DYNAMIC DEPTH ROUTING OF LARGE LANGUAGE MODELS WITH MIXTURE OF SEGMENT OPERATORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) process every input through a deep stack of layers, which makes inference slow and costly, yet many inputs do not need the full stack. Existing methods reduce this depth either by permanently pruning layers or by learning to skip layers per input, but most decide about one layer at a time, even though the quality of the reduced model depends on the whole schedule of which layers are skipped, run or reused. We propose MoSO (Mixture of Segment Operators), which casts depth reduction as learning a *routing schedule* over a frozen pretrained model. Its actions are *segment operators* that skip, run or repeat a short group of consecutive layers; this family contains static pruning, layer skipping and looped computation as special cases, and, under explicit linear-block conditions, we show that a segment repeat can realize maps that no layer-by-layer action sequence can. A lightweight router scores all candidate segments jointly, so each decision is made in the context of the schedule being built, and the schedule is learned end to end without any search over schedules. Across five LLMs from three model families, MoSO attains the highest average accuracy at each reported operating point, with its lead over the strongest baseline growing at the tighter one, and controlled experiments show that the learned schedule contributes to it.

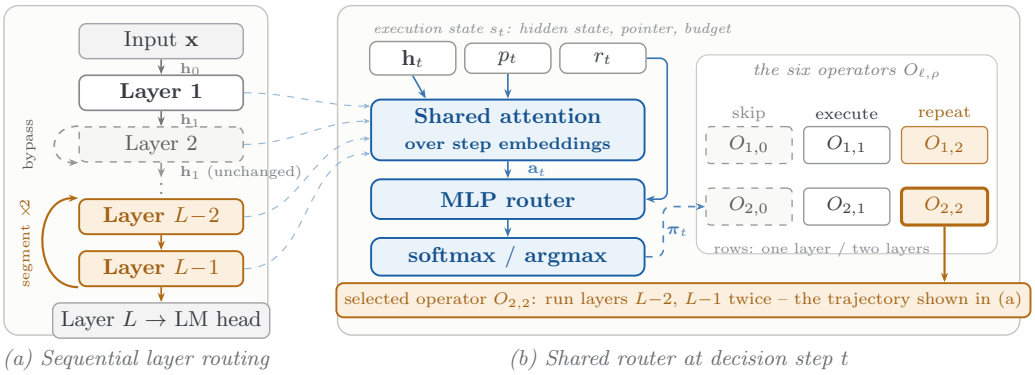
1 INTRODUCTION

Large language models (LLMs) process every input by passing it through a deep stack of transformer layers, one after another – 28 to 32 layers for the models studied here, and many more at larger scale. Because the layers run in sequence, the depth of this stack largely determines how long an answer takes and how much it costs to produce (Kaplan et al., 2020; Hoffmann et al., 2022). Making inference cheaper is therefore a practical necessity wherever LLMs are deployed under a latency or cost budget: interactive assistants that must answer in real time, high-volume serving where compute is the dominant expense, and on-device or edge deployment where hardware is limited. At the same time, much of this depth is redundant. Adjacent layers often change the model's internal representation only slightly, and not every input needs the full stack: a simple factual question can be answered with far less computation than a multi-step reasoning problem (Schuster et al., 2022; Fan et al., 2025a; Gromov et al., 2025). This raises a natural question, and it is the problem this paper studies: *given a pretrained LLM and a compute budget, which layers should be run for a given input, and which can be safely skipped?*

Two families of methods answer this question, and both have been successful in practice. *Static depth pruning* scores each layer's importance once, on a small calibration set, and permanently removes the least important ones: ShortGPT (Men et al., 2025), SLEB (Song et al., 2024), LaCo (Yang et al., 2024) and BlockPruner (Zhong et al., 2025) remove a modest fraction of the layers of modern LLMs with only moderate loss in quality, need no training, and are simple to deploy at scale. *Dynamic depth routing* instead lets the model decide per input: a small learned router, trained with the model or attached to it, chooses at each layer whether to run it or skip it (Raposo et al., 2024; He et al., 2025; Luo et al., 2025), and Dr.LLM (Heakl et al., 2026) further allows a layer to be run twice. Together these lines have shown that large parts of an LLM's depth can be removed at inference time, and that the choice can be learned from data.

