

DIGS: Differentiable, Incremental, Global, Scalable Pruning for Language Models

Bingcong Li¹, Junlin Xian¹, Tao Jiang¹, Jianwen Lin¹^{*}, Cheng Zou¹,
and Gengli Zhang¹

Ant Group, Shenzhen, China

{libingcong.lbc,xianjunlin.xjl,jt467304}@antgroup.com

{aochen.ljw,zoucheng.zc,zhanggengli.zgl}@antgroup.com

Abstract. The rapid scaling of large language models (LLMs) and vision-language models (VLMs) has sharply increased compute and storage costs, motivating extensive research on model pruning. Yet most pruning pipelines underuse data scaling during pruning, i.e., increasing the pruning-phase token budget. We revisit L0-regularized structured channel pruning, valued for differentiability, incremental scheduling, and global decision-making. We uncover a long-horizon instability: the Lagrange multiplier grows during training, amplifies pruning gradients, and causes late-stage oscillations in both gradients and loss. We derive sufficient L2-norm bounds on pruning-specific gradients and reformulate the loss to enforce these bounds per update, replacing dual ascent with a coefficient determined by the gradient constraint. Building on this foundation, we introduce DIGS (Differentiable, Incremental, Global, Scalable pruning), which enables stable and efficient structured channel pruning without auxiliary stabilizers while supporting much larger pruning token budgets. Applied to the Qwen2.5VL LLM backbone, DIGS scales pruning beyond 100B tokens and achieves competitive results on public multi-modal benchmarks. Within this setting, larger pruning token budgets yield monotonic performance gains at fixed sparsity.

Keywords: LLM, VLM, Pruning

1 Introduction

As foundation models—including large language models (LLMs) and vision and language models (VLMs)—continue to scale, training and deployment costs have surged. Pruning has therefore become a central tool to reduce compute and memory while retaining capability [13, 41].

Prior work highlights three design choices that consistently improve pruning accuracy: (1) global, budget-aware decisions across layers to place sparsity where redundancy is highest [3, 21, 25]; (2) surrogate-differentiable gates so task-loss gradients directly shape pruning, outperforming proxy-based heuristics [8, 17, 20];

^{*} Corresponding author.

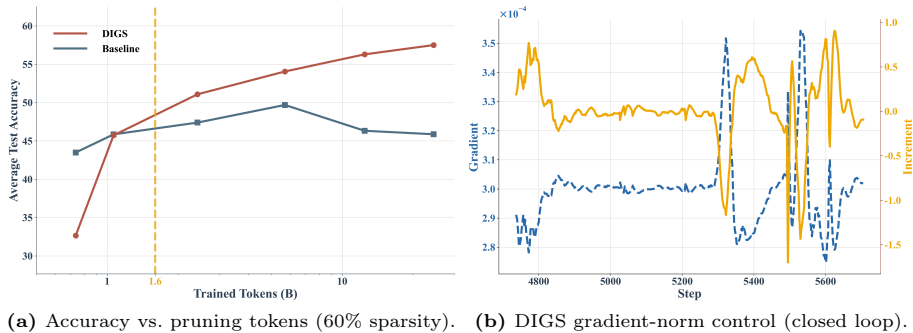


Fig. 1: Scaling benefits and stability with DIGS. (a) At 60% sparsity, DIGS (red) improves monotonically with the prune-phase token budget and surpasses L0-Dual (blue) from ~ 2.4 B tokens; by contrast, L0-Dual peaks near 5.7B tokens and then degrades. The orange dashed line marks 1.6B tokens—the upper end of prune-phase budgets explored in Sheared LLaMA under a fixed 5B-token overall budget (Table 5 of [37]). (b) DIGS bounds the per-step gate-gradient norm (left axis; scaled by 10^{-4}) via a closed-loop update on λ with target $c_g = 3 \times 10^{-4}$, keeping gradients bounded and preventing the late-stage oscillations observed with dual-ascent L0-Dual.

(3) incremental schedules that ramp sparsity gradually to avoid abrupt capacity collapse and forgetting [26, 33, 36–38] and (4) data scaling during pruning. Together, these choices preserve capacity where it matters and stabilize training.

The value of data scaling is method-dependent. Post-hoc, non-differentiable methods (e.g., Wanda [29] and SparseGPT [9]) typically saturate once local statistics are estimated. Thus, larger calibration sets bring diminishing returns. In contrast, training-based, surrogate-differentiable pruning can exploit larger token budgets because task-loss gradients continue to reshape weights and gates across layers, provided training remains stable over long horizons [26, 37].

L0-regularized structured channel pruning is a natural baseline [20]. It provides global budget awareness via a single sparsity constraint, is surrogate-differentiable through gating relaxations, and admits incremental schedules to ramp up sparsity [34, 37]. However, when pushed to long horizons, we consistently observe an instability: the Lagrange multiplier used to enforce the sparsity budget accumulates over time, amplifying pruning-specific gradients and inducing late-stage oscillations in both gradients and loss. In practice, this instability becomes a scalability bottleneck: it limits how far we can extend the pruning-phase token budget without sacrificing training stability. If we can safely extend the token budget in the pruning phase, we can further reduce the accuracy gap between pruned and dense models at a fixed sparsity.

This failure mode is intrinsic to dual-ascent-style enforcement rather than a simple artifact of optimizer choice or learning-rate tuning. As the multiplier grows, gate updates become constraint-dominated, which amplifies noise and drives oscillations near the end of training. This undermines the potential of

data scaling during pruning: even with more data, the dynamics become erratic, preventing the model from fully harvesting the benefits of a larger token budget.

In this work, we reformulate L_0 pruning around a simple principle: per-step L_2 -norm bounds on pruning-specific gradients. Concretely, we constrain the gradient magnitude for gate variables at each step and, from this constraint, derive a closed-form coefficient that replaces dual ascent [34, 38]. This coefficient is determined by the gradient bound rather than by accumulating the Lagrange multiplier, directly suppressing the amplification that triggers late-stage oscillations. The result is a surrogate-differentiable, globally budget-aware, and incrementally scheduled pruning objective that remains stable over long horizons.

Building on this foundation, we introduce **DIGS**—a differentiable, incremental, global, and scalable pruning framework—that enables stable structured channel pruning with much larger *prune-phase* token budgets. In our main VLM setting, we prune the LLM backbone of Qwen2.5VL-3B while keeping the vision encoder frozen, and *scale the pruning phase beyond 100B tokens*.

