Finetuning Vision-Language-Action Models Requires Fewer Layers Than You Think

Gia-Binh Nguyen[1,2] **Trong-Bao Ho**[2] **Thien-Loc Ha**[2] **Khoa Vo**[3] **Philip Lund Møller**[4]
Quang T. Nguyen[2] **Long Dinh**[1,2] **Tuan Dam**[5] **Vu Duong**[1] **Tung M. Luu**[6] **Trung Le**[7]
Tran Nguyen Le[4] **Minh Vu**[1,2] **An Thai Le**[1,2,13] **Ngan Le**[3] **Daniel Sonntag**[8,9]
James Zou[12] **Jan Peters**[9,13] **Duy M. H. Nguyen**[†,9,10,11] **Ngo Anh Vien**[†,1,2]

[1]Center for AI Research, VinUniversity [2]VinRobotics [3]University of Arkansas
[4]Technical University of Denmark [5] Hanoi University of Science and Technology [6]KAIST
[7]Monash University [8]Oldenburg University [9]DFKI [10]University of Stuttgart
[11]IMPRS-IS [12]Stanford University [13]Technische Universität Darmstadt
[†]Project Leads.

Abstract: Vision-Language-Action (VLA) models pre-trained on massive video-robot datasets have revolutionized robotic manipulation, yet their multi-billion parameter architectures impose prohibitive computational burdens during downstream fine-tuning and real-time inference. In this work, we reveal a highly non-trivial architectural characteristic of these continuous control foundation policies (e.g., π_0, GR00T-N1.5): despite being trained on diverse physical trajectories, they exhibit severe layer-wise representational redundancy. To exploit this, we introduce a structural compression pipeline that is entirely training-free, bypassing the need of existing methods to load full-scale models to learn optimized token reductions or dynamic layer selectors. Instead, using only a single forward pass via Centered Kernel Alignment to identify redundant layer features, we remove twin layers to permanently compress the model depth by up to 50% across both the VLM backbone and the continuous control policy head. Downstream fine-tuning of this streamlined architecture yields a dual acceleration benefit: a 40–50% reduction in training time and up to 30% faster real-time inference, while matching or exceeding full-scale base model performance. We comprehensively validate our method across **three simulation benchmarks** (LIBERO, RoboCasa, SimplerEnv) and **10 diverse real-world manipulation tasks** across **4 unique robotic embodiments**. These results prove that advanced VLAs require significantly fewer layers than previously assumed, offering a highly compute-efficient paradigm for scalable robot learning. Project page: https://clpvla.github.io/

Keywords: Vision-Language-Action, Model Optimization, Transformer Pruning

1 Introduction

Recent years have witnessed a major paradigm shift in robotic manipulation driven by the emergence of Vision-Language-Action (VLA) models. By framing physical control as a multimodal translation problem, early architectures like RT-2 [1], OpenVLA [2], and CogACT [3] demonstrated remarkable generalization capabilities across language-conditioned tasks. More recently, state-of-the-art (SOTA) architectures have transitioned toward continuous action generation using diffusion or flow-matching objectives, such as π_0 [4] and the GR00T family [5], yielding significantly more robust trajectory generation, smoother control paths, and tighter physics-priors. However, these architectural *breakthroughs are tightly coupled with massive scale*; modern generalist VLA policies regularly span *billions of parameters*. This extreme scale introduces a severe computational bottleneck, manifesting as staggering hardware and cluster time costs during downstream training, alongside high operational latency and heavy memory-bandwidth overhead that strictly constrain real-time edge inference.

To alleviate these computational costs, the embodied AI community has explored several distinct optimization strategies. One line of work accelerates action generation or visual processing through parallel decoding, decoupling speculative execution, token pruning, and temporal caching (training-free), including OpenVLA-OFT [6], Knowledge Insulating [7], EfficientVLA [8], SpecPrune-VLA [9]. Another line designs lighter VLA architectures from scratch, such as RoboMamba [10], Flower-VLA [11], SmolVLA [12], and NORA [13]. These models improve accessibility, but may not fully retain the broad capabilities inherited from large-scale pretrained VLA backbones. A third direction modifies pretrained models through training-adaptive computation, such as DeeR-VLA [14]and Mole-VLA [15], which learn to exit early or dynamically route inputs through a subset of layers. While promising, these approaches often require auxiliary routing modules, additional training objectives, or task-dependent runtime decisions.

Despite promising efficiency gains, existing VLA optimization frameworks face three critical bottlenecks at the intersection of model compression and physical deployment. First, *the architectural and evaluation scope remains narrow*: current techniques focus on older, text-token autoregressive baselines (e.g., OpenVLA) rather than modern continuous-control models (π_0, GR00T-N1.5), with real-world validation often restricted to 1–2 simple tasks [8, 16]. Second, *training-free methods fail to accelerate the expensive downstream fine-tuning phase*, which remains a major bottleneck in robot learning; for instance, a typical training on Libero often requires up to 20 hours on 4$\times$ A100 GPUs to reach convergence [17, 18]. Third, while *training-adaptive methods* enable faster training and inference, they *introduce immense architectural complexity* via auxiliary routing subcomponents or distillation pipelines [15, 14], fundamentally altering the core model structure and creating friction with downstream learning algorithms. Together, these challenges raise a key question: *"Can redundant layers be removed before fine-tuning without sacrificing policy performance?"*

We solve these challenges via a structurally clean and efficient framework that *eliminates the need to maintain full backbone depth during fine-tuning*, specifically targeting modern continuous control foundation models such as π_0 and GR00T-N1.5. Our key observation is that consecutive transformer blocks in modern VLA backbones often produce highly correlated token representations, suggesting substantial depth-wise redundancy. Specifically, we propose CLP (**C**KA-guided **L**ayer **P**runing), which performs a single forward pass and uses Centered Kernel Alignment (CKA) [19, 20, 21] to identify groups of representationally redundant layers. These layers are then permanently removed before fine-tuning. Unlike token pruning, caching, speculative decoding, or dynamic routing, CLP produces a statically smaller model that reduces active parameters, memory usage, training cost, and inference latency without adding new modules or auxiliary objectives.

We comprehensively validate our framework across three popular backbones (π_0, GR00T-N1.5, SmolVLA) over *3 simulation benchmarks* (LIBERO [22], RoboCasa [23], SimplerEnv [24]) and *10 demanding real-world manipulation tasks* spanning *4 distinct robotic embodiments* (Aloha Single and bimanual arms [25], UR10 and UR5 [26, 27]). In simulation, our method eliminates roughly 30% of trainable parameters and training time while maintaining competitive success rates. In real-world physical deployments, our streamlined architecture matches or even surpasses full-scale baselines. Crucially, this structural compression acts as an effective regularizer in *data-scarce regimes*: restricting training to only 10% of LIBERO data lifts success rates from 77.7% to 84.6%, and on real-world tasks with limited datasets of just 100 demonstrations, it delivers a *15% to 20% performance boost* over full-scale models.

Our main contributions are summarized as follows:

- **Accessible Adaptation of SOTA VLAs:** We demonstrate that SOTA continuous-control foundations (such as π_0 and GR00T-N1.5) can be fine-tuned with substantially reduced depth, lowering memory, training, and inference cost without introducing auxiliary routing or early-exit modules.
- **Pre-Finetuning Backbone Compression**. To this end, we propose CLP, a simple calibration-based framework that identifies representationally redundant transformer blocks using CKA and removes them before downstream adaptation.

