 10 to 30 epochs with batch size 16 per GPU. We employ the AdamW optimizer with fixed learning rate $1e^{-2}$ and initial learning rate $1e^{-4}$ and a linearly decaying learning rate schedule for Stage 2. We multiply the learning rate with the ratio of 0.1 to the backbone when training. All models are trained using 512×512 input images, with random cropping and color jitter applied for data augmentation. More details can refer to Appendix A.1.1.

Table 2: mIoU fine-tuning comparison on ADE20K under different vision foundation models with multiple linear attention baselines. We reproduce all baselines to ensure a fair comparison. The segmentation head is Mask2former for all the methods.

	Backbone	Res.	Params (M)	FLOPS (G)	Peak Mem. (GB)	Throughput (imgs/s)	mIoU (%)
Softmax [39]	DINOv2-L	512^2	304.20	310.60	1.3181	36.52	56.73
Hedgehog [60]	DINOv2-L	512^2	305.77	265.41$_{\downarrow 14.5\%}$	1.2221$_{\downarrow 7.3\%}$	37.44$_{\uparrow 2.5\%}$	22.52
LoLCATS [61]	DINOv2-L	512^2	305.77	265.41$_{\downarrow 14.5\%}$	1.2221$_{\downarrow 7.3\%}$	37.44$_{\uparrow 2.5\%}$	17.42
Linformer [50]	DINOv2-L	512^2	305.77	262.19$_{\downarrow 15.6\%}$	1.2221$_{\downarrow 7.3\%}$	45.41$_{\uparrow 25.3\%}$	18.82
Performer [10]	DINOv2-L	512^2	304.20	265.41$_{\downarrow 14.6\%}$	1.2164$_{\downarrow 7.7\%}$	36.48$_{\uparrow 0.1\%}$	41.16
Cosformer [40]	DINOv2-L	512^2	304.20	265.41$_{\downarrow 14.6\%}$	1.2226$_{\downarrow 7.2\%}$	39.55$_{\uparrow 8.2\%}$	27.17
Nyströmformer [53]	DINOv2-L	512^2	304.20	265.20$_{\downarrow 14.6\%}$	1.2163$_{\downarrow 7.7\%}$	18.29$_{\downarrow 50.8\%}$	41.16
Monarch [57]	DINOv2-L	512^2	304.20	269.44$_{\downarrow 13.3\%}$	1.2304$_{\downarrow 6.7\%}$	18.31$_{\downarrow 48.9\%}$	44.95
ViT-AdaLA (Stage 2)	DINOv2-L	512^2	304.20	262.19$_{\downarrow 15.6\%}$	1.2163$_{\downarrow 7.7\%}$	41.56$_{\uparrow 16.1\%}$	52.46
ViT-AdaLA (Ours)	DINOv2-L	512^2	304.20	262.19$_{\downarrow 15.6\%}$	1.2163$_{\downarrow 7.7\%}$	41.56$_{\uparrow 16.1\%}$	55.55
Softmax [59]	SigLIP-L	512^2	316.74	312.46	1.3636	41.05	54.4
ViT-AdaLA (Stage 2)	SigLIP-L	512^2	316.74	264.04$_{\downarrow 15.5\%}$	1.2620$_{\downarrow 7.4\%}$	47.05$_{\uparrow 14.6\%}$	51.39
ViT-AdaLA (Ours)	SigLIP-L	512^2	316.74	264.04$_{\downarrow 15.5\%}$	1.2620$_{\downarrow 7.4\%}$	47.05$_{\uparrow 14.6\%}$	53.16
Softmax	IN1K ViT-L	512^2	304.15	310.60	1.3179	36.47	50.83
ViT-AdaLA (Stage 2)	IN1K ViT-L	512^2	304.15	262.19$_{\downarrow 15.6\%}$	1.2161$_{\downarrow 7.7\%}$	42.56$_{\uparrow 16.7\%}$	46.92
ViT-AdaLA (Ours)	IN1K ViT-L	512^2	304.15	262.19$_{\downarrow 15.6\%}$	1.2161$_{\downarrow 7.7\%}$	42.56$_{\uparrow 16.7\%}$	49.75

After pretraining, we benchmark performance on classification and semantic segmentation against existing linear attention baselines. Additionally, we perform ablation studies to analyze the impact of each training stage.

4.1 Comparison on Classification

Experimental setup. We conduct experiments on the ImageNet-1K [16] dataset. We report top-1 accuracy, parameter, throughput, and GFLOPs in Tab. 1. Throughput and peak memory (batch size 1) are measured on a single H100 GPU. This measurement setup extends to Tables 2 and 3. The baselines are constructed by replacing softmax attention modules with their linear counterparts. Full training details are provided in the Appendix A.1.1.

Result analysis. Tab. 1 shows that ViT-AdaLA achieves the highest top-1 accuracy among VFMs, *maintaining accuracy within 1% of the original softmax backbone while preserving efficiency*. We can also observe that for decoder-based linearization methods like Hedgehog [60] and LoLCATS [61], the final performance drops significantly since the linearized ViT has not been fully aligned with the VFM backbone, and *aligning attention alone is insufficient to transfer adequate prior knowledge*. This demonstrates that:

> *For vision models, it is essential to align the final feature representations with the original VFM backbone teacher to ensure effective knowledge transfer.*

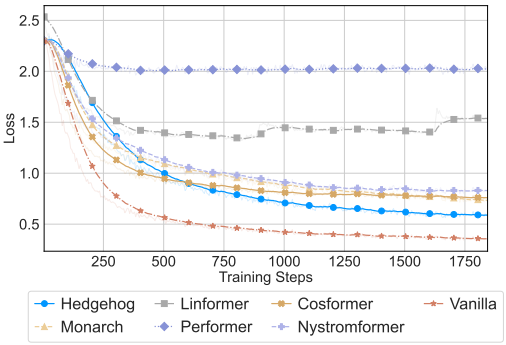


Figure 6: Training loss comparison of different linear attention variants in Stage 1 on DINOv2-L. To ensure a fair comparison, the query, key, and value weights are tuned in every layer for each baseline. Vanilla linear attention exhibits superior approximation performance compared to the other variants.

For training-from-scratch methods, low-rank approximation yields better than activation-based techniques. However, they incur greater memory and computation due to the mathematical complexity required for high-quality approximation. Nevertheless, these methods still fail to match the performance of ViT-AdaLA or even its Stage 2 baseline. This indicates: *linearization is superior to training-from-scratch methods for extracting prior knowledge from VFMs.*

Table 3: mIoU fine-tuning comparison on Cityscapes under different input resolutions (512 vs. 1024) based on DINOv2-L. We reproduce all baselines to ensure a fair comparison. The segmentation head is Mask2former for all the methods.

	Backbone	Res.	Params (M)	FLOPS (G)	Peak Mem. (GB)	Throughput (imgs/s)	mIoU (%)
Softmax [39]	DINOv2-L	512^2	304.20	310.60	1.3181	36.52	74.86
Hedgehog [60]	DINOv2-L	512^2	305.77	265.41$_{\downarrow 14.5\%}$	1.2221$_{\downarrow 7.3\%}$	37.44$_{\uparrow 2.5\%}$	43.97
LoLCATS [61]	DINOv2-L	512^2	305.77	265.41$_{\downarrow 14.5\%}$	1.2221$_{\downarrow 7.3\%}$	37.44$_{\uparrow 2.5\%}$	34.71
Linformer [50]	DINOv2-L	512^2	305.77	262.19$_{\downarrow 15.6\%}$	1.2221$_{\downarrow 7.3\%}$	45.41$_{\uparrow 25.3\%}$	37.24
Performer [10]	DINOv2-L	512^2	304.20	265.41$_{\downarrow 14.6\%}$	1.2164$_{\downarrow 7.7\%}$	36.48$_{\uparrow 0.1\%}$	43.11
Cosformer [40]	DINOv2-L	512^2	304.20	265.41$_{\downarrow 14.6\%}$	1.2226$_{\downarrow 7.2\%}$	39.55$_{\uparrow 8.2\%}$	44.52
Nyströmformer [53]	DINOv2-L	512^2	304.20	265.20$_{\downarrow 14.6\%}$	1.2163$_{\downarrow 7.7\%}$	18.29$_{\downarrow 50.8\%}$	48.97
Monarch [57]	DINOv2-L	512^2	304.20	269.44$_{\downarrow 13.3\%}$	1.2304$_{\downarrow 6.7\%}$	18.31$_{\downarrow 48.9\%}$	50.84
ViT-AdaLA (Stage 2)	DINOv2-L	512^2	304.20	262.19$_{\downarrow 15.6\%}$	1.2163$_{\downarrow 7.7\%}$	41.56$_{\uparrow 16.1\%}$	71.32
ViT-AdaLA (Ours)	DINOv2-L	512^2	304.20	262.19$_{\downarrow 15.6\%}$	1.2163$_{\downarrow 7.7\%}$	41.56$_{\uparrow 16.1\%}$	72.40
Softmax [39]	DINOv2-L	1024^2	304.20	1241.0	3.2836	7.07	80.98
Hedgehog [60]	DINOv2-L	1024^2	305.77	442.24$_{\downarrow 64.4\%}$	1.3823$_{\downarrow 57.9\%}$	15.38$_{\uparrow 117.4\%}$	44.17
LoLCATS [61]	DINOv2-L	1024^2	305.77	442.24$_{\downarrow 64.4\%}$	1.3823$_{\downarrow 57.9\%}$	15.38$_{\uparrow 117.4\%}$	33.66
Linformer [50]	DINOv2-L	1024^2	305.77	429.35$_{\downarrow 65.4\%}$	1.3999$_{\downarrow 57.4\%}$	17.83$_{\uparrow 152.0\%}$	26.63
Performer [10]	DINOv2-L	1024^2	304.20	442.24$_{\downarrow 64.4\%}$	1.3765$_{\downarrow 58.1\%}$	15.23$_{\uparrow 115.3\%}$	42.58
Cosformer [40]	DINOv2-L	1024^2	304.20	442.24$_{\downarrow 64.4\%}$	1.3945$_{\downarrow 57.5\%}$	15.29$_{\uparrow 116.1\%}$	44.50
Nyströmformer [53]	DINOv2-L	1024^2	304.20	442.24$_{\downarrow 64.4\%}$	1.3764$_{\downarrow 58.1\%}$	14.61$_{\uparrow 106.5\%}$	47.56
Monarch [57]	DINOv2-L	1024^2	304.20	500.24$_{\downarrow 59.7\%}$	1.5469$_{\downarrow 52.9\%}$	9.74$_{\uparrow 37.6\%}$	55.18
ViT-AdaLA (Stage 2)	DINOv2-L	1024^2	304.20	429.35$_{\downarrow 65.4\%}$	1.3764$_{\downarrow 58.1\%}$	15.95$_{\uparrow 125.4\%}$	77.53
ViT-AdaLA (Ours)	DINOv2-L	1024^2	304.20	429.35$_{\downarrow 65.4\%}$	1.3764$_{\downarrow 58.1\%}$	15.95$_{\uparrow 125.4\%}$	78.73

4.2 Comparison on Semantic Segmentation

Experimental setup. We further conduct experiments on ADE20K [62] and Cityscapes [12] to provide a more fine-grained evaluation of ViT-AdaLA when transferring from VFMs. For both semantic segmentation datasets, we employ the Mask2Former head [9] across all baselines. We consider two experimental settings: evaluating different VFMs on ADE20K in Tab. 2, and assessing the impact of input resolutions on Cityscapes in Tab. 3.