Yet an important challenge remains open. Most existing methods use a layer-wise primitive: each routing action concerns *one layer at a time*, which limits the schedules they can represent. The

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Sequential layer routing *(b) Shared router at decision step t*

Figure 1: **Overview of MoSO. (a)** At each step the router decides whether to skip, run, or run twice the next one or two layers of a frozen L-layer LLM with input $\mathbf{x}$ ($\mathbf{h}_i$: hidden state after layer i, $\mathbf{h}_0$ the embedding); here layer 2 is skipped and layers $L-2$, $L-1$ are run twice as one segment. **(b)** At step t, from the current hidden state $\mathbf{h}_t$, the pointer p_t to the next layer and the remaining budget r_t, attention over step embeddings (output $\mathbf{a}_t$) and an MLP give a distribution $\boldsymbol{\pi}_t$ over six segment operators $O_{\ell,\rho}$ (ℓ layers run ρ times), and the best one is applied.

quality of a reduced model, however, depends on the *whole schedule* of which layers are skipped, run or reused, not on each choice separately: removing two neighbouring layers can be harmless or destructive depending on what else is removed, and the layers a good schedule removes need not be the ones a per-layer importance score would pick. Layer-by-layer decisions also leave out structured actions that a schedule can use, such as treating two adjacent layers as a single unit to skip or to run again. The consequence is visible exactly where depth reduction is most valuable: as the compute budget tightens, the accuracy of existing methods falls off sharply (Table 1). Addressing this is difficult for two reasons. First, the space of schedules grows exponentially with depth, so it cannot be searched per input at inference time, and it is too large to supervise exhaustively during training. Second, the decisions are discrete, so the schedule must be learned through a relaxation that still yields a good hard decision at inference, while the model's own weights stay frozen.

We address this challenge with MoSO (*Mixture of Segment Operators*), which treats depth reduction as *learning a routing schedule* over the pretrained model. Its basic unit is a short contiguous group of layers – a *segment* – rather than a single layer, and its actions are *segment operators* that skip a segment, run it, or run it twice (Figure 1). This one family contains static pruning, layer skipping, skip-or-repeat routing and looped computation as special cases, and we show that, under explicit conditions, its segment-level actions are strictly more expressive than per-layer skip, execute and repeat. A lightweight router, trained on top of the frozen model, scores all candidate segments *jointly*, so that every decision is made in the context of the schedule being built, and the whole schedule is learned end to end from the model's own training signal – language modelling and distillation from the unreduced model – without any search over schedules. The compute budget is exposed as a single, intuitive control, the number of layers to skip. We evaluate MoSO on five LLMs from three model families (LLaMA, Qwen and Mistral), against static pruners and dynamic routers, with every method compared at its *measured* computational cost. Our contributions are:

- **A unified formulation of depth reduction as a routing schedule over segment operators.** It places static pruning, layer skipping, skip-or-repeat routing and looped computation in one action space as special cases, and we prove that its segment-level operators make it strictly more expressive than per-layer skip, execute and repeat under explicit linear-block conditions (§3.2).
- **MoSO, a method that learns this schedule on a frozen LLM.** A joint, cross-layer router plans which segments to skip, run or repeat for each input, and is trained end to end from language modelling without search over schedules (§3.3, §3.4).
- **Consistent gains across models and budgets.** At the reported operating points, MoSO attains the highest Avg_6 in all ten settings, with margins that grow to $+5$ to $+12$ points at the tight budget, and controlled experiments show that the learned schedule contributes to this gain (§4).

2 Related Work

Deep residual networks, and LLMs in particular, are known to carry substantial depth redundancy (Veit et al., 2016; Huang et al., 2016; Gromov et al., 2025; Csordás et al., 2025; Sun et al., 2025). **Static depth pruning** exploits it by permanently removing layers chosen on calibration data, using hidden-state similarity (ShortGPT (Men et al., 2025)), greedy perplexity search (SLEB (Song et al., 2024)), layer merging (LaCo (Yang et al., 2024)) or sub-block search (Zhong et al., 2025; Zhang et al., 2024), and is complementary to width pruning (Ashkboos et al., 2024; Ma et al., 2023). **Dynamic depth routing** instead adapts computation to the input, through confidence-based early exits (Schuster et al., 2022; Varshney et al., 2024; Fan et al., 2025a), layer dropout with self-speculative decoding (Elhoushi et al., 2024), token-level bypassing trained with the model (Raposo et al., 2024), routers retrofitted to a frozen model that skip layers (He et al., 2025; Luo et al., 2025) or also repeat them (Heakl et al., 2026), or per-input search over layer compositions (Li et al., 2025), which PoLar (Li et al., 2026b) distils into a segment-level predictor. **Adaptive and looped computation** treats depth as a resource, applying a shared block a variable number of times (Graves, 2016; Dehghani et al., 2019; Bae et al., 2025); recent looped transformers scale this idea to LLMs and study its design (Geiping et al., 2025; Saunshi et al., 2025; Fan et al., 2026; Li et al., 2026a; Gao et al., 2026; Zhang et al., 2026; Huang et al., 2026), while elastic models (Devvrit et al., 2024; Cai et al., 2024) and mixtures of experts (Shazeer et al., 2017; Fedus et al., 2022) adapt along the width axis instead. MoSO brings these lines into one action space – static pruning and layer skipping are its skip/execute slice, skip-or-repeat routing its single-layer slice, and looped computation its repeat operators – and departs from them in two ways. It makes a contiguous segment of the backbone's own layers the unit of routing, which is strictly more expressive than per-layer skip, execute and repeat under explicit conditions (§3.2) and, unlike looped models, reuses distinct pretrained layers chosen per input rather than a shared block on a fixed schedule; and, unlike PoLar, it learns the whole schedule jointly and end to end on a frozen model, without search-based supervision or pretraining with routing in place. A detailed comparison, with a positioning table of prior depth-pruning and depth-amplification methods, is given in Appendix A.