Our contributions are threefold:

- We identify and analyze a long-horizon instability in L_0 -regularized structured channel pruning: multiplier-driven gradient amplification that causes late-stage oscillations in gradients and loss.
- We establish **DIGS Core** as a control objective (continuous sparsity push without over-amplification), and realize it with **GNCL-Opt** (Gradient-Norm-Constrained Lagrangian Optimization), a custom-backward implementation that substantially reduces the training-time compute cost of DIGS.
- We present DIGS as a scalable framework validated on LLM backbones, including an LLM-backbone pruning instantiation inside a VLM.

2 Related Work

Pruning taxonomies. LLM / VLM pruning spans unstructured, semi-structured (e.g., $n:m$), and structured regimes (channels / heads / MLP units / layers). Criteria range from magnitude and movement signals to regularization-driven formulations and distillation-aware variants [23, 26, 38]. We focus on *training-based, structured* pruning under feasible post-training budgets [13, 16].

Post-hoc vs. training-based pruning. Post-hoc methods such as SparseGPT and Wanda prune with small calibration sets and no long-horizon training during pruning [9, 29]. These approaches typically saturate once local statistics are estimated. They are orthogonal to our setting, where task-loss gradients continue to reshape weights and gates over extended horizons.

Dual-ascent L_0 . Hard-concrete L_0 regularization provides a differentiable surrogate for pruning [20]. For structured pruning of language models, prior work extends L_0 with an augmented-Lagrangian (dual-ascent) formulation to *explicitly* enforce a global sparsity constraint. We refer to this family as Lagrangian L_0 -regularized structured pruning, denoted as **L0-Dual** [34, 37, 38]. Our method

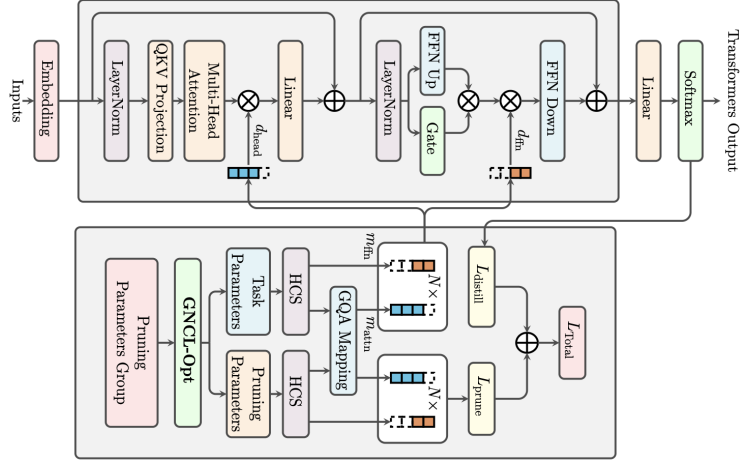


Fig. 2: DIGS realization diagram. GNCL-Opt means Gradient-Norm Controlled Lagrangian Optimization. HCS means hard-concrete sampling, which converts parameters into masks. GQA mapping maps each query head to its shared KV group and expands group-level gates into query-level masks to keep Q-KV alignment under grouped-query attention, which will be detailed in the supplement. The objective layer keeps the additive form $L_{\text{distil}} + L_{\text{prune}}$ without additional loss weighting. In contrast, GNCL-Opt applies gradient scaling in the backward control path.

retains the global budget but replaces dual ascent with per-step gradient-norm control, addressing the long-horizon instability we observe in L0-Dual.

Structured pruning at scale and prune-phase budgets. PAT performs pruning-aware training using approximately 0.256B tokens [17]. *ShearedLlama* investigates the allocation of data between pruning and continued pretraining within a fixed 5B-token overall budget, exploring prune-phase budgets ranging from 0.2B to 1.6B tokens, ultimately allocating 0.4B tokens for pruning [37]. AST utilizes 10B tokens for pruning mask learning [14]. In contrast, our experiment utilizes 100B token, while keeping total compute well below the cost of pretraining.

3 Method

We introduce **DIGS**, a *Differentiable, Incremental, Global, and Scalable* pruning framework. It reformulates constraint enforcement in L_0 -regularized structured pruning. DIGS improves long-horizon optimization stability while maintaining global sparsity budgets. Figure 2 details our method.

3.1 Notation and Problem Setup

Let θ denote model weights and w_i a structured unit (e.g., a channel or attention head). We attach to each w_i a stochastic gate $z_i \in \{0, 1\}$ and define the effective

parameter $\tilde{w}_i = z_i w_i$. Optimization over discrete gates is intractable. We use the *hard-concrete* relaxation [20] parameterized by a real-valued logit α_i :

$$\begin{aligned} u &\sim \mathcal{U}(0, 1), \quad t = \sigma\left(\frac{\log u - \log(1 - u) + \alpha_i}{\beta}\right), \\ \hat{z}_i &= \text{clamp}(t(r - l) + l, 0, 1), \quad z_i = \mathbb{I}[\hat{z}_i > 0.5]. \end{aligned} \quad (1)$$

During training we use the continuous gates $\hat{z}_i \in [0, 1]$ and at evaluation we hard-threshold. Let $\hat{z}(\alpha)$ denote the collection $(\hat{z}_i)_i$.

Weighted global sparsity. To respect that each structured unit removes a different number of parameters, we measure global sparsity as an *expected, parameter-weighted* fraction of removed parameters under the hard-concrete gates. Let $s_i > 0$ denote the parameter mass associated with unit i (i.e., the number of parameters that would be removed if unit i were pruned; for attention heads it aggregates all affected projections, and for FFN channels it includes the paired column / row), and let $S \triangleq \sum_{i=1}^N s_i$. Define:

$$R(\alpha) \triangleq \frac{1}{S} \sum_{i=1}^N s_i \pi_i(\alpha_i), \quad \pi_i(\alpha_i) \equiv \Pr_{\hat{z}_i \sim \text{HC}(\alpha_i)}[\hat{z}_i = 0], \quad (2)$$

which lies in $[0, 1]$. The closed-form expression of $\pi_i(\alpha_i)$ and its gradient w.r.t. α_i follow [20] and are provided in the supplement.