- **Diversified Multi-Embodiment Validation**: We conduct extensive evaluations across multiple simulation benchmarks and 10 real-world manipulation tasks spanning four robotic embodiments, demonstrating that our non-invasive depth reduction is robust, platform-agnostic, and highly generalizable. Beyond achieving competitive performance against prior acceleration methods, CLP consistently improves sample efficiency under limited-data regimes, acting as an effective structural regularizer.

2 Related Work

VLA Models and Efficient Architectures. Current VLA frameworks primarily split into two paradigms: autoregressive architectures that output discrete action tokens (e.g., RT-1/2, OpenVLA, SpatialVLA) and diffusion- or flow-matching-based models that generate continuous control sequences (e.g., Octo, CogACT, π_0, GR00T-N1.5). While both exhibit exceptional generalization, their deep transformer backbones incur severe computational overhead. This bottleneck has motivated a growing line of research focused on training inherently smaller, high-capability VLA models from scratch. For instance, RoboMamba [10] integrates linear-complexity State Space Models (SSMs) [28] to achieve a multi-fold speedup in inference over attention-heavy baselines. Concurrently, lightweight frameworks like FLOWER-VLA [11], SmolVLA [12] and NORA [13] leverage compact multi-modal backbones (such as Florence-2 [29] or Qwen-2.5-VL [30]) and optimized action-token designs to maximize task performance while maintaining a restricted parameter footprint.

While these approaches improve accessibility, they typically require training or designing a new compact model from scratch and may not fully inherit the broad capabilities of large pretrained VLA foundations. In contrast, our method is complementary: we directly compress existing pretrained VLAs by removing redundant transformer layers before downstream fine-tuning.

Acceleration Techniques for VLA Model. Recent VLA efficiency methods can be further categorized into :*(i) Training-free method* and *(ii) training-adaptive ones*. The first one optimizes computational density by filtering token sequences using multi-modal and kinetic cues: VLA-Cache [16] exploits temporal frame continuity to adaptively cache and reuse token KV states [31, 32]; Efficient-VLA [8] distills a compact, spatially diverse token subset prioritized by task relevance; SpecPrune-VLA [9] executes a speculative dual-level pruning that eliminates background tokens using historical attention and depth-dependent importance; and ADP [33] dynamically gates token dropping by coupling text semantics directly with real-time end-effector kinetics. Collectively, these frameworks shift the computational burden by adaptively matching token processing to the immediate demands of the manipulation environment.

Conversely, training-adaptive frameworks structurally modify or expand the network to enforce computational sparsity, balancing efficiency and policy performance through targeted optimization. For instance, MoLe-VLA [15] integrates sparse routing layers to activate task-specific sub-networks. Moving depth-wise, DeeR-VLA [14] embeds intermediate prediction heads to enable dynamic early-exiting for simpler environmental states, while AC^2-VLA [34] employs an action-prior router to jointly govern token dropping and layer skipping. Though potent, these methods require substantial architectural modifications and training overhead to stabilize their dynamic routing mechanisms.

The CLP accelerates both downstream training and real-time inference, unlike inference-only training-free methods. By avoiding multi-stage distillation or auxiliary adaptive modules, such as the training-adaptive one, it also enables simple integration of advanced VLA adaptation, such as future knowledge prediction [17, 35] and chain-of-thought reasoning [36, 37], while remaining highly effective in low-data regimes where additional trainable components often overfit (Figure3-c).

3 Preliminaries

Vision-Language-Action Model. Let $\mathbf{x}^{\text{lang}}$ denote a text instruction and $\mathbf{x}^{\text{img}} \in \mathbb{R}^{H\times W\times 3}$ denote the RGB observation. A continuous-action VLA policy predicts an action chunk $\mathbf{a} \in \mathbb{R}^{T_a\times d_a}$ conditioned on these multimodal inputs. State-of-the-art continuous-control foundations (e.g., π_0 [4], GR00T [5],

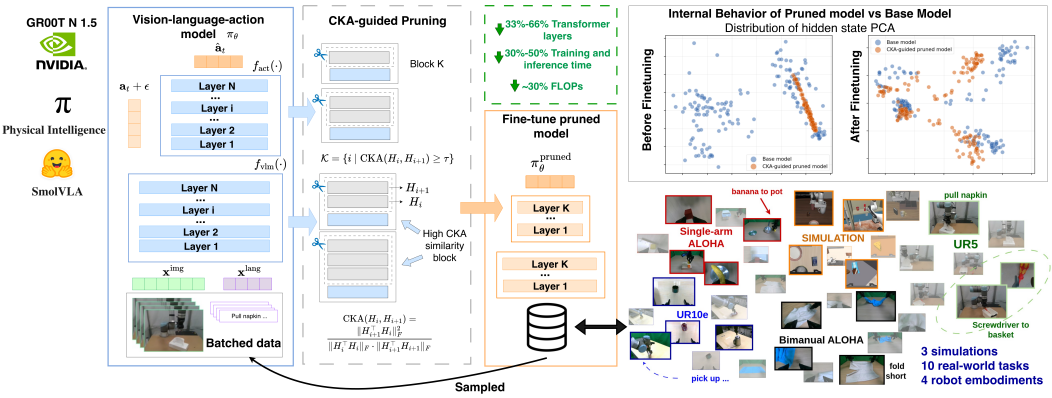

Figure 1: **Overview of the proposed CLP framework.** CLP prunes representationally redundant transformer layers via CKA, reducing network depth by up to 66% and training/inference cost by up to 50%. Fine-tuning restores the latent geometry of the compressed model, enabling competitive performance across three simulation benchmarks, 10 real-world tasks, and four robotic embodiments.

SmolVLA [12]) generally share a decoupled architecture: a Vision-Language Model (VLM) backbone that extracts environmental context, followed by a flow- or diffusion-based action-generation head.

VLM Backbone. Following modern efficiency conventions [33], the VLM backbone is modeled as a stack of N_v transformer layers. Let H_ℓ^{vlm} denote the hidden token representation after the ℓ-th VLM layer, and let F_ℓ^{vlm} denote the corresponding transformer block. The input hidden state H_0^{vlm} is obtained by embedding the language and visual observations, and the final context representation is $Z = H_{N_v}^{\text{vlm}}$:

$$H_0^{\text{vlm}} = \text{Embed}\left(\mathbf{x}^{\text{lang}}, \mathbf{x}^{\text{img}}\right), \quad H_\ell^{\text{vlm}} = F_\ell^{\text{vlm}}\left(H_{\ell-1}^{\text{vlm}}\right) \quad \forall \ell \in \{1, \ldots, N_v\}. \tag{1}$$

Action-Generation Head. For flow-matching-based generation [38], the action head parameterizes a velocity field to transport a Gaussian noise vector $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ toward the target action $\mathbf{a}$ along a linear interpolation path $\mathbf{a}_t = (1-t)\epsilon + t\mathbf{a}$ for time step $t \in [0, 1]$. Let H_m^{act} denote the hidden action-token representation after the m-th action layer, and let F_m^{act} denote the corresponding transformer block in an action head with N_a layers. The forward pass is formalized as:

$$H_0^{\text{act}} = \text{Embed}_{\text{act}}\left(\mathbf{a}_t, t\right), \quad H_m^{\text{act}} = F_m^{\text{act}}\left(H_{m-1}^{\text{act}}; \Phi_m(Z)\right) \quad \forall m \in \{1, \ldots, N_a\}, \tag{2}$$

where $\Phi_m(Z)$ represents the cross-conditioning signal derived from the VLM context Z and injected into the m-th action layer (e.g., via decoder cross-attention or token prefixing).