Result analysis. Since segmentation requires more low-level and fine-grained features than classification, the ability of linearization to extract robust prior knowledge is essential for maintaining high performance in dense prediction tasks. As shown in Tab. 2, ViT-AdaLA demonstrates strong performance across various VFMs, rivaling even supervised baselines such as the IN1K-pretrained ViT [18] This highlights *the generalizability of ViT-AdaLA in effectively distilling prior knowledge from diverse VFMs and transferring it to different downstream tasks*.

We further explore the scaling ability of our ViT-AdaLA for higher resolution images as shown in Tab. 3. Our linear approach overcomes the efficiency bottleneck of softmax attention, delivering $>$50% memory savings and 2$\times$ faster inference.

Moreover, ViT-AdaLA generalizes across scales: although distilled at 512^2, its performance improves from 72.40% to 78.73% when scaled up to 1024^2, which shows the following property:

> *ViT-AdaLA can scale to higher-resolution images, even pretrained on lower-resolution ones.*

Such a property enables more efficient pretraining and broader applications of ViT-AdaLA. Ultimately, this flexibility resolves the tension between training costs and inference quality, establishing ViT-AdaLA as a versatile and practical paradigm for large-resolution dense prediction tasks.

4.3 Ablation Study

4.3.1 Effectiveness of Pretraining Stages

We ablate Stages 1 and 2 on DINOv2-L/ADE20K and SAM-H/SAM-HQ (Table 4). For SAM-H, training uses an internal foreground-segmentation dataset and evaluation uses SAM-HQ.

Stage 1: Attention Alignment. Stage 1 improves the Stage-2 result by 3.09 mIoU on ADE20K and 4.01 IoU on SAM-HQ. Figure 7 shows a faster decrease in Stage-2 feature-MSE with Stage 1 initialization.

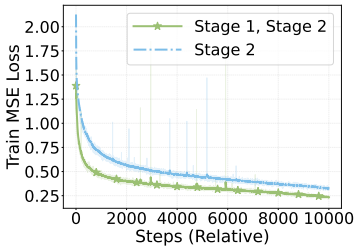


Figure 7: Stage-2 feature-MSE loss with/without Stage 1 (DINOv2-L).

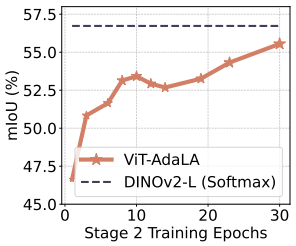


Figure 8: ADE20K performance across different Stage 2 training epochs for DINOv2-L.

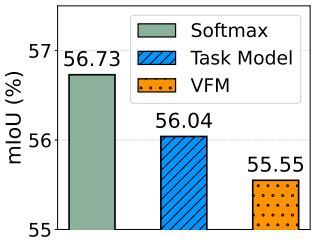
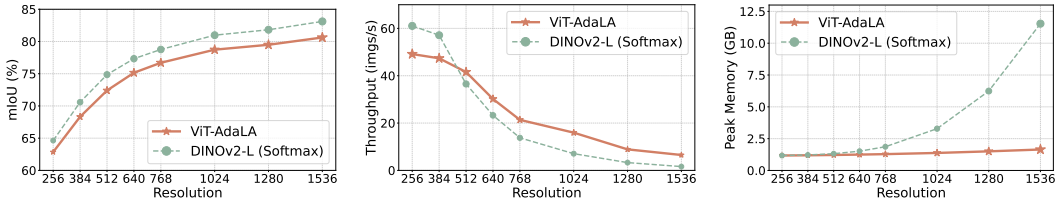


Figure 9: ADE20K performance of adapting task model or VFM for DINOv2-L.



(a) mIoU (b) Throughput (c) Peak memory, batch size one

Figure 10: Resolution scalability on Cityscapes: mIoU, throughput, and peak memory.

Table 4: Stage 1/2 ablation on ADE20K (DINOv2-L) and SAM-HQ (SAM-H).

Stage 1	Stage 2	ADE20K mIoU	SAM-HQ IoU
×	×	22.92	63.65
✓	×	19.37	72.00
×	✓	52.46	73.47
✓	✓	**55.55**	**77.48**
Softmax teacher		56.73	–

Stage 1's standalone effect is task-dependent: on ADE20K it obtains 19.37 mIoU, below the unaligned baseline (22.92), whereas on SAM-HQ it raises IoU from 63.65 to 72.00 (+8.35). These results support Stage 1 as a complementary initialization that can also improve foreground segmentation independently.

Figure 6 supports vanilla linear attention as our default through its lower Stage-1 alignment loss.

Stage 2: Feature Alignment. *Recovering pretrained compatibility.* With the attention architecture fixed, Stage 2 alone raises ADE20K mIoU from 22.92 to 52.46 (recovering 87.4% of the teacher gap) and SAM-HQ IoU from 63.65 to 73.47 (Table 4). Direct-replacement failure therefore need not imply inadequate linear-attention capacity; it can reflect disrupted compatibility with pretrained blocks.

Local fidelity is not sufficient. On ADE20K, Stage 1 alone reduces mIoU by 3.55, yet adds 3.09 with Stage 2: its transfer benefit is conditional on global adaptation. Stage 1 fits attention on teacher inputs; the student receives features altered by preceding replacements. Stage 2 aligns final features under the student's actual forward pass, letting the whole encoder co-adapt without layer-by-layer equality (Appendix A.2.3). It converts local initialization into useful end-to-end transfer.

Recovery is transferable. After Stages 1–2, frozen-backbone transfer retains 98.4% of teacher ImageNet-1K accuracy and 99.7% on Flowers-102 (Table 5); held-out Flowers-102 CKA rises from 0.196 to 0.993 (Table 20). Recovery is thus visible on unseen data without downstream backbone updates. With the same full pipeline, Nyströmformer and Monarch gain 13.44 and 9.95 mIoU (Table 18), supporting a shared representation-transfer objective across attention modules. Figure 8 shows Stage-2 progress.

VFM linearization should preserve the encoder's transferable representation, not merely approximate its individual attention operators.

Table 5: DINOv2-L frozen-backbone transfer after Stage 2. Matched task heads are trained; retained quality is the student-to-teacher ratio. ADE20K values are approximate.

Task	Head / frozen (M)	Softmax	ViT-AdaLA	Retained
ImageNet-1K linear probe	1.0M / 304M	82.66	81.30	98.4%
Flowers-102 linear probe	0.10M / 304M	99.71	99.37	99.7%
ADE20K Mask2Former	14.8M / 304M	∼50.0	∼49.5	∼99.0%

4.3.2 Resolution Scalability Analysis

Although adapted at 512^2, ViT-AdaLA transfers to Cityscapes resolutions from 256^2 to 1536^2 (Figure 10). It maintains strong mIoU while improving throughput and memory efficiency at high resolution. At 256^2 and batch size one, however, softmax is faster; the linearized model overtakes it by 512^2. Matched throughput measurements are given in Appendix A.2.1.

4.3.3 Directly Adapting the Task Model

Figure 9 compares adapting a task-trained teacher with adapting the general-purpose VFM. The task-trained teacher reaches 56.04 mIoU on ADE20K, compared with 55.55 for the VFM teacher, suggesting that task-specific representations can also be transferred. Both settings use full downstream fine-tuning after conversion; this is not a frozen-backbone evaluation.

4.3.4 Frozen-Backbone Transfer

We freeze all 304M DINOv2-L backbone weights after Stage 2 and train matched task heads (Table 5). ViT-AdaLA retains 98.4% of the softmax teacher's ImageNet-1K accuracy, 99.7% on held-out Flowers-102, and approximately 99.0% ADE20K mIoU. Thus, Stage 3 full-backbone fine-tuning is not required for these tasks; the transferred features remain useful with only head training.

4.3.5 Effect of QKV-Only Tuning

Table 6 compares QKV-only tuning with full-network tuning. The latter improves ImageNet-1K accuracy by 3.0 points on CLIP-L and 9.6 points on DINOv2-L, and ADE20K mIoU by 7.44 and 6.73 points, respectively. The gap indicates that QKV-only updates, while useful for local attention alignment in Stage 1, do not suffice for the complete adaptation; updating the wider network recovers substantially more task performance.

Table 6: Full-network versus QKV-only tuning on CLIP-L and DINOv2-L at 512^2 resolution.

Backbone	Tuning	ImageNet-1K Acc.	ADE20K mIoU
CLIP-L	Full network	85.5	42.85
CLIP-L	QKV only	82.5	35.41
DINOv2-L	Full network	86.0	55.55
DINOv2-L	QKV only	76.4	48.82

5 Conclusion

We introduce ViT-AdaLA, a framework that adapts Vision Foundation Model (VFM) priors into linearized ViTs without large-scale pretraining. Our three-stage alignment process effectively distills priors, accelerates convergence, and ensures robust resolution scalability. This paradigm offers a fresh perspective for linearization research, paving the way for high-resolution image applications. Future research can explore more efficient and effective linearization architectures.

Acknowledgments and Disclosure of Funding

This project is partially supported by the Adobe gift funding program and NSF grant #2429836 under the NSF Responsible Design, Development, and Deployment of Technologies (ReDDDoT) program. The ReDDDoT program is jointly sponsored by NSF and the Ford Foundation, the Patrick J. McGovern Foundation, Pivotal Ventures, the Eric and Wendy Schmidt Fund for Strategic Innovation, and Siegel Family Endowment.

References

[1] Sabbir Ahmed, Jingtao Li, Weiming Zhuang, Chen Chen, and Lingjuan Lyu. Mixa: A mixed attention approach with stable lightweight linear attention to enhance efficiency of vision transformers at the edge. In *ICCV*, pages 21187–21196, 2025.