At evaluation, we report the realized sparsity with the same weights:

$$R_{\text{eval}}(z) = \frac{1}{S} \sum_{i=1}^N s_i \mathbb{I}[\hat{z}_i \leq 0.5]. \quad (3)$$

3.2 Instability of Dual Ascent

The baseline approach we evaluate against—**L0-Dual**—updates the dual variable λ proportional to the violation of $R(\alpha) = c$, which increasingly weights the pruning penalty during training [34, 37]. It enforces the sparsity constraint $R(\alpha) = c$ via a Lagrangian regularization term. The resulting objective takes the form $L(\lambda) = L_{\text{distill}} + \lambda L_{\text{prune}}$. With gate gradients denoted by $\nabla_{\hat{z}}(\cdot)$, the effective update direction becomes

$$\hat{z} \leftarrow \hat{z} - \eta(\nabla_{\hat{z}} L_{\text{distill}} + \lambda \nabla_{\hat{z}} L_{\text{prune}}), \quad (4)$$

and is then propagated to α through $\hat{z}(\alpha)$. Over long horizons, λ accumulates, making updates constraint-dominated and amplifying noise, which leads to late-stage oscillations in both gate gradients and total loss and ultimately limits how far we can scale the pruning-phase token budget.

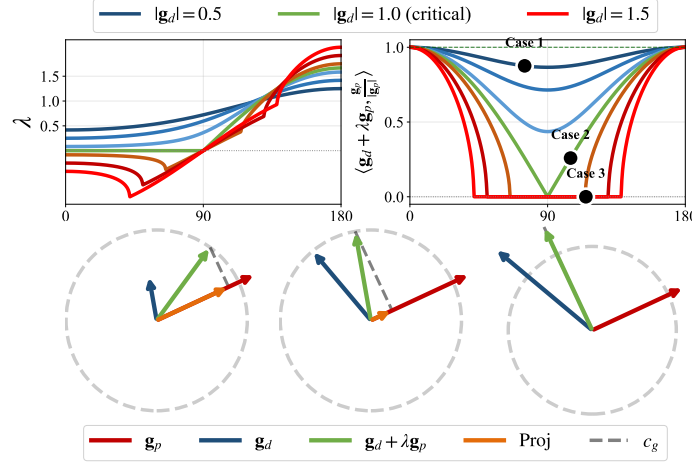


Fig. 3: Geometric cases of DIGS control. As shown in the left-hand curve, when $\angle(g_d, g_p) < 90^\circ$, larger $\|g_d\|$ leads to smaller λ ; when $\angle(g_d, g_p) > 90^\circ$, larger $\|g_d\|$ leads to larger λ . The right-hand curve visualizes the projection of the composite gradient onto the pruning direction g_p . In infeasible regions, the least-squares branch removes the component along g_p in the composite update (zero projection on g_p). Cases 1, 2, and 3 correspond to the left, middle, and right subplots below, respectively.

3.3 Gradient-Norm Controlled Regularization

DIGS replaces dual ascent with an explicit *per-step* $L2$ upper bound on the gate-gradient. Let $g_t = \nabla_{\hat{z}} L(\lambda)$ be the composite gate gradient and $g_p = \nabla_{\hat{z}} L_{\text{prune}}$. Instead of indirectly controlling magnitudes via λ , we choose $\Delta\lambda$ such that:

$$\|g_t + \Delta\lambda g_p\| \leq c_g, \quad (5)$$

where $c_g > 0$ is a user-chosen norm target. We solve the *feasible* case by saturating the upper bound (i.e., equality in Eq. (5)); define the discriminant:

$$D = (g_t^\top g_p)^2 - \|g_p\|^2 \|g_t\|^2 + \|g_p\|^2 c_g^2. \quad (6)$$

When $D \geq 0$ we take the nonnegative root:

$$\Delta\lambda_{\text{feas}} = \frac{-g_t^\top g_p + \sqrt{D}}{\|g_p\|^2 + \varepsilon_{\text{den}}}, \quad \varepsilon_{\text{den}} > 0, \quad (7)$$

and otherwise (infeasible upper bound) use the least-squares minimizer that yields the smallest attainable norm,

$$\Delta\lambda_{\text{LS}} = -\frac{g_t^\top g_p}{\|g_p\|^2 + \varepsilon_{\text{den}}}. \quad (8)$$

We then update the coefficient with a nonnegativity projection, which prevents the pruning term from pulling in the wrong direction when $|R(\alpha) - c| > 0$:

$$\lambda \leftarrow \max(\lambda + \Delta\lambda, 0), \quad \Delta\lambda = \begin{cases} \Delta\lambda_{\text{feas}}, & D \geq 0, \\ \Delta\lambda_{\text{LS}}, & D < 0, \end{cases} \quad (9)$$

Adaptive target near saturation. Sigmoid-like gate Jacobians shrink late in training and $\|g_p\|$ can become very small. To avoid inflating $\Delta\lambda$ in such cases, we scale the target as:

$$c_g^{\text{eff}} = c_g \cdot \frac{\|g_p\|}{\|g_p\| + \varepsilon_{\text{scale}}}, \quad \varepsilon_{\text{scale}} > 0, \quad (10)$$

and use c_g^{eff} in Eq. (5)–(7). Here, ε_{den} only stabilizes the denominator in the coefficient solve, whereas $\varepsilon_{\text{scale}}$ controls the adaptive target when $\|g_p\|$ becomes small. This preserves stability.

Geometry summary. Let $g_d \triangleq \nabla_z L_{\text{distill}}$ denote the gradient backpropagated from the distillation loss, and let $g_p \triangleq \nabla_z L_{\text{prune}}$ denote the gradient backpropagated from the pruning loss. We now analyze how using the DIGS-computed λ in the objective $L = L_{\text{distill}} + \lambda L_{\text{prune}}$ changes the gradient on pruning parameters. As shown in Figure 3, the geometric cases imply four operational conclusions used throughout training:

- (i) if $\angle(g_d, g_p) < 90^\circ$, the task and pruning gradients are aligned; as $\|g_d\|$ increases, the combined update can become overly aggressive, so a smaller effective λ is used to slow pruning and stabilize optimization;
- (ii) if $\angle(g_d, g_p) > 90^\circ$, the distillation gradient opposes the pruning gradient; as $\|g_d\|$ increases, pruning is increasingly suppressed by distillation pressure and sparsity progress is hindered, so a larger λ is used to counterbalance the effect;
- (iii) these two regimes realize DIGS compromise control between task fitting and sparsity progress;
- (iv) in infeasible regions, this typically corresponds to a very large $\|g_d\|$, indicating substantial remaining optimization room for distillation; therefore, the least-squares branch cancels the g_p -direction component in the composite update (zero projection on g_p), effectively pausing pruning progress until the distillation objective becomes better stabilized.

In Figure 3, Case 1 and Case 2 are both feasible (normal-control) regimes, where DIGS controls the composite gradient to satisfy the norm target (saturating the bound to $\|g_d + \lambda g_p\| = c_g$) while keeping a positive projection on the pruning direction g_p , so pruning continues in a controlled manner. Case 3 is the infeasible regime, where the least-squares branch removes the g_p -direction component (zero projection on g_p), effectively pausing pruning while distillation continues to optimize.