The final layer output designates the predicted velocity field $\hat{\mathbf{u}}_t = f_{\text{act}}(Z, \mathbf{a}_t, t) = H_{N_a}^{\text{act}}$, which is optimized via the Flow Matching objective: $\mathcal{L}_{\text{FM}} = \mathbb{E}_{t,\mathbf{a},\epsilon}\left[\left\|f_{\text{act}}\left(Z, \mathbf{a}_t, t\right) - (\mathbf{a} - \epsilon)\right\|_2^2\right]$. Despite varying cross-conditioning strategies (Φ_m) across baselines, this shared formulation frames both the VLM and action modules as deep transformer blocks, making their intermediate hidden states structurally compatible for layer-wise similarity analysis.

Centered Kernel Alignment. CKA [19] quantifies representation similarity between layers while remaining invariant to orthogonal transformations and isotropic scaling, making it ideal for identifying structural redundancies [39, 40, 41]. Given hidden states $H_i, H_j \in \mathbb{R}^{n \times d}$ across n tokens and d dimensions, the alignment of their Gram matrices ($K = HH^\top$) via the centered Hilbert-Schmidt Independence Criterion (HSIC) reduces for centered linear kernels to a ratio of Frobenius norms:

$$\text{CKA}(H_i, H_j) = \frac{\text{HSIC}(K_i, K_j)}{\sqrt{\text{HSIC}(K_i, K_i) \cdot \text{HSIC}(K_j, K_j)}} = \frac{\left|H_j^\top H_i\right|_F^2}{\left|H_i^\top H_i\right|_F \cdot \left|H_j^\top H_j\right|_F}. \tag{3}$$

The score bounded within $[0, 1]$ indicates representational similarity as it approaches 1. In our framework, a high CKA score between adjacent VLA layers signals minimal feature transformation, rendering those blocks prime candidates for structured pruning.

4 CKA-Guided Layer Pruning

Representational Plateaus: Diagnosing Layer Redundancy in Deep VLAs. To better understand the internal mechanics driving deep VLA architectures, we initiate an empirical exploration into a central question: *"how does information actually evolve across network depth?"*. Leveraging the CKA metric established in Section 3, we trace the hidden state trajectories of representative continuous-control foundations as they map multimodal context to physical actions. This tracking reveals a striking structural phenomenon: rather than progressing uniformly or incrementally, the feature representations exhibit distinct zones of stagnation, uncovering widespread layer redundancy across both the backbone and the action head.

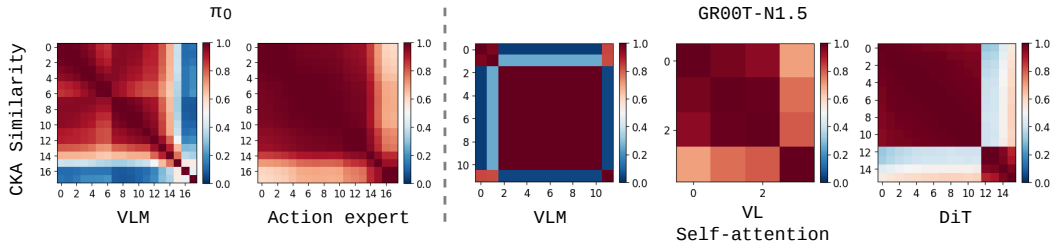

Figure 2: **CKA similarity profiles across π_0 and GR00T-N1.5 sub-modules.** The heatmaps illustrate pairwise representation alignment among transformer layers inside the VLM backbones, action heads, and DiT blocks. The extensive, contiguous plateaus of high similarity (dark red) across both model families signify minimal representational changes between successive layers, pinpointing candidate zones for structured pruning.

As illustrated in Figure 2, this behavior manifests in both π_0 and GR00T-N1.5 as large, contiguous blocks of high inter-layer similarity, indicating that consecutive layers perform highly redundant operations while major feature transformations are concentrated within a few distinct transitions. This pervasive redundancy motivates a concrete structural hypothesis: *these high-similarity regions represent layers that can potentially be removed without degrading downstream policy performance*. However, because high representational similarity alone does not inherently guarantee stability, we only utilize these CKA profiles as an empirical diagnostic to isolate candidate blocks, verifying their pruning viability through structured layer removal and subsequent low-data fine-tuning.

CKA-Guided Pruning. Given a pre-trained VLA policy, we target the structured removal of transformer layers within a prunable module $\mathcal{M}$ (e.g., the VLM backbone or action head). Let $\mathcal{I}_{\mathcal{M}} = \{1, \ldots, L_{\mathcal{M}}\}$ index the ordered layers of $\mathcal{M}$. For a target pruning budget $k_{\mathcal{M}}$, we isolate a removal set $\mathcal{R}_{\mathcal{M}} \subset \mathcal{I}_{\mathcal{M}}$ ($|\mathcal{R}_{\mathcal{M}}| = k_{\mathcal{M}}$) to construct the compressed policy $\pi_\theta^{\text{pruned}} = \text{RemoveLayers}(\pi_\theta, \mathcal{R}_{\mathcal{M}})$. Because all transformer blocks within $\mathcal{M}$ share identical hidden dimensions, layer removal simply reconnects the remaining predecessor and successor blocks. This design enables direct fine-tuning under the native training objective without requiring auxiliary routing parameters, distillation losses, or architectural modifications.

We estimate representational redundancy using a compact calibration set $\mathcal{D}_{\text{cal}}$ sampled from training episodes. By executing a forward pass over $\mathcal{D}_{\text{cal}}$, we extract and concatenate the token representations across calibration examples to construct a unified layer activation matrix $\bar{H}_\ell^{\mathcal{M}}$ for each $\ell \in \mathcal{I}_{\mathcal{M}}$. We quantify sequential redundancy by computing CKA between consecutive layers:

$$s_\ell^{\mathcal{M}} = \text{CKA}\left(\bar{H}_{\ell-1}^{\mathcal{M}}, \bar{H}_\ell^{\mathcal{M}}\right), \quad \ell = 2, \ldots, L_{\mathcal{M}}. \tag{4}$$

An elevated similarity score $s_\ell^{\mathcal{M}} \approx 1.0$ indicates that layer ℓ introduces minimal representational change relative to layer $\ell - 1$, marking it as a candidate for structured pruning. To prevent the disjointed removal of isolated layers due to local sampling noise, we aggregate adjacent redundant layers into contiguous blocks. Successive layers are clustered into the same block if they satisfy: $s_\ell^{\mathcal{M}} \geq \tau, \quad \ell = 2, \ldots, L_{\mathcal{M}}$, where τ is a designated similarity threshold. This constraint partitions the module into a set of contiguous high-similarity blocks $\mathcal{B}_{\mathcal{M}} = \{B_1, \ldots, B_Q\}$. For each block B, we retain its initial layer $r(B)$ as a functional anchor and pool the remaining layers into a candidate