[2] Aviv Bick, Tobias Katsch, Nimit Sharad Sohoni, Arjun D Desai, and Albert Gu. Llamba: Scaling distilled recurrent models for efficient language processing. In *First Workshop on Scalable Optimization for Efficient and Adaptive Foundation Models*, 2025.

[3] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, and Judy Hoffman. Hydra attention: Efficient attention with many heads. In *ECCV*, pages 35–49, 2022.

[4] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *ICLR*, 2023.

[5] Han Cai, Junyan Li, Muyan Hu, Chuang Gan, and Song Han. Efficientvit: Lightweight multi-scale attention for high-resolution dense prediction. In *ICCV*, pages 17302–17313, 2023.

[6] Ruoyu Chen, Hua Zhang, Siyuan Liang, Jingzhi Li, and Xiaochun Cao. Less is more: Fewer interpretable region via submodular subset selection. In *ICLR*, 2024.

[7] Ruoyu Chen, Xiaoqing Guo, Kangwei Liu, Siyuan Liang, Shiming Liu, Qunli Zhang, Hua Zhang, and Xiaochun Cao. Where mllms attend and what they rely on: Explaining autoregressive token generation. *arXiv preprint arXiv:2509.22496*, 2025.

[8] Ruoyu Chen, Siyuan Liang, Jingzhi Li, Shiming Liu, Maosen Li, Zhen Huang, Hua Zhang, and Xiaochun Cao. Interpreting object-level foundation models via visual precision search. In *CVPR*, pages 30042–30052, 2025.

[9] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *CVPR*, pages 1290–1299, 2022.

[10] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. In *ICLR*, 2021.

[11] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 4(5):11, 2015.

[12] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, pages 3213–3223, 2016.

[13] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

[14] Tri Dao, Beidi Chen, Nimit S Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. Monarch: Expressive structured matrices for efficient and accurate training. In *ICML*, pages 4690–4721, 2022.

[15] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*, volume 35, pages 16344–16359, 2022.

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *ACL*, pages 4171–4186, 2019.

[18] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICML*, 2020.

[19] Qihang Fan, Huaibo Huang, and Ran He. Breaking the low-rank dilemma of linear attention. In *CVPR*, pages 25271–25280, 2025.

[20] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *AISTAS*, volume 15, pages 315–323, 11–13 Apr 2011.

[21] Daniel Goldstein, Eric Alcaide, Janna Lu, and Eugene Cheah. Radlads: Rapid attention distillation to linear attention decoders at scale. *arXiv preprint arXiv:2505.03005*, 2025.

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

[23] Dongchen Han, Xuran Pan, Yizeng Han, Shiji Song, and Gao Huang. Flatten transformer: Vision transformer using focused linear attention. In *ICCV*, pages 5961–5971, 2023.

[24] Dongchen Han, Yifan Pu, Zhuofan Xia, Yizeng Han, Xuran Pan, Xiu Li, Jiwen Lu, Shiji Song, and Gao Huang. Bridging the divide: Reconsidering softmax and linear attention. In *NeurIPS*, volume 37, pages 79221–79245, 2024.

[25] Dongchen Han, Ziyi Wang, Zhuofan Xia, Yizeng Han, Yifan Pu, Chunjiang Ge, Jun Song, Shiji Song, Bo Zheng, and Gao Huang. Demystify mamba in vision: A linear attention perspective. In *NeurIPS*, volume 37, pages 127181–127203, 2024.

[26] Jiayi Han, Longbin Zeng, Liang Du, Xiaoqing Ye, Weiyang Ding, and Jianfeng Feng. Modify self-attention via skeleton decomposition for effective point cloud transformer. In *AAAI*, volume 36, pages 808–816, 2022.

[27] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1 (2):3, 2022.

[28] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*, pages 5156–5165, 2020.

[29] Soroush Abbasi Koohpayegani and Hamed Pirsiavash. Sima: Simple softmax-free attention for vision transformers. In *WACV*, pages 2607–2617, 2024.

[30] Disen Lan, Weigao Sun, Jiaxi Hu, Jusen Du, and Yu Cheng. Liger: Linearizing large language models to gated recurrent structures. In *ICML*, 2025.

[31] Yifan Li, Wentao Bao, Botao Ye, Zhen Tan, Tianlong Chen, Huan Liu, and Yu Kong. Window token concatenation for efficient visual large language models. In *CVPRW*, pages 3187–3197, 2025.

[32] Yifan Li, Zhixin Lai, Wentao Bao, Zhen Tan, Anh Dao, Kewei Sui, Jiayi Shen, Dong Liu, Huan Liu, and Yu Kong. Visual large language models for generalized and specialized applications. *arXiv preprint arXiv:2501.02765*, 2025.

[33] Yifan Li, Xin Li, Tianqin Li, Wenbin He, Yu Kong, and Liu Ren. Vit-split: Unleashing the power of vision foundation models via efficient splitting heads. In *ICCV*, pages 1979–1989, October 2025.

[34] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755, 2014.

[35] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, and Yunfan Liu. Vmamba: Visual state space model. In *NeurIPS*, volume 37, pages 103031–103063, 2024.

[36] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, pages 10012–10022, 2021.

[37] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *CVPR*, pages 12009–12019, 2022.

[38] Zeyu Liu, Souvik Kundu, Lianghao Jiang, Anni Li, Srikanth Ronanki, Sravan Bodapati, Gourav Datta, and Peter A Beerel. Lawcat: Efficient distillation from quadratic to linear attention with convolution across tokens for long context modeling. In *EMNLP Findings*, pages 20865–20881, 2025.

[39] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research Journal*, 2024.

[40] Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosformer: Rethinking softmax in attention. In *ICLR*, 2022.

[41] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763, 2021.

[42] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In *NeurIPS*, volume 34, pages 13937–13949, 2021.

[43] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. In *NeurIPS*, volume 37, pages 68658–68685, 2024.

[44] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *WACV*, pages 3531–3539, 2021.

[45] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, pages 10347–10357, 2021.

[46] Chien Van Nguyen, Ruiyi Zhang, Hanieh Deilamsalehy, Puneet Mathur, Viet Dac Lai, Haoliang Wang, Jayakumar Subramanian, Ryan A Rossi, Trung Bui, Nikos Vlassis, et al. Lizard: An efficient linearization framework for large language models. *arXiv preprint arXiv:2507.09025*, 2025.

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, volume 30, 2017.

[48] Feng Wang, Timing Yang, Yaodong Yu, Sucheng Ren, Guoyizhe Wei, Angtian Wang, Wei Shao, Yuyin Zhou, Alan Yuille, and Cihang Xie. Adventurer: Optimizing vision mamba architecture designs for efficiency. In *CVPR*, pages 30157–30166, 2025.

[49] Shusen Wang and Zhihua Zhang. Improving cur matrix decomposition and the nyström approximation via adaptive sampling. *JMLR*, 14(1):2729–2769, 2013.

[50] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[51] Guoyizhe Wei and Rama Chellappa. Vit-linearizer: Distilling quadratic knowledge into linear-time vision models. *arXiv preprint arXiv:2504.00037*, 2025.

[52] Chong Wu, Maolin Che, and Hong Yan. The cur decomposition of self-attention matrices in vision transformers. *Authorea Preprints*, 2024.

[53] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *AAAI*, volume 35, pages 14138–14148, 2021.

[54] Yunyang Xiong, Bala Varadarajan, Lemeng Wu, Xiaoyu Xiang, Fanyi Xiao, Chenchen Zhu, Xiaoliang Dai, Dilin Wang, Fei Sun, Forrest Iandola, et al. Efficientsam: Leveraged masked image pretraining for efficient segment anything. In *CVPR*, pages 16111–16121, 2024.

[55] Yixing Xu, Chao Li, Dong Li, Xiao Sheng, Fan Jiang, Lu Tian, and Emad Barsoum. Qt-vit: Improving linear attention in vit with quadratic taylor expansion. In *NeurIPS*, pages 83048–83067, 2024.

[56] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *ICML*, 2024.

[57] Can Yaras, Alec S Xu, Pierre Abillama, Changwoo Lee, and Laura Balzano. Monarchattention: Zero-shot conversion to fast, hardware-aware structured attention. *arXiv preprint arXiv:2505.18698*, 2025.

[58] Wang Zeng, Sheng Jin, Wentao Liu, Chen Qian, Ping Luo, Wanli Ouyang, and Xiaogang Wang. Not all tokens are equal: Human-centric visual analysis via token clustering transformer. In *CVPR*, pages 11101–11111, 2022.

[59] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *ICCV*, pages 11975–11986, 2023.

[60] Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Re. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. In *ICLR*, 2024.

[61] Michael Zhang, Simran Arora, Rahul Chalamala, Benjamin Frederick Spector, Alan Wu, Krithik Ramesh, Aaryan Singhal, and Christopher Re. Lolcats: On low-rank linearizing of large language models. In *ICLR*, 2025.

[62] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, pages 633–641, 2017.

[63] Yuan Zhou, Qingshan Xu, Jiequan Cui, Junbao Zhou, Jing Zhang, Richang Hong, and Hanwang Zhang. Care transformer: Mobile-friendly linear visual transformer via decoupled dual interaction. In *CVPR*, pages 20135–20145, 2025.

[64] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: efficient visual representation learning with bidirectional state space model. In *ICML*, pages 62429–62442, 2024.

Appendix Contents

A Experiments **16**

A.1 Experimental Setup and Baselines 16

A.2 Efficiency and Alignment Diagnostics 17

A.3 Training Cost and Model Ablations 20

A.4 Task Results and Batch-Size Scaling 21

A.5 Protocols and Transfer Validation 24

A.6 Alignment Objectives and Teacher Ablations 26

B More Visualization Results **26**

C Limitations **29**

D Future Directions **29**

E Impact Statement **29**

A Experiments

A.1 Experimental Setup and Baselines

We summarize the training configuration and the reproduced baseline implementations.

A.1.1 Configurations of ViT-AdaLA under Different VFMs

We train Stage 1 on COCO for four epochs (batch size 32 per GPU, fixed learning rate 10^{-2}) and Stage 2 on ImageNet-22K for 10–30 epochs depending on the backbone (batch size 16 per GPU, initial learning rate 10^{-3} with polynomial decay and a 0.1 backbone learning-rate multiplier). Both stages use AdamW, eight H100 GPUs, and 512×512 inputs with random cropping and color jitter. Table 7 gives the per-backbone settings.

Table 7: Hyperparameters of different training stages and VFMs.