3 Mixture of Segment Operators

Consider a pretrained autoregressive transformer with L decoder layers whose parameters stay frozen. Writing f_i for the i-th transformer block, the model computes

$$\mathbf{h}_i \;=\; f_i(\mathbf{h}_{i-1}) \;=\; \mathbf{h}_{i-1} + f_i^{\mathrm{res}}(\mathbf{h}_{i-1}), \qquad i = 1, \dots, L, \tag{1}$$

with f_i^{res} the residual branch of block i, $\mathbf{h}_0$ the token embedding and $\mathbf{h}_L$ the final hidden state read by the frozen language-model head. Most existing depth-reduction methods choose one action ρ_i per layer and compose $\mathbf{h}_L = f_L^{\rho_L} \circ \cdots \circ f_1^{\rho_1}(\mathbf{h}_0)$ (f^{ρ} the ρ-fold composition, $f^0 = \mathrm{id}$), with $\rho_i \in \{0, 1\}$ for skip or execute (ShortGPT (Men et al., 2025), LayerSkip (Elhoushi et al., 2024), Router-Tuning (He et al., 2025)) or $\rho_i \in \{0, 1, 2\}$ when a layer may also be repeated (Heakl et al., 2026). Each decision concerns one layer and is made separately from the others. MoSO replaces these per-layer actions with a *routing schedule* over segments of layers, built from three ingredients: a parameterized family of *segment operators* that unifies the action sets of prior methods (§3.1), whose theoretical properties justify it (§3.2); a *cross-layer router* that chooses among these operators jointly (§3.3); and an end-to-end training procedure on the frozen model (§3.4).

3.1 Parameterized Segment Operator Family

Per-layer action sets cannot express behaviours such as treating two adjacent layers as a single unit, or reusing such a pair as a whole. We therefore formulate depth control as selection from a family of operators indexed by two integers – how many layers to take next, and how many times to run them. The router walks the stack while carrying an execution state $s_t = (\mathbf{h}_t, p_t, r_t)$ at decision step $t \in \{1, \dots, N\}$ ($N \le L-2$): the current hidden state $\mathbf{h}_t$ (indexed by step, unlike $\mathbf{h}_i$ in Eq. (1)), a pointer p_t to the next unconsumed layer, and the remaining budget r_t in layer evaluations, with r_1 set by the compute budget. With $F_{p,\ell} := f_{p+\ell-1} \circ \cdots \circ f_p$ the *segment* of ℓ consecutive layers starting at layer p and $F_{p,\ell}^{\rho}$ its ρ-fold self-composition ($F_{p,\ell}^{0} := \mathrm{id}$), the *parameterized operator family* consists of the state maps

$$O_{\ell,\rho}(\mathbf{h},\, p,\, r) \;=\; \big(F_{p,\ell}^{\rho}(\mathbf{h}),\;\; p + \ell,\;\; r - \ell\rho\big), \tag{2}$$