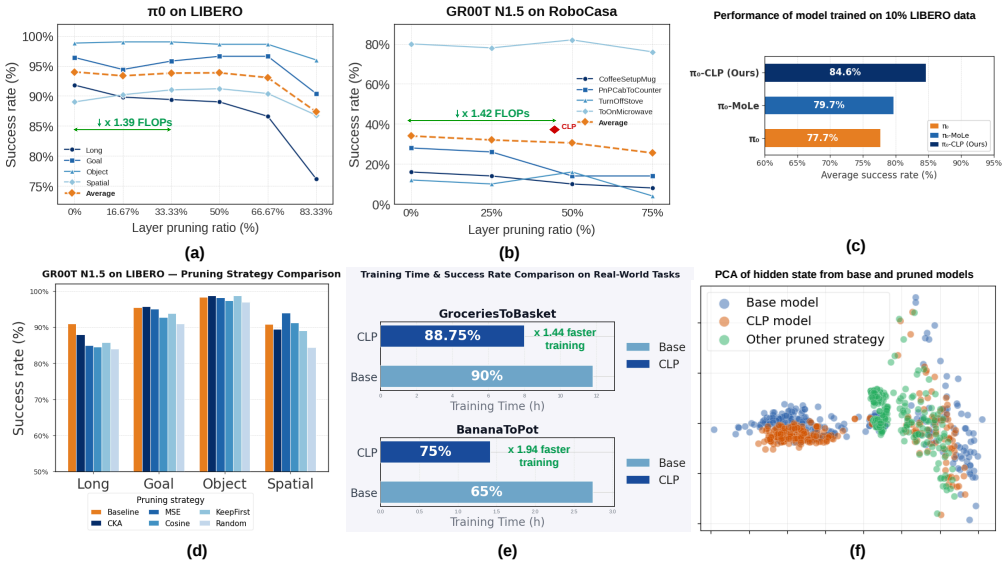

Figure 3: **Analysis of CLP evaluation across benchmarks and real-world tasks. (a)** Success rate of π_0 on LIBERO with different layer pruning ratio; **(b)** Success rate of GR00T N1.5 on RoboCasa across pruning ratios; **(c)** Comparison with dynamic layer skipping method (MoLe-VLA [15]) vs ours; **(d)** Comparison of different pruning strategies on GR00T N1.5 across LIBERO benchmark; **(e)** Training time and success rate on real-world manipulation tasks; **(f)** PCA visualization of hidden states (state/future tokens and action tokens) for the Base model, CKA-guided pruned model with different pruning strategies.

pruning set:

$$\mathcal{P}_{\mathcal{M}} = \bigcup_{B \in \mathcal{B}_{\mathcal{M}}} \left(B \setminus \{r(B)\} \right). \tag{5}$$

We calibrate τ to ensure the candidate pool satisfies $|\mathcal{P}_{\mathcal{M}}| \geq k_{\mathcal{M}}$. The final removal set $\mathcal{R}_{\mathcal{M}}$ is constructed by isolating the $k_{\mathcal{M}}$ most redundant layers within this pool:

$$\mathcal{R}_{\mathcal{M}} = \mathrm{TopK}_{\ell \in \mathcal{P}_{\mathcal{M}}} \left(s_{\ell}^{\mathcal{M}}, k_{\mathcal{M}} \right). \tag{6}$$

This targeted selection process is executed independently across each prunable module and formalized in Algorithm 1 (Appendix). The pruning routine is entirely static, permanently truncating network depth prior to downstream adaptation. Consequently, it shrinks both the fine-tuning training footprint and operational inference latency without introducing runtime computational overhead.

5 Experimental Results and Analysis

We conduct a diverse set of experiments to examine the performance of pruned VLA models trained with our `CLP` models across simulation and real-world settings and robot embodiment configurations. Details are presented in the Appendix. We first investigate the fundamental architectural properties of layer redundancy in VLAs and dissect the internal mechanisms of our framework (**RQ1** & **RQ2**). We then benchmark our method against standard baselines on large-scale manipulation suites (**RQ3**) and validate its physical feasibility on real-world robot platforms (**RQ4**):

- **RQ1 (Compression Trade-offs)**: To what extent can modern VLAs be structurally truncated before policy performance degrades?
- **RQ2 (Latent Behavior & Ablation)**: Is CKA uniquely optimal for identifying functional redundancy, and how does downstream adaptation reshape the pruned latent space?
- **RQ3 (Baseline Comparison)**: How does our static pruning framework compare against state-of-the-art training-free and training-adaptive baselines across varied data regimes?
- **RQ4 (Real-World Deployment)**: Do the computed FLOP savings translate directly to wall-clock training speedups and high-frequency real-time inference on physical hardware?

Table 1: **Efficiency comparison across models.** Model size, trainable parameters, training time (60000 steps), FLOPs, and inference speed on RTX 4070 reported on Libero benchmark.

Model	Model Size		Trainable Params		Training Time (hours)		GFLOPs		Inference Speed (ms)	
	Base	CLP	Base	CLP	Base	CLP	Base	CLP	Base	CLP
π_0	3.5B	2.7B ↓22.9%	3.1B	2.3B ↓25.8%	15.5	11.2 ↓27.8%	3073	2196.5 ↓28.5%	211	152 ↓27.9%
GR00TN1.5	2.7B	2B ↓25.9%	1.07B	0.75B ↓30.1%	10.7	7.4 ↓30.8%	1010	512.4 ↓49.3%	121	85 ↓29.8%
SmolVLA	450M	354M ↓21.3%	100M	63M ↓37%	24.75	8.83 ↓64.3%	598.4	536.1 ↓10.41%	201	137 ↓31.84%

RQ1. Compression Trade-Offs. We first investigate the limits of layer removal across distinct model families. As demonstrated in Figure 3, both π_0 on LIBERO (a) and GR00T-N1.5 on RoboCasa (b) exhibit *flat performance profiles up to a* 50% *pruning ratio*. This capability allows us to cut total FLOPs by $\times 1.39$ and $\times 1.42$ with π_0 and GR00T-N1.5, respectively, with negligible loss in policy success rate. These findings validate our core hypothesis: a substantial fraction of deep VLA layers are functionally redundant and can be statically removed prior to fine-tuning.

We also present in the Table 1 an overall efficiency reported across several factors such as model size, trainable parameters, training time, GFLOPs, and inference speed across three VLA backbones π_0, GR00T-N1.5, and SmolVLA [12]. It can be seen that CLP demonstrates generalizability across structurally diverse VLA backbones, uniformly compressing total *(i) model size* by 21.3%–25.9% and *(ii) cutting trainable parameters* by 25.8%–37.0% from sub-billion to multi-billion scales as well as (iii) saving GFLOPs up to 50% while preserving equivalent success rate (Table 2).