3.4 GNCL-Opt Realization

A direct implementation first computes gradients and then solves for λ , requiring two backward passes per optimizer step: one to obtain gate-gradient statistics for

solving $\Delta\lambda$, and another to update parameters under the refreshed coefficient. We therefore adopt a one-step realization named Gradient-Norm Controlled Lagrangian Optimization, **GNCL-Opt**, avoiding an extra backward.

In forward, GNCL-Opt duplicates the pruning parameters into two branches: one branch contributes to L_{distill} and the other to L_{prune} . The total objective is kept additive as $L = L_{\text{distill}} + L_{\text{prune}}$ (without explicitly multiplying L_{prune} by λ in forward). In backward, GNCL-Opt receives the two gate-gradient signals, computes λ using Eq. (7)–(9), and rescales only the pruning branch gradient before merging the two branches.

Equivalence and efficiency. This backward-path realization is equivalent to the DIGS formulation that uses $L_{\text{distill}} + \lambda L_{\text{prune}}$, because scaling the pruning gradient by λ in backward yields the same gate update direction as weighting loss by λ in forward. At the same time, it avoids the computation of a separate control-update pass, enabling a single-pass realization of the DIGS control rule.

4 Experiments

4.1 Training Configuration

Model and pruning scope. We conduct validation on a production-grade VLM, **Qwen2.5VL-3B** [4]. For the VLM setting, because the LLM component dominates parameters and compute, we *prune only the LLM backbone* and keep the ViT encoder and multimodal projector outside the pruning scope. Pruning vision encoders or projectors is not evaluated in this submission and is left to future work. Our setup is chosen to *isolate the effect of scaling the pruning-phase token budget*, which is the central variable of interest.

Sparsity and token budget. For Qwen2.5VL, we start from **0%** sparsity and reach **60%** in a single run. We treat the *token budget during pruning* as the primary independent variable.

Optimization setup. Hard-concrete gate parameters follow *ShearedLlama*. The learning rate for model parameters is 1e-4. To decouple LR warmup from sparsity scheduling [37], we first run **1000** dense warmup steps *without pruning* and save that checkpoint. All pruning runs start from this checkpoint, and the reported pruning steps *exclude* these 1000 warmup steps. For **L0** pruning, the gate parameter α uses a learning rate of 1e-2.

Runtime and hardware. We use sequence packing, BF16 precision, gradient checkpointing, and DeepSpeed ZeRO-2. Training runs on **8×A100** GPUs; each forward step processes **432k** global-batch tokens after sequence packing, with **4**-step gradient accumulation.

Distillation objective. Our final default distillation objective is **forward KL without CE** (Eq. (11)). We keep this setting fixed in all main-result runs and treat other losses only as ablations. The teacher is *dense* Qwen2.5VL-3B; the student’s embedding and LM head are frozen. The loss is computed over response tokens (causal-only), excluding the prompt.

$$L_{\text{distill}} = \text{KL}(p_t \parallel p_s). \quad (11)$$

4.2 DIGS-Specific Settings

Gradient-constraint sweep. We sweep **six** gradient-constraint coefficients c_g on a logarithmic grid over $[3 \times 10^{-4}, 3 \times 10^{-2}]$ to *span the pruning-phase token budget* from 0.7B to 24.5B tokens (B = billion). In DIGS, the *pruning rate emerges adaptively* from the gradient-norm constraint; consequently, the choice of c_g *indirectly* determines how much data is needed to reach the target sparsity.

Naming. We name each 0% $\rightarrow$ 60% pruning *run* by its *final* prune-phase budget B (denoted “FR”). A checkpoint from that run is written as DIGS-FRBB@30% or DIGS-FRBB@60%, e.g., DIGS-FR24.5B@30% and DIGS-FR24.5B@60% are the 30% / 60% checkpoints from the same DIGS-FR24.5B run. Unless otherwise noted, the same naming applies to L0-DUAL.

No sparsity warmup. We *do not* use a sparsity warmup for DIGS. Unlike dual methods, DIGS prevents multiplier explosion via per-step gradient-norm control, so no warmup is required for stability at larger budgets.

4.3 Baselines

We compare DIGS against **L0-Dual** (following the *ShearedLlama* configuration). For fairness, the *backbone* (*Qwen2.5VL-3B*), *distillation objective*, *optimizer*, and *data pipeline* are identical across methods, and all runs start from the same warm-start checkpoint. To match DIGS under the *same pruning-phase token budget*, we adjust only the sparsity warmup steps to match the training steps in **L0-Dual** and denote the baselines as L0-Dual-0.7B / 1.0B / 2.4B / 5.7B / 12.5B / 24.5B. Warmup acts purely as a budget-matching mechanism and does not alter other training details.

4.4 Datasets

To better align with the post-training distribution, our pruning-phase mixture combines (i) *text* pretraining and SFT datasets [1, 5, 12, 32, 40] and (ii) *multi-modal* pretraining and SFT datasets [10, 27, 28], all from public sources. Since **Qwen2.5VL-3B** is an SFT model that does not natively continue pretraining, we construct four *pseudo-pretraining* tasks for pruning by converting Pretrain data into chat template: *text-image matching*, *image-conditioned captioning*, *text-conditioned image reasoning*, and *text continuation*. Full data composition, sampling ratios, and filtering criteria are described in the supplement.

Table 1: Results at 30% sparsity. $Tok@ckpt(B)$ denotes the tokens consumed up to reaching the 30% checkpoint within a single $0\% \rightarrow 60\%$ pruning run. Δ_{Acc} represents the accuracy drop between the pruned model and the dense model (Qwen2.5VL-3B). **MMBench** results are also shown. Abbrev.: **MMSt**=MMStar, **MathV**=MathVista, **Hallu**=HallusionBench, **OCR**=OCRBench. Each two-line metric cell follows the upper/lower benchmark order shown in the header.

Method (Run / Ckpt)	Tok@ckpt (B)	MMBench Hallu	MMSt AI2D	MMMU OCR	MathV MMVet	Avg	Δ_{Acc}
Qwen2.5VL 3B (dense)	–	74.75 46.60	56.30 81.40	51.20 82.80	61.20 65.90	65.01	0.00
L0-DUAL FR5.7B@30%	3.1	68.17 49.55	48.46 70.17	40.33 78.20	53.70 57.52	58.26	-6.75
DIGS FR5.7B@30%	2.7	64.33 44.96	51.26 67.09	37.33 76.70	53.30 53.48	56.04	-8.97
L0-DUAL FR24.5B@30%	11.9	63.40 48.24	49.86 67.64	40.77 76.60	52.70 55.59	56.85	-8.16
DIGS FR24.5B@30%	10.3	68.54 44.75	52.33 70.88	41.89 78.50	57.30 55.87	58.75	-6.26

4.5 Evaluation Protocol

We evaluate with VLMEvalKit on diverse multimodal benchmarks [7]: **MMStar** [6], **MMMU-Dev-Val** [42], **AI2D-Test** [15], **MathVista-Mini-val** [22], **HallusionBench** [11], **MMVet** [39], **OCRBench** [19], and **MMBench** [18]. Unless otherwise noted, we use a fixed decoding configuration and prompt templates across all models (details in the supplement). Compared with loss-based proxies, this protocol directly measures capability boundaries at inference time.