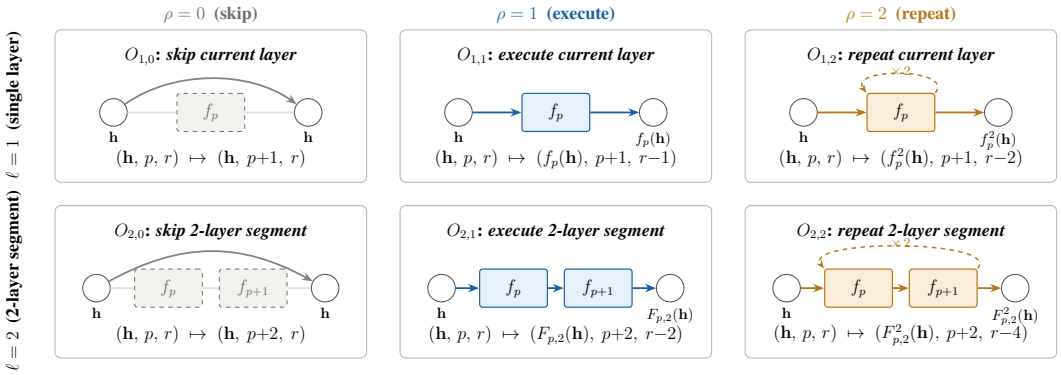


Figure 2: **Parameterized segment operator family** $\mathcal{O} = \{O_{\ell,\rho} : \ell \in \{1,2\}, \rho \in \{0,1,2\}\}$**.** Rows give how many layers an operator spans, columns how many times it runs them (zero = skip, one = execute, two = repeat); gray arcs mark the residual pathway through a skipped segment. The ℓ=1 row is the per-layer skip/execute/repeat action set; the ℓ=2 row adds segment-level operators, of which the segment repeat $O_{2,2}$ is a strict expressive extension under the conditions of Proposition 3. Every operator has the transition $O_{\ell,\rho}(\mathbf{h}, p, r)$=$(F^{\rho}_{p,\ell}(\mathbf{h}),\ p{+}\ell,\ r{-}\ell\rho)$ and cost $\ell\rho$.

each running the ℓ layers at the pointer ρ times, advancing the pointer past them and charging its cost $c(O_{\ell,\rho}) = \ell\rho$ layer evaluations to the budget.

We use $\ell \in \{1,2\}$ and $\rho \in \{0,1,2\}$, which gives the six operators $\mathcal{O}$ of Figure 2: skip $(O_{1,0}, O_{2,0})$, execute $(O_{1,1}, O_{2,1})$ and repeat $(O_{1,2}, O_{2,2})$ a single layer or a two-layer segment. This compact family already contains the action spaces of prior methods as slices: static pruning is the input-independent $\{O_{1,0}, O_{1,1}\}$ slice, learned layer skipping its input-dependent version, skip-or-repeat routing the single-layer slice $\{O_{1,0}, O_{1,1}, O_{1,2}\}$, and looped computation the repeat operators (Table 2, Appendix A; larger (ℓ, ρ) in Appendix B.2). Because all operators share one state signature, a single router can choose among all of them, so the family turns the choice between these previously separate methods into a decision that is learned per input. One convention completes the formulation: the first and last layers, which map into and out of the model's embedding space, are always executed, so the schedule covers layers $2, \ldots, L{-}1$ and the operators valid at pointer p, written $\mathcal{O}_{\text{valid}}(p)$, satisfy $p + \ell - 1 \leq L - 1$. For a schedule $(\ell_t, \rho_t)_t$, the number of skipped layers $K = \sum_t \ell_t \mathbf{1}[\rho_t{=}0]$ is the compute control, and the realized compute on an input $\mathbf{x}$, $C(\mathbf{x}) = 2 + \sum_t \ell_t \rho_t$, counts the layers actually evaluated; all compute figures we report are measured (§4.1).

3.2 Theoretical Properties

A larger action space is worth learning only if it is well behaved and larger. We establish both: the error of any schedule is controlled by quantities a router can target (Theorem 1), enlarging the action space can only improve the best achievable compute–quality trade-off (Proposition 2), and the segment operators make it strictly larger than the layer-wise space under the conditions of Proposition 3. Proofs are in Appendix C.

We assume every block f_i is Lipschitz with constant at most $1 + \eta_i$ ($\eta_i \geq 0$) on a set of hidden states that all blocks map into itself (Assumption 1, Appendix C.1), and write $\mathbf{y}_{t-1}$ for the hidden state entering step t under a hard schedule and $\Lambda_t := \mathrm{Lip}(F_{p_t,\ell_t})$ for the Lipschitz constant of the segment used at that step.