RQ2. Latent Behavior & Ablation. To understand this performance recovery, we analyze the hidden-state geometry of the compressed model using PCA (Figure 1, top-right). Before fine-tuning, structural pruning severely contracts the latent space, collapsing representations into a narrow subspace relative to the diverse distribution of the base model. After target-task adaptation, however, the remaining layers reorganize their feature pathways, restoring a representation manifold closely aligned with the original network. This geometric "manifold restoration" explains how the compressed policy regains expressive capacity while maintaining baseline-level manipulation performance.

Subsequently, we explore alternative block-selection CKA, including (i) Mean-squared error (MSE); (ii) COSINE similarity; (iii) RANDOM, where layers are selected uniformly at random, and finally (iv) KEEP-FIRST, in which the last k layers are removed, retaining only the earliest layers of the network. As shown in Figure 3-d, CKA consistently delivers the most stable performance across all benchmarks, closely matching the unpruned baseline while maintaining the highest average success rate. In contrast, localized similarity metrics and heuristic baselines can produce more unstable degradation, particularly on long-horizon and spatial tasks. Hidden-state analysis in Figure3-f further reveals that these alternatives distort representations into isolated subspaces, whereas CKA preserves the global topology necessary for effective post-pruning adaptation.

RQ3. State-of-the-art Compression Comparison. To evaluate the practical utility of CLP, we (i) compare with training-free pruning across multiple VLA settings. On LIBERO (Table 2), CLP consistently achieves a superior efficiency–performance trade-off, delivering 1.39–1.47$\times$ speedups while maintaining near-baseline success rates across three modern VLA backbones. Furthermore, it is important to note that these token-pruning methods primarily accelerate inference and *do not reduce the cost of downstream fine-tuning*, which remains a major bottleneck for VLA adaptation.

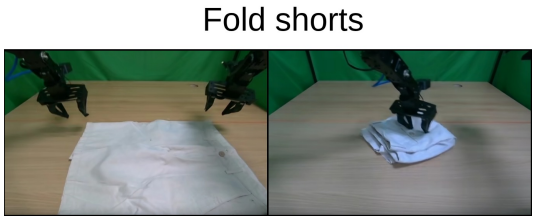

Figure 4: Robot experiments with folding shorts.

We further benchmark against (ii) trainable MoLe-VLA [15] in a challenging few-shot setting using only 10% of the LIBERO training data (Fig. 3-c). Despite its simplicity, CLP achieves an 84.6% average success rate, surpassing both the full π_0 baseline (77.7%) and π_0-MoLe (79.7%), while reducing training time by 1.38$\times$. Finally, on (iii) SimplerEnv with GR00TN1.5 (Tab. 3), CLP improves

the average success rate from 16.6% to 20.0% while shortening training time from 22.9 to 15.7 hours. These results show that structure-aware compression coupled with lightweight fine-tuning provides a more effective and scalable alternative to both inference-only pruning and parameter-expanding adaptive architectures, particularly in low-data regimes.

Table 2: LIBERO benchmark comparison with training-free acceleration methods.

Method	**Success Rate (%)**				**Avg. SR**	**Speedup ↑**
	Spatial	Object	Goal	Long	**(%)**	
OpenVLA-OFT [6]	97.6%	96.5%	97.9%	94.5%	96.6%	1.00×
FastV [42]	94.6%	95.8%	94.0%	88.8%	93.3%	1.44×
DivPrune [43]	92.4%	91.2%	89.0%	84.8%	89.4%	1.46×
EfficientVLA [8]	96.5%	91.1%	96.0%	72.1%	88.9%	1.52×
ADP [33]	97.6%	98.4%	97.4%	84.2%	94.4%	1.35×
π_0	94.6%	98.2%	95.4%	90.0%	94.6%	1.00×
π_0-SpecPrune-VLA [44]	96.6%	98.0%	95.2%	84.2%	93.5%	1.31×
π_0-CLP **(Ours)**	95.0%	99.2%	95.0%	86.4%	93.9%	1.39×
GR00T-N1.5	90.8%	98.4%	95.4%	91.0%	93.9%	1.00×
GR00T-N1.5-CLP **(Ours)**	89.4%	98.8%	95.8%	88.6%	93.0%	1.42×
SmolVLA	71.8%	92.2%	87.4%	57.2%	77.15%	1.00×
SmolVLA-CLP **(Ours)**	75.6%	93.0%	81.6%	56.2%	76.75%	1.47×

Table 3: **Evaluation of GR00T N1.5 on SimplerEnv**

Model	**Training Time (hours)**	**WidowX**							**Task(All) Avg.**
		Carrot Plate	Eggplant Basket	Spoon Towel	Stack Cube	Eggplant Sink	Close Drawer	Open Drawer	
GR00TN1.5	22.9	26	**34**	18	**8**	8	12	10	16.57
GR00TN1.5-CLP	15.7	**34**	14	**38**	4	**16**	**24**	**10**	**20**

RQ4. Real-World Deployment. We validate CLP through 10 real-world manipulation tasks spanning four robotic embodiments (UR10, UR5, single-arm ALOHA, and bimanual ALOHA) (Figures. 1 and 4). Despite reducing model depth, CLP maintains or improves overall task performance (Tab. 4), increasing the average success rate from 73.5% to 75.9% on GR00TN1.5 while outperforming the full model on several challenging tasks, including napkin serving (+20%), screwdriver placement (+15%), banana-to-pot transfer (+10%), and bimanual cloth folding (+5%). Importantly, these gains are accompanied by substantial reductions in fine-tuning time, achieving up to 1.94× faster training in physical deployments (Fig. 3-e and Appendix.)

We hypothesize that these gains arise from an implicit regularization effect: removing redundant layers reduces model capacity and discourages overfitting to task-specific noise. During fine-tuning, the remaining layers reorganize to recover the expressive latent structure of the original network, often yielding better adaptation under limited data. The consistency of these improvements across diverse tasks, objects, and robot embodiments demonstrates that CLP enhances efficiency without compromising real-world control capability.

Table 4: **Performance on Real-world manipulation tasks (GR00TN1.5).** Task success rate (%).

Model	**Single Arm**								**Bimanual**		**Avg.**
	UR10			**UR5**		**ALOHA Single Arm**			**ALOHA Bimanual**		
	Groceries ToBasket	Open Kettle	Close Kettle	Serve Napkin	Screwdriver ToBasket	Banana ToPot	Cube ToDrawer	Block Stacking	Fold Shorts	Fly Towel	
GR00TN1.5	**90**	**100**	**100**	45	15	65	**75**	**80**	90	**75**	73.5
Gr00N1.5-CLP	89	95	**100**	**65**	**30**	**75**	60	75	**95**	70	**75.9**

6 Conclusion

We demonstrate that a simple yet effective *CKA-guided Layer Pruning* strategy can substantially simplify state-of-the-art VLA models, including π_0 and GR00T-N1.5, without architectural modifications or adaptive routing mechanisms. Across extensive simulation and real-world evaluations, we show that modern VLA backbones contain significant structural redundancy, allowing 30–50% of transformer layers to be removed while maintaining competitive performance. Beyond achieving a favorable efficiency–performance trade-off, `CLP` consistently accelerates adaptation and exhibits strong gains in low-data and real-world robotic settings. We hope these findings encourage the community to rethink the necessity of ever-growing VLA architectures and inspire more efficient, scalable, and sustainable foundation policies.