	Stage 1	**Stage 2**				**Stage 3**	
Epochs	4	DINOv2	SigLIP	CLIP	IN-1K	Cls	seg
		30	10	15	20	10	20
Batch size	32	16				8	
Learning rate	1e-2	1e-3				1e-4	
Weight decay	0.05						
Learning schedule	fixed	Polynomial (decay factor of 0.9)					
Optimizer	AdamW						
Datasets	COCO	ImageNet22K				ImageNet1K	ADE20K/Cityscapes
Heads	N/A	N/A				Linear	Mask2former
λ	N/A	5	1	4000	1	N/A	

A.1.2 Details of Compared Baselines

In our experiments, we compare with multiple linear attention baselines including Hedgehog [60], LoLCATS [61], Linformer [50], Performer [10], Cosformer [40], Nystromformer [53] and Monarch Attention [57]. We reproduce all of these methods according to their perspective publicly available codebase. Below are the details about these baselines:

Hedgehog: Hedgehog [60] proposes two trainable linear attention feature maps $\phi(\cdot)$ after queries and keys. It then learns these two feature maps to mimic standard attention. By employing a distillation loss to train these maps directly against the original attention scores, Hedgehog bridges the performance gap between models. This approach allows the model to maintain linear complexity while recovering most power of the original model.

LoLCATS: LoLCATS [61] adopts a two-stage methodology to linearize LLMs, converting their quadratic complexity attention into a faster, subquadratic form without the massive cost of retraining from scratch. In the first stage, attention transfer, the model utilizes learnable feature maps (inspired by the Hedgehog architecture) trained via the MSE loss to approximate the final attention outputs, rather than the attention matrix itself. The second stage, low-rank adjusting, employs LoRA to tune the Q, K, V, and O projections followed by task-specific supervised fine-tuning. This methodology scales exceptionally well, delivering high performance and efficiency across decoder-based LLMs.

Monarch Attention: Monarch Attention [57] introduces a zero-shot method to convert standard softmax attention into a hardware-efficient, subquadratic form using Monarch matrices. Different from previous methods that require extensive retraining, Monarch Attention allows the plug-and-play replacement of attention layers in pretrained Transformers, maintaining high accuracy. This is achieved by the mathematical property that any dense matrix can be approximated by Monarch matrices [14]. By decomposing the attention operation into these structured components, they achieve a complexity of $\mathcal{O}(n\sqrt{n})$. The proposed method proves effective across diverse architectures,

including ViTs, various language model configurations (encoder-based and encoder-decoder), and Diffusion Transformers.

Nyströmformer: Nyströmformer [53] proposes to approximate the self-attention using the Nyström method [49] by sampling a subset of columns and rows. The core of this method involves a small number of "landmark" points to reconstruct the full, inherently low-rank attention matrix. This method further improves accuracy with an iterative Moore-Penrose pseudoinverse approximation and residual connections that stabilize training. Nyströmformer allows the model to process sequences with thousands of tokens efficiently while maintaining performance comparable performance comparable to standard BERT [17] models. In our experiments, we set the number of "landmark" points to 128, which is the head dimension.

Linformer: Linformer [50] introduces a low-rank approximation of self-attention based on the empirical observation that attention matrices have low intrinsic rank. It achieves linear complexity by projecting the key and value sequences into a lower-dimensional space using learned projection matrices, reducing the size of the attention computation from $N \times N$ to $N \times k$ with $k \ll N$. These projections are shared across heads and layers, allowing efficient scaling to long sequences with minimal overhead. Experiments show that Linformer matches or closely approaches standard Transformer performance on NLP tasks while dramatically lowering memory and computation costs. In our experiments, we set $k = 32$.

Performer: Performer [10] rethinks self-attention by replacing the softmax attention kernel with a random feature approximation (FAVOR+), enabling exact linear time attention in expectation. By mapping queries and keys into a positive random feature space, the attention computation is reformulated as a sequence of associative matrix products, avoiding explicit $N \times N$ attention matrices. This design preserve the probabilistic interpretation of softmax attention while strong numerical stability and unbiased estimates. Experiments show that Performer scales effectively to very long sequences with competitive accuracy, though performance depends on the number of random features and can degrade when approximation variance is high.

Cosformer: Cosformer [40] revists the softmax operation in self-attention and proposes replacing it with a cosine-based reweighting mechanism that enables linear-time attention. The authors argue that softmax's effectiveness comes from two key properties: nonnegativity of the attention matrix and a non-linear reweighting that concentates attention distributions. Specifically, Cosformer utilizes the ReLU-based kernel to ensure non-negativity and cosine-based reweighting to introduce a "locality bias" that stabilizes training and focuses the model on more relevant local correlations. In experiments, Cosformer evaluates its effectiveness on encoder-based and decoder-based language models.

A.2 Efficiency and Alignment Diagnostics

We examine resolution scaling and the local-to-global alignment errors that motivate Stage 2.

A.2.1 Additional Resolution-Dependent Inference Measurements

Table 8 supplements the resolution curves in Figure 10 with end-to-end throughput on one H100. At 256^2 and batch size one, the linearized model is slower; it overtakes softmax by 512^2 and gains substantially at 1024^2. Wider models may require longer sequences, but a width-dependent crossover was not measured.

Table 8: End-to-end throughput (images/s) for DINOv2-L on one H100. The 256^2 batch-size results are also shown in Table 16.

Resolution	Batch size	Softmax	ViT-AdaLA
256^2	1	62.83	49.28
256^2	16	252.29	260.54
512^2	1	36.52	41.56
1024^2	1	7.07	15.95

Table 9: Reconstruction loss for DINOv2-L after Stage 1.

Metric	Mean Layer-wise Loss (0–23)	Sum of Layer-wise Losses	Final Output Loss (Accumulated)
Value	0.0148	0.3546	2.2839

A.2.2 Layer Sensitivity

To investigate layer-wise sensitivity, we measured the approximation loss for each layer of DINOv2-L (Stage 1) in Fig. 11.

The isolated per-layer losses range from 0.0027 to 0.0511, with a sum of 0.3546. Replacing all layers gives a final-output discrepancy of 2.2839. Because these quantities are measured at different outputs, the ratio is a diagnostic of compounded representation drift rather than a layer-independent amplification coefficient.

The diagnostic motivates Stage 2: matching each attention block locally does not by itself preserve the representation of the full pretrained vision encoder.

A.2.3 Accumulated Representation Drift in ViTs

Tab. 9 compares the L2 reconstruction loss of individual layers after Stage 1, *i.e.*, Layer-wise Distillation, against the final output loss when all layers are replaced with linear attention simultaneously.

Note: Individual layer losses range from 0.0027 to 0.0511.

- **The compounding effect.** The sum of isolated layer losses is 0.3546, while the final-output discrepancy after replacing all layers is 2.2839. These losses have different output locations and scales, so their ratio should not be interpreted as a formal error-propagation coefficient. The result nevertheless shows that local alignment alone does not control final representation drift.

Local mismatch versus propagated drift. Let f_i and g_i denote the complete teacher and student blocks, including attention, normalization, feed-forward, and residual operations; the last mapping also includes the final encoder normalization. For a fixed image, their input token features are $\mathbf{h}_i$ and $\hat{\mathbf{h}}_i$, respectively, flattened for the following analysis. Define input drift $\boldsymbol{\delta}_i = \hat{\mathbf{h}}_i - \mathbf{h}_i$ and the block mismatch evaluated on the teacher input, $\boldsymbol{\epsilon}_i = g_i(\mathbf{h}_i) - f_i(\mathbf{h}_i)$. An exact decomposition is

$$\boldsymbol{\delta}_{i+1} = \underbrace{g_i(\mathbf{h}_i + \boldsymbol{\delta}_i) - g_i(\mathbf{h}_i)}_{\text{response to shifted input}} + \underbrace{\boldsymbol{\epsilon}_i}_{\text{teacher-input block mismatch}}. \tag{5}$$

Stage 1 fits attention outputs on shared teacher inputs. It therefore reduces a source of block mismatch, but neither its attention-output loss nor an isolated block loss directly measures the first term in Eq. 5.

For sufficiently small drift, a first-order expansion with $\mathbf{J}_i = \partial g_i / \partial \mathbf{h}|_{\mathbf{h}_i}$ gives

$$\boldsymbol{\delta}_{i+1} \approx \mathbf{J}_i \boldsymbol{\delta}_i + \boldsymbol{\epsilon}_i, \qquad \boldsymbol{\delta}_K \approx \mathbf{P}_{K,0}\boldsymbol{\delta}_0 + \sum_{i=0}^{K-1} \mathbf{P}_{K,i+1}\boldsymbol{\epsilon}_i, \tag{6}$$

where $\mathbf{P}_{K,j} = \mathbf{J}_{K-1} \cdots \mathbf{J}_j$ for $j < K$ and $\mathbf{P}_{K,K} = \mathbf{I}$. The initial drift $\boldsymbol{\delta}_0$ is zero when the input embeddings are shared, but can change when Stage 2 updates the student's embedding parameters. Each local mismatch is transformed by the remaining blocks before contributing to final drift. Consequently, minimizing individual local errors is not equivalent to minimizing their combined effect at the encoder output: downstream sensitivities and interactions between error directions also matter.

Role of Stage 2. Stage 2 directly supervises the final representation under the student's actual intermediate inputs. Updating the whole encoder allows both block mismatches and their propagation to change jointly, rather than restricting adaptation to QKV weights or requiring layer-by-layer equality. Its objective preserves the teacher's external feature mapping while leaving the student's internal computation free to adapt. The recovery from 22.92 to 52.46 mIoU with Stage 2 alone (Table 4) is consistent with this distinction; it does not establish which Jacobians changed or quantify

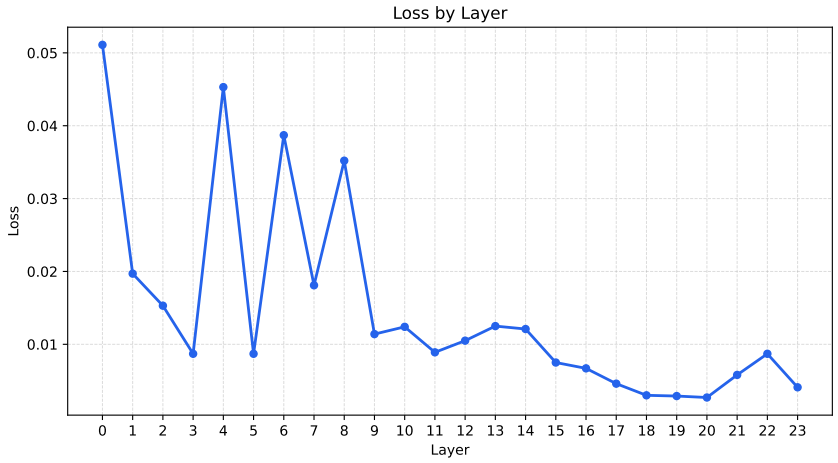


Figure 11: Layer-wise loss of Stage1 for DINOv2-L.

amplification. Eq. 6 is a local explanatory approximation, not an empirical measurement, and can be inaccurate for large drift.