Theorem 1 (Operator-trajectory approximation)**.** *Under Assumption 1, let $(\ell_t, \rho_t)_{t=1}^N$ be an operator trajectory partitioning layers $2, \ldots, L{-}1$, with layers 1 and L always executed and skips emitting the identity. With $m_t(0){=}1$, $m_t(1){=}0$, $m_t(2){=}\Lambda_t$, the routed and full final states satisfy*

$$\left\|\mathbf{h}_L^{\text{routed}} - \mathbf{h}_L^{\text{full}}\right\| \ \leq\ \sum_{t=1}^{N} A_t\, m_t(\rho_t)\, \left\|F_{p_t,\ell_t}(\mathbf{y}_{t-1}) - \mathbf{y}_{t-1}\right\|, \qquad A_t \ =\ (1{+}\eta_L) \prod_{t'=t+1}^{N} \Lambda_{t'}. \quad (3)$$

Theorem 1 decomposes the deviation of a routed model from the original into per-step terms: the change the altered segment would have made to the hidden state, weighted by how much the remaining layers can amplify it. It has two implications for the design of MoSO. A good schedule should skip segments whose residual update is small and repeat segments whose update and Lipschitz constant are small, which is information about the current hidden state and about the position in the stack – exactly what the router of §3.3 observes. The amplification factor A_t accumulates over all later segments, so the cost of the same perturbation depends on where in the stack it occurs, and the router must know its position. The bound is a design principle that learned schedules follow (§4.4).

The approximation result holds for any schedule; the next question is whether routing over the larger family can find better ones. For a routing rule ϕ in a class $\mathcal{F}$, let $R(\phi)$ be its expected task loss and $\bar{C}(\phi) = \mathbb{E}_{\mathbf{x}}[C(\mathbf{x})]$ its expected compute under ϕ, and let $R^\star_{\mathcal{F}}(B) = \inf\{R(\phi) : \phi \in \mathcal{F},\ \bar{C}(\phi) \le B\}$ be the best loss attainable at budget B. Input-independent masks, input-dependent skipping, the single-layer slice and the full family form a nested chain $\mathcal{F}_{\text{static}} \subseteq \mathcal{F}_{\text{bin}} \subseteq \mathcal{F}_{\ell=1} \subseteq \mathcal{F}_{\mathcal{O}}$.

Proposition 2 (Pareto dominance)**.** *Under mild regularity conditions (Appendix C.2), for every budget B, $R^\star_{\mathcal{O}}(B) \le R^\star_{\ell=1}(B) \le R^\star_{\text{bin}}(B) \le R^\star_{\text{static}}(B)$, writing $R^\star_{\mathcal{O}}$ for $R^\star_{\mathcal{F}_{\mathcal{O}}}$ and so on. For the penalized optimum $J^\star_{\mathcal{F}}(\lambda) = \inf_{\phi\in\mathcal{F}}[R(\phi) + \lambda\bar{C}(\phi)]$, $\lambda \ge 0$, the same chain holds, and $J^\star_{\mathcal{O}}(\lambda) < J^\star_{\ell=1}(\lambda)$ whenever, with positive probability, no ℓ=1 trajectory attains the per-input optimum over $\mathcal{O}$.*

Proposition 2 says that optimal routing over the full family can never do worse than any of the methods it contains, at any budget, and does strictly better as soon as some inputs are best served only by a segment repeat; the analogous condition at the first inclusion – the best skip pattern varies across inputs – is what makes input-dependent routing preferable to static pruning. It therefore justifies learning a single router over the whole family instead of committing in advance to one of its slices. Whether segment-level actions are ever necessary is a question of expressivity, settled next.

Proposition 3 (Strict expressive gap)**.** *Let $f_p(\mathbf{h}) = \mathbf{h}+\mathbf{P}\mathbf{h}$ and $f_{p+1}(\mathbf{h}) = \mathbf{h}+\mathbf{Q}\mathbf{h}$ be linear residual blocks with identity skips, $I+\mathbf{P}$ and $I+\mathbf{Q}$ invertible, and $|\det(I+\mathbf{P})|$, $|\det(I+\mathbf{Q})|$ both strictly above or both strictly below 1. If the commutator $[\mathbf{P},\mathbf{Q}] \ne 0$, the segment repeat, whose hidden-state map is $(f_{p+1}\circ f_p)^2$, is not realized by any ℓ=1 composition $O_{1,\rho_{p+1}} \circ O_{1,\rho_p}$, $\rho_p, \rho_{p+1} \in \{0,1,2\}$.*

Repeating a two-layer segment interleaves its layers ($f_{p+1}f_pf_{p+1}f_p$), whereas per-layer repetition can only produce block powers ($f_{p+1}^{\rho_{p+1}} f_p^{\rho_p}$); when the layers do not commute and the determinant condition holds, the interleaved computation is out of reach of any ℓ=1 router. Distinct pretrained layers generically do not commute, so under these conditions $\mathcal{O}$ realizes segment maps that the ℓ=1 slice cannot, and the strict gain of Proposition 2 is attainable in principle; a local version for nonlinear blocks, separating $O_{2,2}$ from the per-layer repeat $O_{1,2}\circ O_{1,2}$, is Corollary 5. This operator-class separation does not imply that a learned router needs or often selects segment repeats on a given model; their use is examined in §4.4. Taken together, the three results say what a router over segment operators should do and why it is worth learning: choose among all operators rather than a fixed slice, condition on the current state and position, and skip where the residual update is small.