7 Limitation

A key limitation of CLP is its use of a global pruning criterion that does not explicitly account for modality-specific token dynamics in manipulation tasks. Our analysis suggests that action and state tokens exhibit distinct representations, motivating future token- or modality-aware pruning strategies. In addition, the current work investigates CLP exclusively in the context of post-pretraining fine-tuning, leaving its application to the pretraining stage an open direction for future work. Rather than relying on heuristic layer selection as in existing approaches, CLP offers a principled mechanism for identifying which layers of a pretrained VLM are most beneficial to retain or adapt when transitioning to a VLA, suggesting its potential as a layer-selection prior during pretraining-stage adaptation.

References

[1] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

[2] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[3] Q. Li, Y. Liang, Z. Wang, L. Luo, X. Chen, M. Liao, F. Wei, Y. Deng, S. Xu, Y. Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

[4] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[5] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[6] M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[7] D. Driess, J. Springenberg, B. Ichter, L. Yu, A. Li-Bell, K. Pertsch, A. Ren, H. Walke, Q. Vuong, L. X. Shi, et al. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *Advances in Neural Information Processing Systems*, 38:102867–102888, 2026.

[8] Y. Yang, Y. Wang, Z. Wen, L. Zhongwei, C. Zou, Z. Zhang, C. Wen, and L. Zhang. Efficientvla: Training-free acceleration and compression for vision-language-action models. *Advances in Neural Information Processing Systems*, 38:40891–40914, 2026.

[9] H. Wang, J. Xu, Y. Xiang, J. Pan, Y. Zhou, Y.-L. Li, and G. Dai. Specprune-vla: Accelerating vision-language-action models via action-aware self-speculative pruning. *International Conference on Machine Learning*, 2026.

[10] J. Liu, M. Liu, Z. Wang, P. An, X. Li, K. Zhou, S. Yang, R. Zhang, Y. Guo, and S. Zhang. Robomamba: Efficient vision-language-action model for robotic reasoning and manipulation. *Advances in Neural Information Processing Systems*, 37:40085–40110, 2024.

[11] M. Reuss, H. Zhou, M. Rühle, Ö. E. Yağmurlu, F. Otto, and R. Lioutikov. Flower: Democratizing generalist robot policies with efficient vision-language-action flow policies. *Conference on Robot Learning (CoRL)*, 2025.

[12] M. Shukor, D. Aubakirova, F. Capuano, P. Kooijmans, S. Palma, A. Zouitine, M. Aractingi, C. Pascal, M. Russi, A. Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[13] C.-Y. Hung, Q. Sun, P. Hong, A. Zadeh, C. Li, U. Tan, N. Majumder, S. Poria, et al. Nora: A small open-sourced generalist vision language action model for embodied tasks. *arXiv preprint arXiv:2504.19854*, 2025.

[14] Y. Yue, Y. Wang, B. Kang, Y. Han, S. Wang, S. Song, J. Feng, and G. Huang. Deer-vla: Dynamic inference of multimodal large language models for efficient robot execution. *Advances in Neural Information Processing Systems*, 37:56619–56643, 2024.

[15] R. Zhang, M. Dong, Y. Zhang, L. Heng, X. Chi, G. Dai, L. Du, D. Wang, Y. Du, and S. Zhang. Mole-vla: Dynamic layer-skipping vision language action model via mixture-of-layers for efficient robot manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 18764–18772, 2026.

[16] S. Xu, Y. Wang, C. Xia, D. Zhu, T. Huang, and C. Xu. Vla-cache: Efficient vision-language-action manipulation via adaptive token caching. *Advances in Neural Information Processing Systems*, 38:164448–164473, 2026.

[17] D. M. Nguyen, N. T. Diep, B. G. Nguyen, T.-B. Ho, D. Le, T. Nguyen, T.-L. Ha, T. Nhiem, B. Thach, N. Tran, T. A. Tran, A. Habuda, P. L. Moeller, T. N. Le, D. Sonntag, M. Niepert, K. Doan, V. Duong, H. Ngo, M. Vu, D. M. Nguyen, A. Le, and V. Ngo. Foca: Future-oriented conditioning for data-efficient vision-language-action adaptation. In *Proceedings of the International Conference on Machine Learning (ICML)*, May 2026.

[18] M. Koo, D. Choi, T. Kim, K. Lee, C. Kim, Y. Seo, and J. Shin. Hamlet: Switch your vision-language-action model into a history-aware policy. *International Conference on Learning Representations (ICLR)*, 2025.

[19] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMlR, 2019.

[20] C. Cortes, M. Mohri, and A. Rostamizadeh. Algorithms for learning kernels based on centered alignment. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 301–309, 2012.

[21] T. Nguyen, M. Raghu, and S. Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *International Conference on Learning Representations (ICLR)*, 2021.

[22] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2023.

[23] S. Nasiriany, A. Maddukuri, L. Zhang, A. Parikh, A. Lo, A. Joshi, A. Mandlekar, and Y. Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems (RSS)*, 2024.

[24] X. Li, K. Gao, H. Zhou, A. Yu, Y. Zhu, H. Ku, S. Tao, J. Gu, S. Ha, X. Peng, and H. Su. Evaluating real-world robot manipulation policies in simulation. In *Conference on Robot Learning (CoRL)*, 2024.

[25] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[26] K. P. Hawkins. Analytic inverse kinematics for the universal robots UR-5/UR-10 arms. Technical report, Georgia Institute of Technology, 2013.

[27] *Universal Robots UR5/UR10 User Manual*. Universal Robots A/S, Odense, Denmark, 2013. Available at https://www.universal-robots.com.

[28] A. Gu and T. Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[29] B. Xiao, H. Wu, W. Xu, X. Dai, H. Hu, Y. Lu, M. Zeng, C. Liu, and L. Yuan. Florence-2: Advancing a unified representation for a variety of vision tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4818–4829, 2024.

[30] Q. Team. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[31] S. Ge, Y. Zhang, L. Liu, M. Zhang, J. Han, and J. Gao. Model tells you what to discard: Adaptive kv cache compression for llms. In *International Conference on Learning Representations*, volume 2024, pages 22975–22988, 2024.

[32] X. Zhou, W. Wang, M. Zeng, J. Guo, X. Liu, L. Shen, M. Zhang, and L. Ding. Dynamickv: Task-aware adaptive kv cache compression for long context llms. *arXiv preprint arXiv:2412.14838*, 2024.

[33] X. Pei, Y. Chen, S. Xu, Y. Wang, Y. Shi, and C. Xu. Action-aware dynamic pruning for efficient vision-language-action manipulation. *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026.

[34] W. Yu, T. Wang, F. Li, J. Li, and L. Zhu. Acˆ 2-vla: Action-context-aware adaptive computation in vision-language-action models for efficient robotic manipulation. *arXiv preprint arXiv:2601.19634*, 2026.

[35] G. R. Team, A. Abdolmaleki, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, A. Balakrishna, N. Batchelor, A. Bewley, J. Bingham, et al. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*, 2025.

[36] C.-P. Huang, Y.-H. Wu, M.-H. Chen, F. Wang, and F.-E. Yang. Thinkact: Vision-language-action reasoning via reinforced visual latent planning. *Advances in Neural Information Processing Systems*, 38:82782–82802, 2026.