The same shifted-input phenomenon occurs in deep Transformers generally. In a bidirectional ViT it can disturb relationships among spatial patches and the final feature geometry; it is not exclusive to vision models.

The Stage-2 output is also evaluated directly by frozen-head transfer (Table 5) and held-out representation metrics (Table 20). We do not infer from this vision-only experiment that LLMs lack accumulated residual errors.

A.2.4 Activation Choices for Linear Attention

Fig. 12 shows the influence of activation functions to the Stage 1 loss. Kernel-based linear attention approximation applys a non-negative feature map $\phi(\cdot)$ to the query and key matrices. We compare four different variants, *i.e.*, softmax, softplus, relu and ELU+1.

While linear attention aims to approximate this, the standard self-attention uses the softmax activation to normalize scores:

$$\mathrm{Softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}.$$

In kernelized linear attention, an exponential feature map $\phi(x) = \exp(x)$ is sometimes used to mimic this behavior, though it can be numerically unstable without proper scaling.

ReLU is a popular choice for linear attention due to its computational simplicity and ability to induce sparsity:

$$\phi(x) = \mathrm{ReLU}(x) = \max(0, x).$$

By mapping negative values to zero, it ensures the non-negativity required for the associative property of matrix multiplication in linear attention.

Softplus [20] is a smooth, differentiable approximation of ReLU:

$$\phi(x) = \mathrm{Softplus}(x) = ln(1 + e^x).$$

This activation is strictly positive and provides continuous gradients throughout the entire domain, which can lead to more stable training trajectories compared to ReLU. Commonly used in the original linear attention [28], the $\mathrm{ELU}(x) + 1$ feature map ensures that the output is always positive:

$$\phi(x) = \mathrm{ELU}(x) + 1 = \begin{cases} x + 1 & \text{if } x > 0 \\ e^x & \text{if } x \leq 0 \end{cases}.$$

This function combines the linear behavior for positive inputs with a smooth exponential decay toward zero for negative inputs.

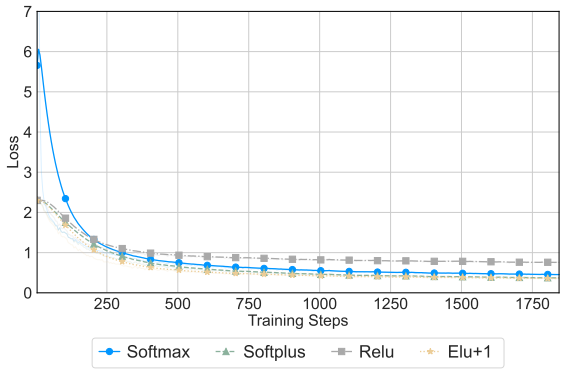


Figure 12: Stage 1 loss comparison across linear-attention activation functions. We compare four activations: softmax, softplus, ReLU, and ELU+1.

As illustrated in Fig. 12, $\mathrm{ELU} + 1$ feature map consistently achieves a lower training loss compared to other activation functions. Given its superior convergence characteristics and numerical stability, we select $\mathrm{ELU} + 1$ as the default activation for our final architecture.

A.3 Training Cost and Model Ablations

We report adaptation time and compare model sizes and scratch training.

A.3.1 Training Time Comparison

Table 10 reports measured per-epoch wall times on eight H100 GPUs. Stage 1 is brief, while Stage 2 dominates the one-time adaptation cost; aggregate GPU-hours are given in Table 19.

Table 10: Comparison of training efficiency across various Visual Foundation Models (VFMs) during Stages 1 and 2. Stage 1 is conducted using the COCO dataset, while Stage 2 utilizes ImageNet-22K for large-scale pretraining. All the experiments are conducted on 8 H100s.

Stages	Stage 1 Time (per epoch)	Stage 2 Time (per epoch)
DINOv2-L	4min 37s	10h 48min 46s
SigLIP-L	4min 19s	11h 22min 15s
CLIP-L	5min 56s	11h 09min 21s
IN1K-L	4min 34s	10h 02min 54s
Mean	**4min 51s**	**10h 55min 41s**

A.3.2 Performance on Smaller Size VFMs

Apart from the results on DINOv2-L, we also report experiments on DINOv2-B in Tab. 11 for comparison. The results show that the performance gap between the original softmax model and our ViT-AdaLA is small, demonstrating the effectiveness of our method across different model sizes.

Table 11: Performance comparison using DINOv2-B on ADE20K, Cityscapes and ImageNet-1K. We pretrained our ViT-AdaLA (DINOv2-B) for 40 epochs.

Method	ADE20K (mIoU)	Cityscapes (mIoU)	IN1K (Acc.)
Softmax (DINOv2-B)	53.93	79.73	84.93
ViT-AdaLA (DINOv2-B)	51.46	74.60	84.37

Table 12: Reference comparison for IN-1K ViT-L. Image counts cover ImageNet-22K Stage 2 for ViT-AdaLA and ImageNet-1K pretraining for the scratch baseline. Data, teacher supervision, and compute are not matched.

Method	Images seen	ADE20K	Cityscapes	IN-1K
Softmax (IN-1K-L)	–	50.83	74.86	85.66
ViT-AdaLA (IN-1K-L)	283.9M	49.75	72.40	84.23
Linear ViT from scratch	256.2M	25.80	45.09	65.72

A.3.3 Comparison with Training from Scratch

We train a vanilla linear-attention ViT on ImageNet-1K for 200 epochs and compare its downstream results with an IN-1K-pretrained ViT-L adapted on ImageNet-22K for 20 Stage-2 epochs (Table 12). ImageNet-22K has approximately 14.20M images per epoch versus 1.28M for ImageNet-1K: these runs see roughly 283.9M and 256.2M training images, respectively. They are of comparable order in image exposures but are neither data- nor compute-matched: our method also uses a pretrained softmax teacher. The scratch curve had plateaued by 200 epochs and additional training gave no meaningful improvement. We use this run as a reference for the benefit of reusing the pretrained VFM, not as proof that adaptation always costs less.

A.4 Task Results and Batch-Size Scaling

We report additional classification and segmentation results, followed by batch-size efficiency.

A.4.1 Experiments on Classification Tasks

Tab. 13 presents the classification benchmarks for various linear attention models based on CLIP-L. From the results, it can be observed that approximation-based linear attention methods, *i.e.*, Monarch and Nyströmformer, significantly outperform other baselines. This demonstrates that a good approximation of softmax is advantageous. However, these two methods still cannot match the performance of our ViT-AdaLA or even the Stage 2 variant. This performance gap suggests that linearization-based approaches are more effective at distilling prior knowledge from pre-trained VFMs compared to traditional training-from-scratch paradigms.

Table 13: Top-1 fine-tuning accuracy comparison on ImageNet-1K under CLIP-L with multiple linear attention baselines. Native/direct conversions use the same downstream full fine-tuning budget; see Table 17. The classification head is a single linear layer for all the methods.

	Backbone	**Res.**	**Params (M)**	**FLOPS (G)**	**Peak Mem. (GB)**	**Throughput (imgs/s)**	**Top-1 Acc. (%)**
Softmax [39]	CLIP-L	512^2	304.20	310.60	1.3181	36.52	86.8
Hedgehog [60]	CLIP-L	512^2	305.77	$265.41_{\downarrow 14.5\%}$	$1.2221_{\downarrow 7.3\%}$	$37.44_{\uparrow 2.5\%}$	67.4
LoLCATS [61]	CLIP-L	512^2	305.77	$265.41_{\downarrow 14.5\%}$	$1.2221_{\downarrow 7.3\%}$	$37.44_{\uparrow 2.5\%}$	67.21
Linformer [50]	CLIP-L	512^2	305.77	$262.19_{\downarrow 15.6\%}$	$1.2221_{\downarrow 7.3\%}$	$45.41_{\uparrow 25.3\%}$	56.8
Performer [10]	CLIP-L	512^2	304.20	$265.41_{\downarrow 14.6\%}$	$1.2164_{\downarrow 7.7\%}$	$36.48_{\uparrow 0.1\%}$	69.9
Cosformer [40]	CLIP-L	512^2	304.20	$265.41_{\downarrow 14.6\%}$	$1.2226_{\downarrow 7.2\%}$	$39.55_{\uparrow 8.2\%}$	70.6
Nyströmformer [53]	CLIP-L	512^2	304.20	$265.20_{\downarrow 14.6\%}$	$1.2163_{\downarrow 7.7\%}$	$18.29_{\downarrow 50.8\%}$	80.7
Monarch [57]	CLIP-L	512^2	304.20	$269.44_{\downarrow 13.3\%}$	$1.2304_{\downarrow 6.7\%}$	$18.31_{\downarrow 48.9\%}$	81.5
Stage 2 only	CLIP-L	512^2	304.20	$262.19_{\downarrow 15.6\%}$	$1.2163_{\downarrow 7.7\%}$	$41.56_{\uparrow 16.1\%}$	83.4
ViT-AdaLA (Ours)	CLIP-L	512^2	304.20	$262.19_{\downarrow 15.6\%}$	$1.2163_{\downarrow 7.7\%}$	$41.56_{\uparrow 16.1\%}$	85.5

A.4.2 Experiments on Segmentation Tasks

Tab. 15 and Tab. 14 present the segmentation results for ADE20K and Cityscapes, respectively. From the results we can see that the overall performance of linear attention baselines have larger gaps with the softmax upper bound compared to the classification results. This is because segmentation requires

stronger prior knowledge from the VFMs than the classification tasks do. Despite these challenges, ViT-AdaLA maintains competitive performance with softmax attention, demonstrating its superior ability to extract and preserve critical priors. Furthermore, ViT-AdaLA scales to high-resolution inputs with significant efficiency gains while suffering minimal performance degradation compared to the standard softmax attention.

Table 14: mIoU fine-tuning comparison on Cityscapes under different input resolutions (512 vs. 1024) based on IN1K ViT-L and SigLIP-L. Native/direct conversions use the same downstream full fine-tuning budget; see Table 17. The segmentation head is Mask2former for all the methods.