3.3 Cross-Layer Router

The router turns these requirements into a schedule. Its defining property is that decisions are made *jointly*: rather than gating each layer in isolation, it represents every decision step and lets each attend to the steps before it, so that what is skipped in one place is decided in the context of the schedule built so far. At step t the router forms an embedding $\mathbf{e}_t$ from a pooled summary of the hidden state, a learned embedding $\mathbf{p}_{p_t}$ of the position in the stack – the quantity Theorem 1 shows to govern amplification – and a prior on the importance of the layer at that position (Appendix B.3). Self-attention over the step embeddings then gives each step a context vector

$$\alpha_{t,j} = \mathrm{softmax}_{j\le t}\big(\mathbf{q}_t^\top \mathbf{k}_j/\sqrt{d_k}\big), \qquad \mathbf{a}_t = \textstyle\sum_{j\le t}\alpha_{t,j}\,\mathbf{v}_j, \tag{4}$$

with learned projections $\mathbf{q}_t{=}\mathbf{W}_Q\mathbf{e}_t$, $\mathbf{k}_j{=}\mathbf{W}_K\mathbf{e}_j$ and $\mathbf{v}_j{=}\mathbf{W}_V\mathbf{e}_j$ of width d_k (two multi-head layers in practice; Table 4, Appendix B.3), through which every decision sees the schedule built so far.

A small MLP maps the context, the position embedding and the remaining budget to a single distribution over all six operators,

$$\mathbf{u}_t = \mathrm{MLP}\big([\,\mathbf{a}_t\,;\,\mathbf{p}_{p_t}\,;\,r_t\,]\big), \qquad \boldsymbol{\pi}_t = \mathrm{softmax}\big((\mathbf{W}^{(\mathcal{O})}\mathbf{u}_t + \mathbf{b}^{(\mathcal{O})})/\tau\big), \tag{5}$$

with output projection $\mathbf{W}^{(\mathcal{O})}, \mathbf{b}^{(\mathcal{O})}$, temperature τ, and operators outside $\mathcal{O}_{\text{valid}}(p_t)$ masked. We use one joint head over (ℓ, ρ) rather than separate heads for segment length and repetition count, because separate heads can only represent product distributions and would lose, during training, any dependence between how many layers to take and how often to run them (Proposition 4, Appendix B.1). Since the router only reads pooled states and small embeddings, its cost is negligible next to that of the layers it controls.

3.4 Optimization

The backbone stays frozen throughout; the trained components and their schedule are detailed in Appendix B.3.1. With loss weights $\lambda_{\text{compute}}, \lambda_{\text{focal}} \geq 0$, training minimizes

$$\mathcal{L}_{\text{MoSO}} \;=\; \mathcal{L}_{\text{LM}} \;+\; \lambda_{\text{compute}}\,\mathcal{L}_{\text{compute}} \;+\; \lambda_{\text{focal}}\,\mathcal{L}_{\text{focal}} \;+\; \mathcal{L}_{\text{aux}}, \tag{6}$$

where $\mathcal{L}_{\text{LM}}$ is the language-modelling loss of the routed model and $\mathcal{L}_{\text{compute}} = \mathbb{E}_{\mathbf{x}}\big[\sum_t \sum_{(\ell,\rho)} \boldsymbol{\pi}_t(\ell,\rho)\,\ell\rho\big]$ is a differentiable surrogate for the realized compute, so that the first two terms trace the penalized trade-off of Proposition 2. The third term addresses the main risk of a widened action space: a softmax over many operators tends to collapse onto one of them early in training, before the others have been explored. With $\bar{\pi}_{\ell,\rho}$ the mean of $\boldsymbol{\pi}_t(\ell,\rho)$ over the steps and inputs of a batch, the *focal-balance* term $\mathcal{L}_{\text{focal}} = -\frac{1}{|\mathcal{O}|}\sum_{(\ell,\rho)}(1-\bar{\pi}_{\ell,\rho})^{\gamma}\log\bar{\pi}_{\ell,\rho}$ (γ=2, after the focal loss (Lin et al., 2017)) damps overused operators and pushes underused ones, and its unique minimizer is the uniform marginal (Appendix C.6), so every operator is explored before the router commits. $\mathcal{L}_{\text{aux}}$ has a distillation term from the unreduced model and an entropy bonus on $\boldsymbol{\pi}_t$ that encourages early exploration (Appendix B.1.3).