[37] W. Zhang, H. Liu, Z. Qi, Y. Wang, X. Yu, J. Zhang, R. Dong, J. He, H. Wang, Z. Zhang, et al. Dreamvla: a vision-language-action model dreamed with comprehensive world knowledge. *Advances in Neural Information Processing Systems*, 38:24195–24228, 2026.

[38] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

[39] A. Gromov, K. Tirumala, H. Shapourian, P. Glorioso, and D. A. Roberts. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*, 2024.

[40] X. Men, M. Xu, Q. Zhang, Q. Yuan, B. Wang, H. Lin, Y. Lu, X. Han, and W. Chen. Shortgpt: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20192–20204, 2025.

[41] X. Chen, Y. Hu, J. Zhang, Y. Wang, C. Li, and H. Chen. Streamlining redundant layers to compress large language models. *arXiv preprint arXiv:2403.19135*, 2024.

[42] L. Chen, H. Zhao, T. Liu, S. Bai, J. Lin, C. Zhou, and B. Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pages 19–35. Springer, 2024.

[43] S. R. Alvar, G. Singh, M. Akbari, and Y. Zhang. Divprune: Diversity-based visual token pruning for large multimodal models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 9392–9401, 2025.

[44] S. Wang, R. Yu, Z. Yuan, C. Yu, F. Gao, Y. Wang, and D. F. Wong. Spec-vla: speculative decoding for vision-language-action models with relaxed acceptance. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 26916–26928, 2025.

Supplement to "Finetuning Vision-Language-Action Models Requires Fewer Layers Than You Think"

A Appendix

A.1 Implementation details

To systematically determine the redundancy across transformer layers, we implemented a data-driven pruning pipeline across both simulated and real-world environments. We sampled data batches from diverse datasets and recorded the intermediate hidden states across all network layers during the forward pass. Utilizing these representations, we computed Centered Kernel Alignment (CKA) matrices to quantify the representational similarity between different transformer layers.

Table 5: Summary of pruning configurations and layer distributions across the evaluated VLA models.

Model	Module	Number of transformer layers in original model	Number of transformer layers pruned	Pruned layer index
π_0	VLM and Action expert	18	12	1, 2, 4, 6, 8, 9
GR00T-N1.5	VLM	12	5	3, 4, 5, 6, 7, 8, 9
	VL-self-attention	4	3	2
	DiT Action head	16	8	1, 2, 4, 5, 6, 7, 10, 11
SmolVLA	VLM and Action expert	16	10	1, 2, 5, 6, 14, 15

Based on the resulting CKA similarity profiles, we defined a threshold, τ, to segment the network into distinct CKA blocks, as illustrated in Figure 1. For each identified block K, we retained only the first layer-hypothesizing that the initial layer is critical for processing and establishing the block's input representations - and pruned all subsequent layers within that same block. This structured pruning methodology was uniformly applied to three popular VLA models: π_0, GR00T-N1.5, and SmolVLA. The specific architectural configurations and post-pruning layer distributions for each model are summarized in Table 5.

Algorithm 1 CKA-Guided Layer Pruning

Require: Pre-trained policy π_θ, target module $\mathcal{M}$, calibration set $\mathcal{D}_{\text{cal}}$, budget $k_{\mathcal{M}}$, threshold τ

1: Extract calibrated hidden representations $\{\bar{H}_1^{\mathcal{M}}, \ldots, \bar{H}_{L_{\mathcal{M}}}^{\mathcal{M}}\}$ from $\mathcal{M}$ over $\mathcal{D}_{\text{cal}}$
2: **for** $\ell = 2$ to $L_{\mathcal{M}}$ **do**
3: $\quad s_\ell^{\mathcal{M}} \leftarrow \text{CKA}(\bar{H}_{\ell-1}^{\mathcal{M}}, \bar{H}_\ell^{\mathcal{M}})$
4: **end for**
5: Group consecutive layers into blocks $\mathcal{B}_{\mathcal{M}}$ where $s_\ell^{\mathcal{M}} \geq \tau$
6: Identify candidate pool $\mathcal{P}_{\mathcal{M}} \leftarrow \bigcup_{B \in \mathcal{B}_{\mathcal{M}}} (B \setminus \{r(B)\})$, where $r(B)$ is the initial layer of block B
7: Select removal set $\mathcal{R}_{\mathcal{M}} \leftarrow \text{TopK}_{\ell \in \mathcal{P}_{\mathcal{M}}}(s_\ell^{\mathcal{M}}, k_{\mathcal{M}})$
8: $\pi_\theta^{\text{pruned}} \leftarrow \text{RemoveLayers}(\pi_\theta, \mathcal{R}_{\mathcal{M}})$
9: **return** $\pi_\theta^{\text{pruned}}$

A.2 Additional Comparisons with Adaptive Training Compression Methods

In this section, we further compare CLP against training-adaptive baselines to demonstrate that CLP not only accelerates inference but also significantly reduces training cost. To ensure a fair comparison, we integrate dynamic layer skipping strategy from MoLe-VLA [15] on π_0 and train all models for the same number of steps. Unlike MoLe, which introduces additional trainable modules for dynamic layer selection and thus incurs even longer training times, CLP operates within a fixed pruned architecture requiring no extra parameters. Experiments on 10% of the LIBERO dataset and a 30-demonstration subset of ROBOCASA show that CLP achieves approximately 28% and 23% reduction in training time compared to the base model, respectively, while simultaneously improving average performance by 6.9% on LIBERO and 2.4% on ROBOCASA. These results suggest that pruning redundant layers not only streamlines inference but also reduces the optimization burden during fine-tuning, offering a more efficient training regime without sacrificing performance (Tables 6 and 9).

Table 6: Libero Benchmark Results training on 10% data (% Success Rate)

Model	**Long**	**Goal**	**Object**	**Spatial**	**Average**	**Training Hours**
π_0	58.8	87.8	82.6	81.6	**77.7**	15.5
π_0 - MoLe	60.2	88.2	86.0	84.4	**79.7**	15.6
π_0-CLP (Ours)	66.2	90.6	89.0	92.6	**84.6**	11.2

A.3 Real-world experiments details

In this work, we conduct experiments on 10 real-world tasks across 4 different robot embodiments. The collected tasks illustrated on Figure 6 cover a diverse range of manipulation skills, from single-arm to bimanual manipulation, and from rigid to deformable object handling, spanning data scales from 100 to 2800 demonstrations. Details are summarized in Table 7.

Table 7: Real-world task details.