	Backbone	**Res.**	**Params (M)**	**FLOPS (G)**	**Peak Mem. (GB)**	**Throughput (imgs/s)**	**mIoU (%)**
Softmax [39]	IN1K ViT-L	512^2	304.15	310.60	1.3181	36.52	65.11
Hedgehog [60]	IN1K ViT-L	512^2	305.72	265.41$_{\downarrow 14.5\%}$	1.2221$_{\downarrow 7.3\%}$	37.44$_{\uparrow 2.5\%}$	37.69
LoLCATS [61]	IN1K ViT-L	512^2	305.72	265.41$_{\downarrow 14.5\%}$	1.2221$_{\downarrow 7.3\%}$	37.44$_{\uparrow 2.5\%}$	25.04
Linformer [50]	IN1K ViT-L	512^2	305.72	262.19$_{\downarrow 15.6\%}$	1.2221$_{\downarrow 7.3\%}$	45.41$_{\uparrow 25.3\%}$	17.30
Performer [10]	IN1K ViT-L	512^2	304.15	265.41$_{\downarrow 14.6\%}$	1.2164$_{\downarrow 7.7\%}$	36.48$_{\uparrow 0.1\%}$	39.44
Cosformer [40]	IN1K ViT-L	512^2	304.15	265.41$_{\downarrow 14.6\%}$	1.2226$_{\downarrow 7.2\%}$	39.55$_{\uparrow 8.2\%}$	39.33
Nyströmformer [53]	IN1K ViT-L	512^2	304.15	265.20$_{\downarrow 14.6\%}$	1.2163$_{\downarrow 7.7\%}$	18.29$_{\downarrow 50.8\%}$	45.58
Monarch [57]	IN1K ViT-L	512^2	304.15	269.44$_{\downarrow 13.3\%}$	1.2304$_{\downarrow 6.7\%}$	18.31$_{\downarrow 48.9\%}$	48.35
Stage 2 only	IN1K ViT-L	512^2	304.15	262.19$_{\downarrow 15.6\%}$	1.2163$_{\downarrow 7.7\%}$	41.56$_{\uparrow 16.1\%}$	62.24
ViT-AdaLA (Ours)	IN1K ViT-L	512^2	304.15	262.19$_{\downarrow 15.6\%}$	1.2163$_{\downarrow 7.7\%}$	41.56$_{\uparrow 16.1\%}$	64.16
Softmax [39]	IN1K ViT-L	1024^2	304.15	1241.0	3.2836	7.07	72.21
Hedgehog [60]	IN1K ViT-L	1024^2	305.72	442.24$_{\downarrow 64.4\%}$	1.3823$_{\downarrow 57.9\%}$	15.38$_{\uparrow 117.4\%}$	38.07
LoLCATS [61]	IN1K ViT-L	1024^2	305.72	442.24$_{\downarrow 64.4\%}$	1.3823$_{\downarrow 57.9\%}$	15.38$_{\uparrow 117.4\%}$	25.98
Linformer [50]	IN1K ViT-L	1024^2	305.72	429.35$_{\downarrow 65.4\%}$	1.3999$_{\downarrow 57.4\%}$	17.83$_{\uparrow 152.0\%}$	7.78
Performer [10]	IN1K ViT-L	1024^2	304.15	442.24$_{\downarrow 64.4\%}$	1.3765$_{\downarrow 58.1\%}$	15.23$_{\uparrow 115.3\%}$	40.86
Cosformer [40]	IN1K ViT-L	1024^2	304.15	442.24$_{\downarrow 64.4\%}$	1.3945$_{\downarrow 57.5\%}$	15.29$_{\uparrow 116.1\%}$	40.14
Nyströmformer [53]	IN1K ViT-L	1024^2	304.15	442.24$_{\downarrow 64.4\%}$	1.3764$_{\downarrow 58.1\%}$	14.61$_{\uparrow 106.5\%}$	43.15
Monarch [57]	IN1K ViT-L	1024^2	304.15	500.24$_{\downarrow 59.7\%}$	1.5469$_{\downarrow 52.9\%}$	9.74$_{\uparrow 37.6\%}$	52.18
Stage 2 only	IN1K ViT-L	1024^2	304.15	429.35$_{\downarrow 65.4\%}$	1.3764$_{\downarrow 58.1\%}$	15.95$_{\uparrow 125.4\%}$	66.42
ViT-AdaLA (Ours)	IN1K ViT-L	1024^2	304.15	429.35$_{\downarrow 65.4\%}$	1.3764$_{\downarrow 58.1\%}$	15.95$_{\uparrow 125.4\%}$	68.25
Softmax [39]	SigLIP-L	512^2	304.15	310.60	1.3181	36.52	70.57
Hedgehog [60]	SigLIP-L	512^2	305.72	265.41$_{\downarrow 14.5\%}$	1.2221$_{\downarrow 7.3\%}$	37.44$_{\uparrow 2.5\%}$	39.53
LoLCATS [61]	SigLIP-L	512^2	305.72	265.41$_{\downarrow 14.5\%}$	1.2221$_{\downarrow 7.3\%}$	37.44$_{\uparrow 2.5\%}$	23.09
Linformer [50]	SigLIP-L	512^2	305.72	262.19$_{\downarrow 15.6\%}$	1.2221$_{\downarrow 7.3\%}$	45.41$_{\uparrow 25.3\%}$	21.81
Performer [10]	SigLIP-L	512^2	304.15	265.41$_{\downarrow 14.6\%}$	1.2164$_{\downarrow 7.7\%}$	36.48$_{\uparrow 0.1\%}$	39.93
Cosformer [40]	SigLIP-L	512^2	304.15	265.41$_{\downarrow 14.6\%}$	1.2226$_{\downarrow 7.2\%}$	39.55$_{\uparrow 8.2\%}$	40.56
Nyströmformer [53]	SigLIP-L	512^2	304.15	265.20$_{\downarrow 14.6\%}$	1.2163$_{\downarrow 7.7\%}$	18.29$_{\downarrow 50.8\%}$	43.52
Monarch [57]	SigLIP-L	512^2	304.15	269.44$_{\downarrow 13.3\%}$	1.2304$_{\downarrow 6.7\%}$	18.31$_{\downarrow 48.9\%}$	43.94
Stage 2 only	SigLIP-L	512^2	304.15	262.19$_{\downarrow 15.6\%}$	1.2163$_{\downarrow 7.7\%}$	41.56$_{\uparrow 16.1\%}$	67.35
ViT-AdaLA (Ours)	SigLIP-L	512^2	304.15	262.19$_{\downarrow 15.6\%}$	1.2163$_{\downarrow 7.7\%}$	41.56$_{\uparrow 16.1\%}$	68.20
Softmax [39]	SigLIP-L	1024^2	304.15	1241.0	3.2836	7.07	76.53
Hedgehog [60]	SigLIP-L	1024^2	305.72	442.24$_{\downarrow 64.4\%}$	1.3823$_{\downarrow 57.9\%}$	15.38$_{\uparrow 117.4\%}$	40.17
LoLCATS [61]	SigLIP-L	1024^2	305.72	442.24$_{\downarrow 64.4\%}$	1.3823$_{\downarrow 57.9\%}$	15.38$_{\uparrow 117.4\%}$	24.56
Linformer [50]	SigLIP-L	1024^2	305.72	429.35$_{\downarrow 65.4\%}$	1.3999$_{\downarrow 57.4\%}$	17.83$_{\uparrow 152.0\%}$	6.92
Performer [10]	SigLIP-L	1024^2	304.15	442.24$_{\downarrow 64.4\%}$	1.3765$_{\downarrow 58.1\%}$	15.23$_{\uparrow 115.3\%}$	40.53
Cosformer [40]	SigLIP-L	1024^2	304.15	442.24$_{\downarrow 64.4\%}$	1.3945$_{\downarrow 57.5\%}$	15.29$_{\uparrow 116.1\%}$	39.90
Nyströmformer [53]	SigLIP-L	1024^2	304.15	442.24$_{\downarrow 64.4\%}$	1.3764$_{\downarrow 58.1\%}$	14.61$_{\uparrow 106.5\%}$	41.86
Monarch [57]	SigLIP-L	1024^2	304.15	500.24$_{\downarrow 59.7\%}$	1.5469$_{\downarrow 52.9\%}$	9.74$_{\uparrow 37.6\%}$	46.58
Stage 2 only	SigLIP-L	1024^2	304.15	429.35$_{\downarrow 65.4\%}$	1.3764$_{\downarrow 58.1\%}$	15.95$_{\uparrow 125.4\%}$	72.41
ViT-AdaLA (Ours)	SigLIP-L	1024^2	304.15	429.35$_{\downarrow 65.4\%}$	1.3764$_{\downarrow 58.1\%}$	15.95$_{\uparrow 125.4\%}$	73.33

Table 15: mIoU fine-tuning comparison on ADE20K under ViT-IN1K and SigLIP with multiple linear attention baselines. Native/direct conversions use the same downstream full fine-tuning budget; see Table 17. The segmentation head is Mask2former for all the methods.