4.6 Main Results

We report results at 30% and 60% sparsity to study how prune-phase token budget affects accuracy. Unless noted, methods are compared under matched token budgets; for 60%, each method continues from its own 30% checkpoint. **Clarification on 30% checkpoints.** The 30% rows correspond to *intermediate checkpoints* on the way to 60% sparsity *within one $0\% \rightarrow 60\%$ run*.

(Q1) *Fixed sparsity, increasing tokens.* At **30%** (Table 1), DIGS rises from **56.04** to **58.75** as budget increases from 5.7B to 24.5B, while L0-Dual decreases from **58.26** to **56.85**. At **60%** (Table 2), DIGS rises from **47.92** to **51.93**, whereas L0-Dual falls from **44.51** to **38.97** under the same budget increase.

(Q2) *Head-to-head at matched budgets.* At the smaller budget (**5.7B**), L0-DUAL performs better at **30%** (58.26 vs. 56.04), whereas DIGS performs better at **60%**

Table 2: Results at 60% sparsity. Both methods continue from their own 30% checkpoints. Δ_{Acc} represents the accuracy drop between the pruned model and the dense model (Qwen 2.5 VL 3B). **MMBench** results are also shown. Abbrev.: **MMS**t=MMStar, **MathV**=MathVista, **Hallu**=HallusionBench, **OCR**=OCRBench. Each two-line metric cell follows the upper/lower benchmark order shown in the header.

Method (Run / Ckpt)	Tok@ckpt (B)	MMBench Hallu	MMS	MMMU AI2D	MathV OCR	MMVet	Avg	Δ_{Acc}
Qwen2.5VL 3B (dense)	—	74.75 46.60	56.30 81.40	51.20 82.80	61.20 65.90	65.01	0.00	
L0-DUAL FR5.7B@60%	5.70	41.94 43.46	43.66 47.05	33.00 71.30	37.90 37.80	44.51	-20.50	
DIGS FR5.7B@60%	5.70	50.70 36.53	44.6 55.21	35.00 71.50	47.69 42.20	47.92	-17.09	
L0-DUAL FR24.5B@60%	24.50	26.96 40.66	38.41 41.25	30.77 63.40	36.70 33.62	38.97	-26.04	
DIGS FR24.5B@60%	24.50	59.74 44.43	45.80 57.41	36.88 74.90	51.40 44.91	51.93	-13.08	

(47.92 vs. 44.51). At the larger budget (**24.5B**), DIGS surpasses L0-DUAL at both sparsity levels: **58.75** vs. **56.85** (30%) and **51.93** vs. **38.97** (60%).

(Q3) *From 30% to 60% sparsity.* The same scaling behavior persists at higher sparsity: DIGS remains stable and benefits from larger prune-phase budgets, whereas L0-DUAL degrades at larger budgets.

As shown in Figure 1a, at 60% sparsity, DIGS improves monotonically as the prune-phase token budget increases and surpasses L0-Dual from approximately 2.4B tokens onward, whereas L0-Dual peaks near 5.7B tokens and then degrades. The orange dashed line at 1.6B indicates the upper end of prune-phase budgets commonly explored in *ShearedLlama* under a fixed 5B-token overall budget [37].

Takeaway. Within our setting, scaling the pruning-phase token budget *helps* when the optimization remains stable over long horizons. DIGS preserves this stability and thus benefits from more data, while L0-Dual tends to peak at mid-range budgets and then decline.

4.7 LLM Pruning Setup and Comparison to Prior Work

For fair comparison with prior pruning methods, we conduct the main LLM experiments on **Gemma** [30] and **LLaMA2** [31]. We follow the training data and evaluation protocol of PAT [17]. To align with our VLM pruning setting (where pruning starts from an SFT checkpoint), we also prune LLMs from SFT-adapted checkpoints rather than from raw pretraining checkpoints. Concretely, each base model is first adapted with LoRA SFT, and pruning is then applied.

Table 3: LLM fair-main comparison under a unified protocol. All methods are evaluated with aligned training data, sparsity targets, and evaluation settings on Gemma-2B/7B and Llama2-7B/13B. Base models are first adapted with LoRA SFT and then pruned. DIGS achieves the best overall performance on most settings and remains near-lossless in several high-sparsity (30%) cases.

Sparsity	Method	Gemma-2B	Gemma-7B	Llama2-7B	Llama2-13B
0%	Dense	53.82	71.59	58.76	66.74
25%	LLM-Pruner	40.20	50.29	39.72	39.82
	SliceGPT	39.72	52.13	41.97	46.75
	PAT	52.98	66.68	60.02	66.58
	DIGS	53.65	70.73	57.45	67.12
30%	LLM-Pruner	40.05	41.35	39.73	39.70
	SliceGPT	39.89	44.30	40.14	46.19
	PAT	45.33	64.58	57.81	65.15
	DIGS	49.81	70.12	58.27	66.15

This yields a unified FT-then-prune protocol for comparing LLM-Pruner [24], SliceGPT [2], PAT [17], and DIGS under the same SFT-adapted starting point.

LLM fair-main results. According to Table 3, **DIGS** outperforms prior pruning methods on most settings. The advantage is especially clear at higher sparsity (30%), where DIGS remains nearly lossless relative to the dense baseline on several backbones while preserving stronger accuracy than competing methods.

4.8 Ablations on Stabilization and c_g

We further ask whether the long-horizon gains can be recovered by simpler stabilizers, and how sensitive DIGS is to its gradient-constraint coefficient c_g . For these diagnostic studies, Avg₃ averages MMStar, MathVista-Mini, and OCR-Bench; we use this reduced protocol to compare trends at lower evaluation cost.

Table 4: Stabilizer ablations and c_g sensitivity at 60% target sparsity. Avg₃ averages MMStar, MathVista-Mini, and OCRBench. Left: clipping or capping partially stabilizes L0-Dual, but does not match DIGS under a comparable budget. Right: c_g mainly controls the pruning-token budget needed to reach target sparsity, with performance varying smoothly across the selected log-grid points.