Task	No. Episodes	Embodiment	No. Views	Task Description
Groceries To Basket	2800	UR10e	2	Pick up the tomato can/ coffee bag/ garlic powder/ ketchup/ mayonnaise/mustard/ olive oil/spam can and place them in the grey bin.
Open Kettle	300	UR10e	2	Open the lid of the electric kettle.
Close Kettle	300	UR10e	2	Close the lid of the electric kettle.
Serve Napkin	100	UR5	2	Pull napkin out from the box.
Screwdriver To Basket	100	UR5	2	Pick up screwdriver and place in box.
Banana in Pot	150	Single-arm ALOHA	2	Use the right gripper to pick up the banana and place it into the pot.Then pick up the lid and place it on top of the pot to close it.
Cube To Drawer	150	Single-arm ALOHA	2	Use the right gripper to open the top drawer, pick up the red square block, place it inside the top drawer, and then close the drawer.
Block Stacking	151	Single-arm ALOHA	2	Pick up the yellow triangular block and place it on top of the orange square block.
Fold Shorts	202	Bimanual ALOHA	3	Use both grippers to fold the shorts by bringing one leg over the other, then fold the bottom upward to form a compact shape.
Fly Towel	101	Bimanual ALOHA	3	Use both grippers to grasp two corners of the towel, lift it, and fling it onto the table so that it spreads out flat.

We fine-tuned both GR00T-N1.5 and GR00T-N1.5-CLP on a single NVIDIA H100 GPU with a batch size of 32. All hyperparameters were kept at their default values except `MAX_STEP`. Training steps and training time are reported in Table 8, and evaluation results are presented in Table 4.

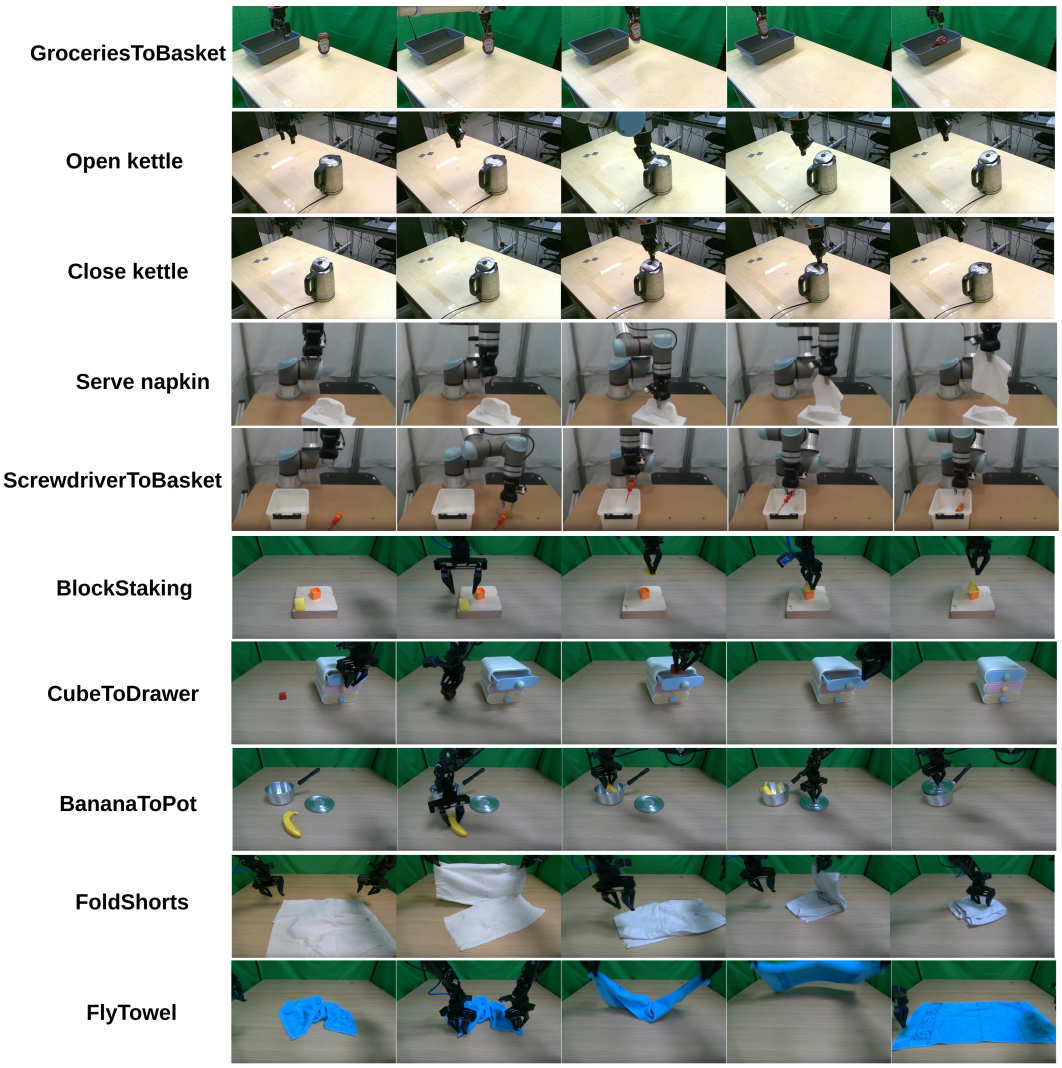

Figure 5: Real-world data samples

Figure 6: Simulation data samples

A.4 Simulation experiments details

LIBERO: We use LIBERO as a benchmark to evaluate our method against other baselines. A 10% subset is randomly sampled from the full LIBERO dataset. We fine-tune all models - π_0, GR00T-N1.5, and SmolVLA - on this subset for 100k steps with a global batch size of 64. We evaluate each task over 50 episodes with an execution length of 10 steps.Results are summarized in Table 2.

ROBOCASA: We used small subsets 30 demos and 100 demos of ROBOCASA and sampled demonstrations from 5 tasks covering diverse skills: *PnPCabToCounter*, *PnPCounterToCab*, *SetUpCoffeeMug*, *TurnOffStove*, and *TurnOnMicrowave*, drawn from the original 24-task dataset. We train both baseline and pruned models on this subset for 100k steps. The global batch size for π_0 is 48,

Table 8: Training step and training time on real-world datasets

Task	GR00T-N1.5		GR00T-N1.5-CLP	
	Training step	Training time (hours)	Training step	Training time (hours)
GroceriesToBasket	100k	11.8	100k	8
OpenKettle	24k	2.8	18k	1.4
CloseKettle	26k	3	19k	1.5
ServeNapkin	10k	1.1	10k	0.7
ScrewdriverToBasket	13k	1.5	13k	1.1
BananaToPot	30k	5.1	25k	2.9
CubeToDrawer	33k	5.6	28k	3.2
BlockStacking	24k	2.8	18k	1.4
FoldShorts	45k	6.5	45k	4.4
FlyTowel	13k	3.2	13k	2.1

fine-tuned on 4 H100 GPUs, while GR00T-N1.5 uses a global batch size of 32 on a single GPU. We evaluate each task over 50 episodes with an execution length of 10 steps. The results shown in Figure 3-b and Table 9

Table 9: Robocasa 30 demos results(% Success Rate)

Model	PnP Cab to Counter	PnP Counter to Cab	Coffee Setup Mug	Turn Off Stove	Turn On Microwave	Average	Training time (hours)
π_0 base	14	16	2	2	44	15.6	17.5
π_0- MoLe	14	18	2	4	50	17.6	17.7
π_0-CLP (Ours)	16	16	4	4	50	18	13.5

SimplerEnv: We fine-tune both the base and pruned GR00T-N1.5 models on the Bridge dataset on a single H100 GPU with batch size of 32 for 200k steps, then evaluate on SimplerEnv WidowX tasks. We evaluate each task over 50 episodes with an execution length of 8 steps. Results are summarized in Table 3.