	Backbone	Res.	Params (M)	FLOPS (G)	Peak Mem. (GB)	Throughput (imgs/s)	mIoU (%)
Softmax [39]	IN1K ViT-L	512^2	304.20	310.60	1.3181	36.52	50.83
Hedgehog [60]	IN1K ViT-L	512^2	305.77	265.41$_{\downarrow 14.5\%}$	1.2221$_{\downarrow 7.3\%}$	37.44$_{\uparrow 2.5\%}$	25.15
LoLCATS [61]	IN1K ViT-L	512^2	305.77	265.41$_{\downarrow 14.5\%}$	1.2221$_{\downarrow 7.3\%}$	37.44$_{\uparrow 2.5\%}$	20.17
Linformer [50]	IN1K ViT-L	512^2	305.77	262.19$_{\downarrow 15.6\%}$	1.2221$_{\downarrow 7.3\%}$	45.41$_{\uparrow 25.3\%}$	11.49
Performer [10]	IN1K ViT-L	512^2	304.20	265.41$_{\downarrow 14.6\%}$	1.2164$_{\downarrow 7.7\%}$	36.48$_{\uparrow 0.1\%}$	31.01
Cosformer [40]	IN1K ViT-L	512^2	304.20	265.41$_{\downarrow 14.6\%}$	1.2226$_{\downarrow 7.2\%}$	39.55$_{\uparrow 8.2\%}$	29.65
Nyströmformer [53]	IN1K ViT-L	512^2	304.20	265.20$_{\downarrow 14.6\%}$	1.2163$_{\downarrow 7.7\%}$	18.29$_{\downarrow 50.8\%}$	41.67
Monarch [57]	IN1K ViT-L	512^2	304.20	269.44$_{\downarrow 13.3\%}$	1.2304$_{\downarrow 6.7\%}$	18.31$_{\downarrow 48.9\%}$	45.05
Stage 2 only	IN1K ViT-L	512^2	304.20	262.19$_{\downarrow 15.6\%}$	1.2163$_{\downarrow 7.7\%}$	41.56$_{\uparrow 16.1\%}$	46.92
ViT-AdaLA (Ours)	IN1K ViT-L	512^2	304.20	262.19$_{\downarrow 15.6\%}$	1.2163$_{\downarrow 7.7\%}$	41.56$_{\uparrow 16.1\%}$	49.75
Softmax [39]	SigLIP-L	512^2	304.20	310.60	1.3181	36.52	54.4
Hedgehog [60]	SigLIP-L	512^2	305.77	265.41$_{\downarrow 14.5\%}$	1.2221$_{\downarrow 7.3\%}$	37.44$_{\uparrow 2.5\%}$	22.13
LoLCATS [61]	SigLIP-L	512^2	305.77	265.41$_{\downarrow 14.5\%}$	1.2221$_{\downarrow 7.3\%}$	37.44$_{\uparrow 2.5\%}$	20.11
Linformer [50]	SigLIP-L	512^2	305.77	262.19$_{\downarrow 15.6\%}$	1.2221$_{\downarrow 7.3\%}$	45.41$_{\uparrow 25.3\%}$	12.37
Performer [10]	SigLIP-L	512^2	304.20	265.41$_{\downarrow 14.6\%}$	1.2164$_{\downarrow 7.7\%}$	36.48$_{\uparrow 0.1\%}$	23.97
Cosformer [40]	SigLIP-L	512^2	304.20	265.41$_{\downarrow 14.6\%}$	1.2226$_{\downarrow 7.2\%}$	39.55$_{\uparrow 8.2\%}$	23.08
Nyströmformer [53]	SigLIP-L	512^2	304.20	265.20$_{\downarrow 14.6\%}$	1.2163$_{\downarrow 7.7\%}$	18.29$_{\downarrow 50.8\%}$	32.54
Monarch [57]	SigLIP-L	512^2	304.20	269.44$_{\downarrow 13.3\%}$	1.2304$_{\downarrow 6.7\%}$	18.31$_{\downarrow 48.9\%}$	38.87
Stage 2 only	SigLIP-L	512^2	304.20	262.19$_{\downarrow 15.6\%}$	1.2163$_{\downarrow 7.7\%}$	41.56$_{\uparrow 16.1\%}$	51.39
ViT-AdaLA (Ours)	SigLIP-L	512^2	304.20	262.19$_{\downarrow 15.6\%}$	1.2163$_{\downarrow 7.7\%}$	41.56$_{\uparrow 16.1\%}$	53.55

A.4.3 DINOv2-L efficiency performance with different batch sizes

To better illustrate efficiency in standard ViT settings, where token sequence lengths are relatively small, we conducted a detailed analysis in Tab. 16 by scaling the batch size (BS) from 1, the default batch size in our main paper, to 128 using a DINOv2-L backbone.

Table 16: DINOv2-L efficiency performance with different batch sizes. The input size is 256×256. Relative changes are reported with respect to Softmax attention under the same batch size.

Batch Size	Attention Type	Latency (ms)	Throughput (imgs/s)	Memory (GB)
1	Softmax	15.916	62.83	1.1770
	Linear	20.291 (+27.5%)	49.28 (−21.6%)	1.1750 (−0.2%)
16	Softmax	63.420	252.29	1.4100
	Linear	61.410 (−3.2%)	260.54 (+3.3%)	1.3625 (−3.4%)
32	Softmax	126.091	253.78	1.6595
	Linear	121.086 (−4.0%)	264.28 (+4.1%)	1.5624 (−5.8%)
64	Softmax	257.052	248.98	2.1535
	Linear	246.232 (−4.2%)	259.92 (+4.4%)	1.9653 (−8.7%)
128	Softmax	510.024	250.97	3.1430
	Linear	487.622 (−4.4%)	262.50 (+4.6%)	2.7621 (−12.1%)

Key Findings. At batch size one, linear attention is slower than softmax, but it gains throughput as the workload grows. At batch size 128, it is 4.6% faster and uses 12.1% less memory, consistent with lower attention-map memory pressure at larger batches.

A.5 Protocols and Transfer Validation

We clarify matched evaluation budgets and test transfer beyond the primary backbone and task.

A.5.1 Baseline Protocol and Adaptation Cost

The main baseline tables compare complete conversion procedures. Native attention baselines do not receive ViT-AdaLA's feature-alignment stage; every method uses the same downstream data, head, and full fine-tuning schedule within each task. The Stage-2-only ablation starts from direct linear replacement. Table 17 clarifies these protocols, and Table 18 reports matched alignment experiments on alternative modules. Stage 2 uses ImageNet-22K and is a substantial one-time adaptation step.

Table 17: Protocol for the main comparison tables. Native denotes each baseline's own conversion or local attention transfer, not ViT-AdaLA Stage 1.

Method group	Local conversion	ViT-AdaLA Stage 2	Same downstream budget
Softmax teacher	None	No	Yes
Hedgehog, LoLCATS	Native attention transfer	No	Yes
Linformer, Performer, Cosformer	Native/direct replacement	No	Yes
Nyströmformer, Monarch	Native/direct replacement	No	Yes
Vanilla linear, Stage 2 only	Direct replacement	Yes	Yes
ViT-AdaLA	Stage 1	Yes	Yes
Nyströmformer, Monarch + our alignment	Stage 1	Yes	Yes

Table 18: Applying the same Stage 1 attention alignment and Stage 2 feature alignment to different DINOv2-L attention modules. Native/direct variants use their original conversion procedures; all rows use the same downstream full fine-tuning budget.

Attention module	Stages 1+2	IN-1K top-1 (%)	ADE20K mIoU
Nyströmformer (native)	No	82.4	41.16
Nyströmformer + alignment	Yes	85.3	54.60
Monarch (native)	No	82.7	44.95
Monarch + alignment	Yes	85.7	54.90
Vanilla + alignment	Yes	**86.0**	**55.55**

Table 19 converts the measured per-epoch wall times in Table 10 into aggregate H100 GPU-hours. These figures cover Stages 1 and 2 only; downstream Stage 3 compute is task-dependent and is excluded.

Table 19: Measured adaptation cost on eight H100 GPUs, excluding downstream Stage 3. GPU-hours equal eight times wall-clock hours.

Backbone	Stage 1 epochs	Stage 1 GPU-h	Stage 2 epochs	Stage 2 GPU-h
DINOv2-L	4	2.46	30	2595.1
SigLIP-L	4	2.30	10	909.7
CLIP-L	4	3.16	15	1338.7
IN-1K ViT-L	4	2.44	20	1607.7

A.5.2 Held-Out Feature Organization

Frozen-backbone transfer is reported in Table 5; here we examine whether the teacher's feature-space structure is preserved without training a task head. We evaluate the DINOv2-L teacher, direct linear replacement, and the Stage-2 ViT-AdaLA checkpoint before downstream fine-tuning. The complete Flowers-102 test split (6,149 images) was unused in Stages 1–2 and hyperparameter selection. All models receive identical 512^2 inputs and aligned patch tokens; no head or model parameter is trained. Nearest neighbors use mean-pooled final patch features with $k = 20$. The metrics test global geometry, sample neighborhoods, spatial token correspondence, and class-cluster structure. High

Table 20: Feature-space agreement with the frozen softmax DINOv2-L teacher on held-out Flowers-102. CKA and nearest-neighbor overlap use pooled final features; cosine similarity uses aligned tokens.

Linearized model	CKA	NN overlap@20	Token cosine	Class-centroid corr.
Direct replacement	0.196	10.0%	0.014	0.098
ViT-AdaLA after Stage 2	**0.993**	**90.5%**	**0.940**	**0.997**

agreement does not imply exact preservation on every dataset; it supports the frozen-head transfer in Table 5.

A.6 Alignment Objectives and Teacher Ablations

We detail checkpoint selection and test alternative parameterizations, joint training, and teacher choices.

A.6.1 Alignment Objective and Checkpoint Selection

The Stage-1 objective sums per-block mean-squared errors rather than averaging over the number of blocks; each student attention module receives the corresponding frozen teacher input. Stage 2 instead updates the complete student to align its final features. The per-backbone epoch budgets and feature-loss weights λ are listed in Table 7; no separate early-stopping criterion is specified by the reported curves.

A.6.2 LoLCATS Parameterization and Joint-Stage Ablations

We compare LoLCATS with LoRA and with full-weight tuning under the same DINOv2-L downstream evaluation. Both omit task-agnostic Stage-2 feature alignment. Increased tuning capacity raises ImageNet-1K accuracy by 0.7 points and ADE20K mIoU by 0.40, far less than the gain obtained with global feature alignment. Stage 1 compares softmax and linear attention on identical

Table 21: LoLCATS tuning capacity versus the complete ViT-AdaLA pipeline on DINOv2-L. All rows use the same downstream full fine-tuning budget.

Method	Feature alignment	IN-1K top-1	ADE20K mIoU
LoLCATS, LoRA	No	61.6	17.42
LoLCATS, full weights	No	62.3	17.82
ViT-AdaLA	Yes	**86.0**	**55.55**

teacher inputs, whereas Stage 2 changes student intermediate features end to end. We tested a simultaneous proxy that relaxes the identical-input condition and combines the losses. On ADE20K with SigLIP-L, it offers almost no improvement over Stage 2 alone; the sequential pipeline performs better (Table 22).

Table 22: Sequential versus simultaneous alignment on ADE20K with SigLIP-L. The joint proxy relaxes the matched-input condition of Stage 1.

Training strategy	ADE20K mIoU
Stage 2 only	51.39
Simultaneous Stage 1 + Stage 2	51.42
Sequential Stage 1 $\rightarrow$ Stage 2	**53.16**

B More Visualization Results

Additional PCA visualizations are provided in Figures 13 and 14. Specifically, Figure 13 presents a comparative analysis with Monarch Attention, while Figure 14 illustrates the impact of Stage 1 and Stage 2 through ablation-based PCA results.

Fig. 13 shows that ViT-AdaLA more closely matches the prior features of VFMs than approximation-based linear attention methods such as Monarch, demonstrating its superior ability to distill prior knowledge from softmax-based VFMs. Fig. 14 analyzes the contribution of different stages, showing that Stage 2 effectively preserves most of the original VFM features, while combining both stages yields the strongest retention of prior knowledge.