(a) Stabilizer ablation			(b) c_g sensitivity					
Variant	#Tok.	Avg ₃	c_g	#Tok.	MM*	MathV	OCR	Avg ₃
L0-Dual	~12B	46.32	$1.2e-2$	1.06B	39.80	34.60	62.80	45.73
L0-Dual+clip	~12B	45.69	$4.7e-3$	2.42B	42.33	42.30	68.60	51.08
L0-Dual+ $\lambda \leq 50$	~19B	52.74	$1.9e-3$	5.71B	44.60	47.69	71.50	54.60
DIGS	~12B	56.29	$7.5e-4$	12.46B	47.07	49.10	72.70	56.29
			$3.0e-4$	24.50B	45.80	51.40	74.90	57.37

Simple stabilizers. In Table 4a, gradient clipping acts after forming the composite gate gradient and shrinks both task and pruning components together. A hard cap on λ improves L0-Dual, but it also slows sparsity growth and requires $\sim 19\text{B}$ tokens to reach the same 60% target. In contrast, DIGS solves for the pruning coefficient before merging the gradients, reaches the target at $\sim 12\text{B}$ tokens, and obtains the highest Avg₃.

c_g sensitivity. Table 4b shows that c_g behaves as a budget-control knob: smaller c_g slows pruning and increases the token budget to hit the target sparsity. Across log-grid points, accuracy improves smoothly as the budget grows and saturates, supporting the use of the fitted budget law for choosing c_g .

4.9 Stability and Training Dynamics

To explain the different data-scaling behaviors observed above, we analyze the training dynamics.

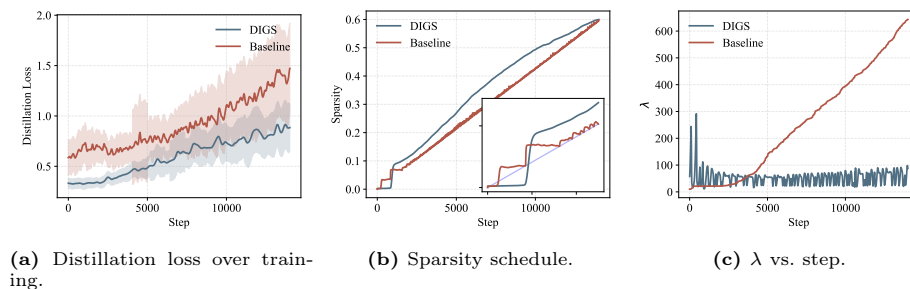


Fig. 4: Training dynamics: DIGS vs. L0-Dual. (a) DIGS attains lower distillation loss with smaller variance. (b) DIGS adapts sparsity via gradient constraints and avoids oscillations. (c) The coefficient λ remains bounded for DIGS (mostly < 100) but grows steadily for **L0-Dual**, exceeding 600 near the end.

Accuracy vs. pruning tokens. With 60% sparsity, the accuracy of **DIGS** *monotonically* improves as the pruning-phase tokens increase, whereas **L0-Dual** peaks at a mid-range budget and then *degrades* thereafter (see Figure 1a). This mirrors the trends in Table 1–Table 2.

Multiplier growth and gradient amplification. In **L0-Dual**, the Lagrange multiplier λ accumulates during training, amplifying pruning-specific gradients and inducing late-stage oscillations (Figure 4c). In contrast, **DIGS** enforces a per-step gradient-norm bound, keeping the effective gate gradient *bounded* and λ at a low level after an initial rise, which supports longer training horizons. **Figure 1b.** We plot the gate-gradient norm (left axis; scaled by 10^{-4}) together with the per-step change in λ (right axis) under a target $c_g = 3 \times 10^{-4}$. DIGS increases λ when the norm falls below the target and decreases it when above, forming a closed loop that keeps the composite gate gradient bounded and avoids the late-stage oscillations we observe with dual ascent.

Table 5: Fitted budget-law coefficients for Eq. (12). Rows use short c_g calibration sweeps at target sparsity R ; a , m , and b denote scale, exponent, and offset. These coefficients estimate the pruning-token budget and choose c_g for the scaled run.

R (%)	a	m	b
30	9.7×10^7	0.57	-7.1×10^8
45	6.4×10^7	0.68	-6.2×10^8
60	5.5×10^7	0.75	-5.9×10^8

Table 6: DIGS 100B-token pipeline attribution. Avg follows the same aggregation protocol as the main tables. Token counts are approximate; rows separate pruning, distillation, and SFT recovery stages.

Stage	Prune tok.	Extra recovery	Avg
Dense	–	–	65.01
Prune-only	114B	–	61.52
+distill	114B	20B	63.94
+distill+SFT	114B	20B+SFT	64.32

Distillation loss and variance. **DIGS** attains a lower distillation loss and a smaller variance than L0-Dual (Figure 4a); the final loss is ≈ 0.884 for DIGS vs. ≈ 1.47 for L0-Dual. Shaded bands indicate sliding-window standard deviation.

Sparsity trajectories. As shown in Figure 4b, **L0-Dual** uses a manually designed linear sparsity warm-up, with fluctuations around the target sparsity, as seen in the purple line in the lower-right subplot. Large sparse gradients occur when the model’s sparsity falls below the target. **DIGS**, on the other hand, adapts sparsity based on gradient constraints, reflecting the model’s loss landscape. It rapidly increases sparsity early on, then slows down in the later stages, which helps the model better match the teacher model as it nears convergence.

4.10 Relationship Between Pruning Parameters and the Number of Training Tokens

Empirical relation and objective. In DIGS, the gradient-constraint coefficient c_g caps the per-step gate-gradient norm and thereby controls the *pruning-phase token budget* needed to reach a target sparsity. With a few short calibration runs at a fixed sparsity R (e.g., 30% / 45% / 60%), we observe a stable power-law relation reported in *tokens* rather than steps:

$$\mathbf{B}(R) \approx a(R) c_g^{-m(R)} + b(R) \quad (12)$$

where a, m, b depend on the architecture, data mixture, and training setup. Fitting details are deferred to

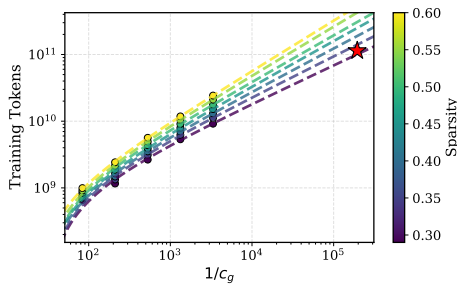


Fig. 5: Budget law. Log-log relation between pruning-phase tokens $\mathbf{B}$ and c_g at different target sparsities R .