The operator choice is discrete, so training relaxes it and updates the hidden state by the mixture

$$\mathbf{h}_{t+1} \;=\; \sum_{(\ell,\rho)\in\mathcal{O}_{\text{valid}}(p_t)} \boldsymbol{\pi}_t(\ell,\rho)\; F^{\rho}_{p_t,\ell}(\mathbf{h}_t), \tag{7}$$

with a straight-through update of the pointer (Bengio et al., 2013), an expected-cost update of the budget, and a temperature annealed from 1.0 to 0.2. At inference the router applies the highest-scoring operator at each step, so exactly one path is executed, and the per-step gap between the soft and the hard update vanishes as the distribution sharpens (Lemma 6, Appendix C.5). The compute budget is set by the number K of layers to skip.

4 Experiments

We ask three questions. Does MoSO give a better compute–accuracy trade-off than existing depth-reduction methods (§4.2)? Does its advantage come from the learned routing schedule (§4.3)? And how does the learned schedule allocate computation across inputs and operators (§4.4)?

4.1 Setup

Models and training. We evaluate on five frozen pretrained LLMs from three families – LLaMA-2-7B (Touvron et al., 2023), LLaMA-3.2-3B, LLaMA-3-8B-Instruct (Grattafiori et al., 2024), Qwen2.5-7B (Qwen Team, 2025) and Mistral-7B (Jiang et al., 2023) – with 28 to 32 layers each. MoSO is trained with a single recipe for every model and budget: a small corpus of reasoning and instruction data, the language-modelling and distillation objective of §3.4, and no search over schedules. Data, hyperparameters and hardware are described in Appendices B.3.1 and D.1.

Evaluation. Our performance metric is zero-shot accuracy on six standard benchmarks – ARC-Challenge (ARC-C) (Clark et al., 2018), HellaSwag (HSwag) (Zellers et al., 2019), MMLU (Hendrycks et al., 2021), PIQA (Bisk et al., 2020), WinoGrande (WG) (Sakaguchi et al., 2021) and BoolQ (Clark et al., 2019) – and their mean, Avg_6. Methods are compared at their *measured* compute: the FLOPs of the reduced model relative to the full (dense) one, which for methods that skip whole layers is the fraction of layer evaluations, $C(\mathbf{x})/L$. Each model is evaluated at two budgets, a loose one that removes about an eighth to a sixth of the layers and a tight one that removes a quarter of them. MoSO results average three seeds (standard deviations in Appendix E.1).

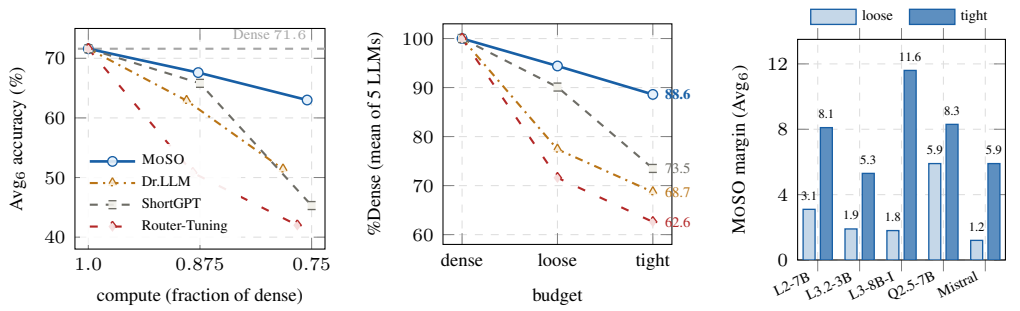
Table 1: **Accuracy at measured compute** (%, zero-shot). Budgets $(L-K)/L$ are given under each backbone; cells are loose / tight. Best, second and third non-dense scores are in **bold**, underlined and *italics*. %Dense: Avg_6 relative to dense. MoSO: mean of three seeds. Input-dependent baselines may realize different compute at the same nominal budget (FLOP ratios in Appendices E.1 and E.3). Dr.LLM uses target-budget sampled supervision; its default settings are compared in Appendix E.3.