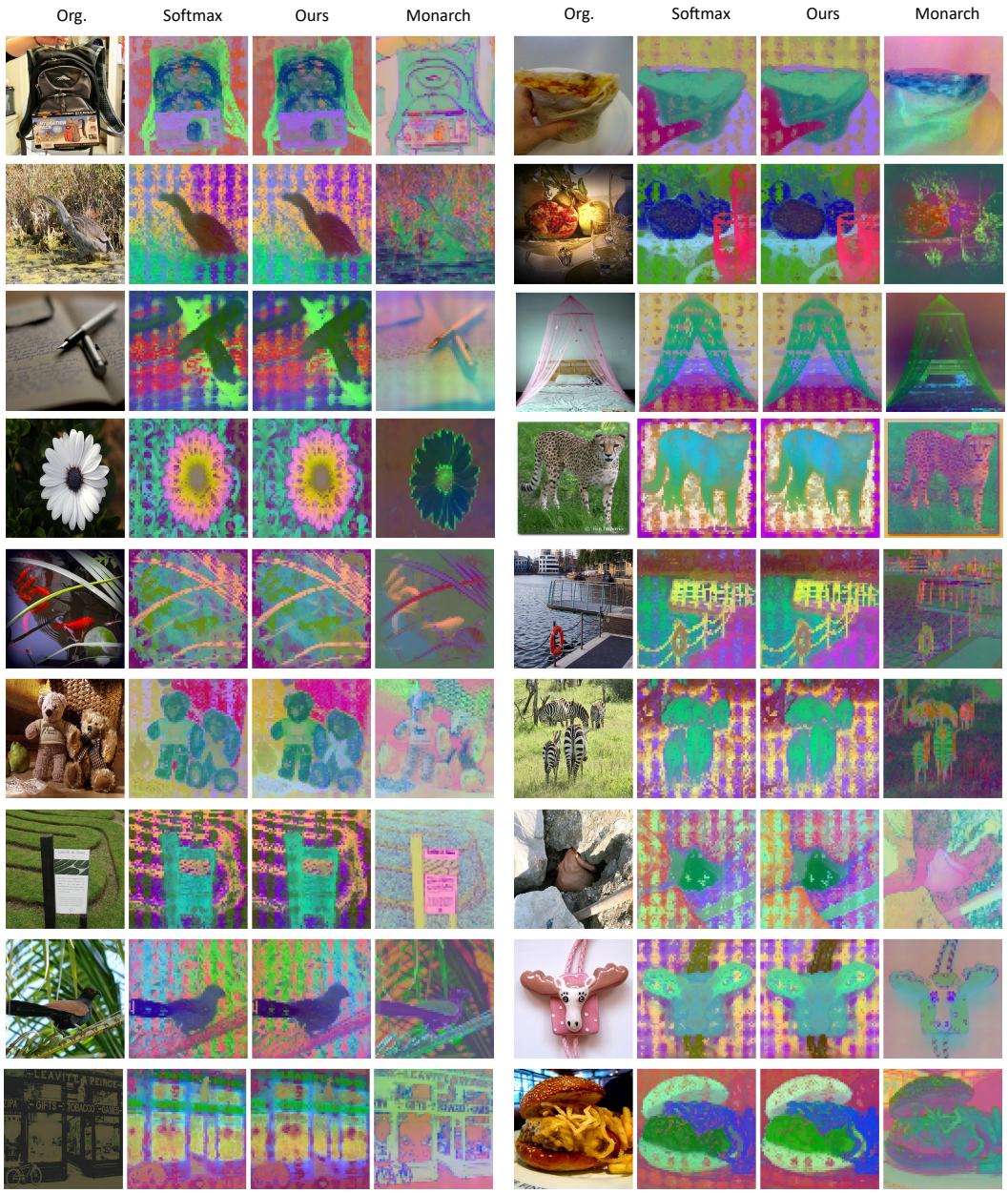


Figure 13: Visualization of PCA-projected features from the final layer of DINOv2-L. Original softmax features, ViT-AdaLA (ours) and Monarch Attention features are compared by projecting to three channels using PCA. These results indicate that ViT-AdaLA can learn more prior knowledge from the original VFM.

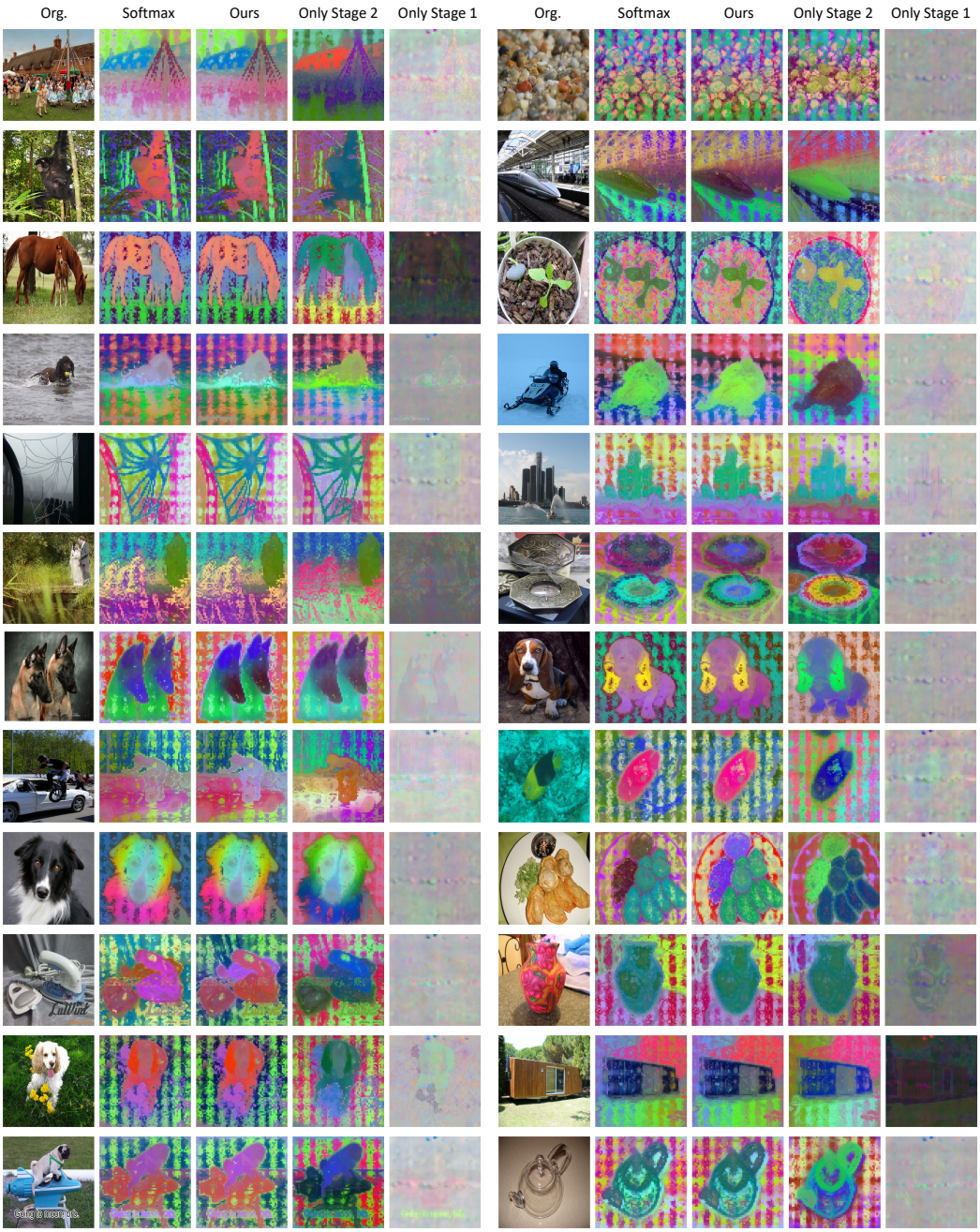


Figure 14: Visualization of PCA-projected features from the final layer of DINOv2-L. We ablate the Stages 1 and 2 training procedures and visualize the resulting PCA features for comparison. The results indicate that Stage 2 is crucial for extracting prior knowledge from VFMs, while the inclusion of Stage 1 further enhances this knowledge transfer.

C Limitations

ViT-AdaLA has several limitations. Its strongest measured inference gains arise in high-resolution dense prediction: at 256^2 and batch size one it is slower than softmax (Table 8). Stage 2 requires ImageNet-22K and substantial one-time H100 compute (Table 19); the scratch reference does not match its data or supervision, and complete totals also depend on task-specific Stage 3 fine-tuning. We evaluate classification, semantic segmentation, and one foreground-segmentation setting, leaving detection and generation untested. We do not report multi-seed uncertainty for the new ablations. Frozen-head transfer preserves most of the teacher's performance on the reported tasks, but this does not establish exact preservation of every feature or task. Improving adaptation efficiency and testing more tasks and backbones remain future work.

D Future Directions

In the future, this linear attention can be extended to vision large language models (VLLMs [32]), which process long visual sequences and thus incur substantial computational overhead. By replacing quadratic self-attention with linear variants, these models could achieve significantly improved scalability when handling high-resolution images or long video inputs. Beyond efficiency, this direction also opens up opportunities for better interpretability. For example, integrating explanation methods [6–8] with linear attention could provide more transparent insights into how visual tokens contribute to model predictions. This is particularly valuable in multimodal settings, where understanding cross-modal interactions remains challenging.

E Impact Statement

Our work focuses on adapting pretrained ViTs to linear attention while preserving their reusable representations. Its strongest measured inference benefits occur for long visual sequences, which could make high-resolution perception more practical. The one-time adaptation stage still requires substantial compute and should be included when assessing deployment costs.

From an ethical perspective, the main experiments use public datasets, while the SAM-H foreground-segmentation ablation uses an internal training set and SAM-HQ for evaluation. The method inherits risks associated with foundation models, including pretraining-data bias and misuse in privacy-sensitive applications. Efficient deployment can broaden both benefits and risks; dataset auditing and responsible use remain important.

Looking forward, we hope our work encourages further research into efficient and adaptable vision models that balance performance, scalability, and responsible deployment. We do not foresee any immediate negative societal impacts arising uniquely from this work beyond those already associated with large-scale visual recognition systems.

NeurIPS Paper Checklist

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

- The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations can be found the App. C.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: This paper doesn't include theoretical results.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.

- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper fully disclose all the information to reproduce the main experiment results in the main content and appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We would like to open source our codes and data once our paper is accepted.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We have specify all the training and test details necessary to understand the results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Alghough we didn't provide the statistical metrics, we have run extensive experiments on large-scale datasets like ImageNet to demonstrate the statistical significance.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).

- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have provide sufficient information on computer resources to reproduce the results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in this paper totally confirm the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: This paper provide an extra impact statement section in the appendix to discuss the societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper doesn't include data or models that have a high risk of misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All the data and models in the paper are publicly available.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper doesn't include any new assets.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper doesn't involve any crowdsourcing experiments and research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper doesn't include any crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

Justification: The core method in this paper does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.