the supplement. Here is a practical recipe:

1. **Calibrate.** From a dense checkpoint, run 4–6 short trials with log-spaced c_g values and *stop as soon as* the target sparsity R is reached; record the associated token budgets $\mathbf{B}$.
2. **Fit.** Robustly fit Eq. (12) to obtain $a(R), m(R), b(R)$; provide fitting details in the supplement.
3. **Invert by budget.** For a desired budget $\mathbf{B}^*$,

$$c_g^* = \left(\frac{a(R)}{\mathbf{B}^* - b(R)} \right)^{1/m(R)}, \quad (13)$$

Notes. Unlike **L0-Dual**, which prescribes sparsity via a warmup schedule, DIGS lets the pruning rate *emerge* from the gradient-norm constraint; given a target sparsity, Eq. (12) enables *budget-driven* selection of c_g . For visualization, we recommend a log–log plot of $\mathbf{B}$ vs. c_g at multiple R (Figure 5), and a coefficient table for a, m, b (Table 5).

4.11 Scaling training data to 100B

Motivated by the *monotonic gains* we observe when scaling the pruning-phase token budget, we scale to $\mathbf{B} \approx 100\text{B}$ at a target sparsity of **33%** (reducing a 3B model to $\sim 2\text{B}$ parameters) to prioritize accuracy. Using the fitted law in Eq. (12), we obtain $c_g^* \approx 5.1 \times 10^{-6}$ (2 significant digits). In practice, the pruning phase consumed $\sim 114\text{B}$ tokens and reached $\mathbf{Avg} = 61.52$. After physical channel removal and $\sim 20\text{B}$ tokens of post-pruning distillation, Avg improves to 63.94. A one-epoch *FineVision* SFT [35] further improves Avg to 64.32, within 0.69 of the dense baseline. Table 6 attributes this final result to each stage instead of reporting per-benchmark metrics.

5 Conclusion

DIGS achieves (1) higher stability over long horizons, (2) monotonic gains with larger pruning-phase token budgets, and (3) competitive accuracy at 60% sparsity using only a fraction of pretraining compute. Across experiments, DIGS is effective for LLM-backbone pruning, including a VLM instantiation where the vision encoder is frozen. More broadly, DIGS introduces a general stable constrained-optimization algorithm that is, in principle, applicable beyond structured channel pruning, including context pruning and MoE expert pruning; exploring these directions is an important avenue for future work.

References

1. Allal, L.B., Lozhkov, A., van Strien, D.: Cosmopedia: how to create large-scale synthetic data for pre-training. Hugging Face Blog & Dataset Card (2024), <https://huggingface.co/blog/cosmopedia>, accessed 30 June 2026

2. Ashkboos, S., Croci, M., Gennari do Nascimento, M., Hoefler, T., Hensman, J.: Slicept: Compress large language models by deleting rows and columns. In: International Conference on Learning Representations. vol. 2024, pp. 11682–11701 (2024)
3. Bai, G., Li, Y., Ling, C., Kim, K., Zhao, L.: Sparsellm: Towards global pruning of pre-trained language models. *Advances in Neural Information Processing Systems* **37**, 46203–46225 (2024)
4. Bai, S., Chen, K., Liu, X., Wang, J., Ge, W., Song, S., et al.: Qwen2.5-VL technical report (2025), arXiv preprint arXiv:2502.13923
5. Bercovich, A., Levy, I., Golan, I., Dabbah, M., El-Yaniv, R., Puny, O., Galil, I., Moshe, Z., Ronen, T., Nabwani, N., et al.: Llama-nemotron: Efficient reasoning models (2025), arXiv preprint arXiv:2505.00949
6. Chen, L., Li, J., Dong, X., Zhang, P., Zang, Y., Chen, Z., Duan, H., Wang, J., Qiao, Y., Lin, D., et al.: Are we on the right way for evaluating large vision-language models? *Advances in Neural Information Processing Systems* **37**, 27056–27087 (2024)
7. Duan, H., Yang, J., Qiao, Y., Fang, X., Chen, L., Liu, Y., Dong, X., Zang, Y., Zhang, P., Wang, J., et al.: Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In: *Proceedings of the 32nd ACM international conference on multimedia*. pp. 11198–11201 (2024)
8. Fang, G., Yin, H., Muralidharan, S., Heinrich, G., Pool, J., Kautz, J., Molchanov, P., Wang, X.: Maskllm: Learnable semi-structured sparsity for large language models. *Advances in Neural Information Processing Systems* **37**, 7736–7758 (2024)
9. Frantar, E., Alistarh, D.: Sparsegpt: Massive language models can be accurately pruned in one-shot. In: *International conference on machine learning*. pp. 10323–10337. PMLR (2023)
10. Gu, S., Zhang, J., Zhou, S., et al.: Infinity-MM: Scaling multimodal performance with large-scale and high-quality instruction data (2024), arXiv preprint arXiv:2410.18558
11. Guan, T., Liu, F., Wu, X., Xian, R., Li, Z., Liu, X., Wang, X., Chen, L., Huang, F., Yacoob, Y., et al.: Hallusionbench: an advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. pp. 14375–14385 (2024)
12. Guha, E., Marten, R., Keh, S., Raoof, N., Smyrnis, G., Bansal, H., Nezhurina, M., Mercat, J., Vu, T., Sprague, Z., Suvarna, A., Feuer, B., Chen, L., Khan, Z., Frankel, E., Grover, S., Choi, C., Muennighoff, N., Su, S., Zhao, W., Yang, J., Pimpalgaonkar, S., Sharma, K., Ji, C.C.J., Deng, Y., Pratt, S., Ramanujan, V., Saad-Falcon, J., Li, J., Dave, A., Albalak, A., Arora, K., Wulfe, B., Hegde, C., Durrett, G., Oh, S., Bansal, M., Gabriel, S., Grover, A., Chang, K.W., Shankar, V., Gokaslan, A., Merrill, M.A., Hashimoto, T., Choi, Y., Jitsev, J., Heckel, R., Sathiamoorthy, M., Dimakis, A.G., Schmidt, L.: OpenThoughts: Data recipes for reasoning models (2025), arXiv preprint arXiv:2506.04178
13. Guo, J., Chen, X., Tang, Y., Wang, Y.: SlimLLM: Accurate structured pruning for large language models. In: *International Conference on Machine Learning*. pp. 20766–20776. PMLR (2025)
14. Huang, W., Hu, Y., Jian, G., Zhu, J., Chen, J.: Pruning large language models with semi-structural adaptive sparse training. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. vol. 39, pp. 24167–24175 (2025)

15. Kembhavi, A., Salvato, M., Kolve, E., Seo, M., Hajishirzi, H., Farhadi, A.: A diagram is worth a dozen images. In: European conference on computer vision. pp. 235–251. Springer (2016)
16. Liang, Y., Wang, Z., Xu, X., Zhou, J., Lu, J.: EfficientLLaVA: Generalizable auto-pruning for large vision-language models. In: 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 9445–9454. IEEE (2025)
17. Liu, Y., Yang, H., Chen, Y., Zhang, R., Wang, M., Du, Y., Du, L.: PAT: Pruning-aware tuning for large language models. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 24686–24695 (2025)
18. Liu, Y., Duan, H., Zhang, Y., Li, B., Zhang, S., Zhao, W., Yuan, Y., Wang, J., He, C., Liu, Z., et al.: Mmbench: Is your multi-modal model an all-around player? In: European conference on computer vision. pp. 216–233. Springer (2024)
19. Liu, Y., Li, Z., Huang, M., Yang, B., Yu, W., Li, C., Yin, X.C., Liu, C.L., Jin, L., Bai, X.: Ocrbench: on the hidden mystery of ocr in large multimodal models. *Science China Information Sciences* **67**(12), 220102 (2024)
20. Louizos, C., Welling, M., Kingma, D.P.: Learning sparse neural networks through L_0 regularization (2018), arXiv preprint arXiv:1712.01312
21. Lu, H., Zhou, Y., Liu, S., Wang, Z., Mahoney, M.W., Yang, Y.: Alphapruning: Using heavy-tailed self regularization theory for improved layer-wise pruning of large language models. *Advances in Neural Information Processing Systems* **37**, 9117–9152 (2024)
22. Lu, P., Bansal, H., Xia, T., Liu, J., Li, C., Hajishirzi, H., Cheng, H., Chang, K.W., Galley, M., Gao, J.: Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In: International Conference on Learning Representations. vol. 2024, pp. 23439–23554 (2024)
23. Lu, Y., Cheng, H., Fang, Y., Wang, Z., Wei, J., Xu, D., Xuan, Q., Yang, X., Zhu, Z.: Reassessing layer pruning in LLMs: New insights and methods (2024), arXiv preprint arXiv:2411.15558
24. Ma, X., Fang, G., Wang, X.: Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems* **36**, 21702–21720 (2023)
25. Sandri, F., Cunegatti, E., Iacca, G.: 2ssp: A two-stage framework for structured pruning of llms (2025), arXiv preprint arXiv:2501.17771
26. Sanh, V., Wolf, T., Rush, A.: Movement pruning: Adaptive sparsity by fine-tuning. *Advances in neural information processing systems* **33**, 20378–20389 (2020)
27. Shan, X., Cao, Q., Han, X., Yu, H., Liang, P.P.: MINT: Multimodal instruction tuning with multimodal interaction grouping (2025), arXiv preprint arXiv:2506.02308
28. Srinivasan, K., Raman, K., Chen, J., Bendersky, M., Najork, M.: Wit: Wikipedia-based image text dataset for multimodal multilingual machine learning. In: Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval. pp. 2443–2449 (2021)
29. Sun, M., Liu, Z., Bair, A., Kolter, Z.: A simple and effective pruning approach for large language models. In: International Conference on Learning Representations. vol. 2024, pp. 4942–4964 (2024)
30. Team, G., et al.: Gemma: Open models based on gemini research and technology (2024), arXiv preprint arXiv:2403.08295
31. Touvron, H., et al.: Llama 2: Open foundation and fine-tuned chat models (2023), arXiv preprint arXiv:2307.09288
32. Wang, Y., Fu, Z., Cai, J., Tang, P., Lyu, H., Fang, Y., Zheng, Z., Zhou, J., Zeng, G., Xiao, C., Han, X., Liu, Z.: Ultra-fineweb: Efficient data filtering and verification for high-quality LLM training data (2025), arXiv preprint arXiv:2505.05427

33. Wang, Y., Ma, M., Wang, Z., Chen, J., Liping, S., Yang, Q., Xu, D., Liu, M., Qin, B.: CFSP: an efficient structured pruning framework for llms with coarse-to-fine activation information. In: Proceedings of the 31st International Conference on Computational Linguistics. pp. 9311–9328 (2025)
34. Wang, Z., Wohlwend, J., Lei, T.: Structured pruning of large language models. In: Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp). pp. 6151–6162 (2020)
35. Wiedmann, L., Zohar, O., Mahla, A., Wang, X., Li, R., Frere, T., von Werra, L., Gosthipaty, A.R., Marafioti, A.: FineVision: Open data is all you need (2025), arXiv preprint arXiv:2510.17269
36. Wu, K., Peng, H., Zhou, Z., Xiao, B., Liu, M., Yuan, L., Xuan, H., Valenzuela, M., Chen, X.S., Wang, X., et al.: Tinyclip: Clip distillation via affinity mimicking and weight inheritance. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 21970–21980 (2023)
37. Xia, M., Gao, T., Zeng, Z., Chen, D.: Sheared llama: Accelerating language model pre-training via structured pruning. In: International Conference on Learning Representations. vol. 2024, pp. 5385–5409 (2024)
38. Xia, M., Zhong, Z., Chen, D.: Structured pruning learns compact and accurate models. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1513–1528 (2022)
39. Yu, W., Yang, Z., Li, L., Wang, J., Lin, K., Liu, Z., Wang, X., Wang, L.: Mm-vet: Evaluating large multimodal models for integrated capabilities. In: International Conference on Machine Learning. pp. 57730–57754. PMLR (2024)
40. Yu, Y., Dai, Z., Wang, Z., Wang, W., Chen, R., Pei, J.: OpenCSG chinese corpus: A series of high-quality chinese datasets for LLM training (2025), arXiv preprint arXiv:2501.08197
41. Yuan, Z., Sun, W., Liu, Y., Zhou, H., Zhou, R., Li, Y., Zhang, Z., Song, W., Huang, Y., Jia, H., Murugesan, K., Wang, Y., He, L., Gao, J., Sun, L., Ye, Y.: EfficientLLM: Efficiency in large language models (2025). <https://doi.org/10.48550/arXiv.2505.13840>, arXiv preprint arXiv:2505.13840
42. Yue, X., Ni, Y., Zhang, K., Zheng, T., Liu, R., Zhang, G., Stevens, S., Jiang, D., Ren, W., Sun, Y., et al.: Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9556–9567 (2024)