		Method (budget:)	MMLU loose / tight	HSwag loose / tight	ARC-C loose / tight	PIQA loose / tight	WG loose / tight	BoolQ loose / tight	Avg_6 loose / tight	%Dense loose / tight
LLaMA-2-7B 0.844 / 0.750		Dense	40.8	75.9	46.3	78.8	69.2	77.7	64.8	100.0
	Static	ShortGPT	34.1 / 23.5	68.4 / *60.1*	39.1 / 35.3	73.2 / 68.1	67.1 / 65.6	**74.9** / 62.2	59.5 / 52.5	91.8 / 81.0
	Static	SLEB	26.5 / 25.1	65.5 / 60.2	37.5 / *33.3*	**75.4 / 72.8**	60.4 / *57.9*	63.2 / *62.1*	54.7 / *51.9*	84.5 / *80.1*
	Static	LaCo	26.0 / *24.5*	62.3 / 54.1	34.5 / 30.3	73.9 / *69.5*	57.1 / 53.8	59.8 / 50.4	52.3 / 47.1	80.7 / 72.7
	Static	BlockPruner	25.5 / 23.0	61.5 / 48.5	33.7 / 27.2	72.3 / 68.0	61.6 / 54.1	66.6 / *62.1*	53.5 / 47.2	82.6 / 72.8
	Dyn.	Router-Tuning	*31.0* / 25.1	*67.1* / 51.0	*37.7* / 28.5	*74.2* / 61.0	*64.5* / 57.0	*67.2* / 57.7	*56.9* / 46.7	*87.9* / 72.1
	Dyn.	SkipGPT	23.8 / 23.8	31.9 / 30.0	22.5 / 25.7	56.1 / 52.7	50.6 / 50.9	54.6 / 39.2	39.9 / 37.1	61.6 / 57.2
	Dyn.	Dr.LLM	28.6 / 25.1	56.8 / 34.0	31.5 / 27.4	64.4 / 58.9	56.0 / 55.6	62.1 / 57.9	49.9 / 43.1	77.0 / 66.6
		MoSO (ours)	**45.4 / 43.8**	**70.9 / 65.9**	**41.4 / 39.6**	75.3 / 71.3	**68.2 / 67.2**	74.7 / **75.5**	**62.6 / 60.6**	**96.7 / 93.5**
LLaMA-3.2-3B 0.857 / 0.750		Dense	54.1	73.6	45.9	77.4	69.9	72.9	65.6	100.0
	Static	ShortGPT	49.8 / 34.2	63.5 / *49.0*	*40.8* / *34.0*	69.8 / 64.1	65.8 / 62.2	61.7 / 45.0	*58.6* / *48.1*	*89.2* / *73.3*
	Static	SLEB	34.4 / 23.4	61.3 / 46.7	33.8 / 28.9	*72.3* / *68.2*	60.2 / 53.9	53.5 / 39.1	52.6 / 43.4	80.1 / 66.1
	Static	LaCo	23.5 / 23.2	51.8 / 42.8	30.2 / 24.6	71.4 / 65.3	54.0 / 52.7	58.7 / *54.5*	48.3 / 43.8	73.5 / 66.8
	Static	BlockPruner	26.2 / 24.8	54.8 / 39.9	28.5 / 26.2	70.7 / 61.1	57.1 / 52.3	60.8 / 53.8	49.7 / 43.0	75.7 / 65.6
	Dyn.	Router-Tuning	28.8 / 24.6	56.1 / 39.5	29.0 / 23.1	68.6 / 61.0	54.1 / 52.0	54.2 / 50.9	48.5 / 41.9	73.9 / 63.8
	Dyn.	SkipGPT	*46.0* / *33.7*	64.9 / 55.3	42.1 / 35.8	**73.8 / 71.7**	*63.2* / *60.3*	**67.1 / 63.3**	59.5 / 53.3	90.7 / 81.3
	Dyn.	Dr.LLM	40.3 / 24.8	*64.2* / 42.9	31.9 / 26.4	65.0 / 60.5	53.8 / 53.3	*62.2* / 52.3	52.9 / 43.4	80.6 / 66.1
		MoSO (ours)	**55.3 / 55.0**	**66.5 / 60.2**	**42.3 / 38.7**	72.9 / 68.7	**68.4 / 66.2**	63.0 / 62.9	**61.4 / 58.6**	**93.5 / 89.3**
LLaMA-3-8B-I 0.875 / 0.750		Dense	63.8	75.9	56.7	78.7	71.4	83.1	71.6	100.0
	Static	ShortGPT	62.6 / 31.0	70.2 / 45.4	49.1 / 35.1	74.5 / 63.8	69.9 / *58.8*	68.4 / 37.5	65.8 / 45.3	91.9 / 63.2
	Static	SLEB	35.0 / 25.1	65.9 / 58.9	42.2 / 33.5	**75.9 / 72.2**	65.1 / 58.4	70.6 / 53.5	59.1 / *50.3*	82.6 / *70.2*
	Static	LaCo	31.0 / 23.2	60.8 / 47.4	37.5 / 27.2	74.2 / 66.7	59.1 / 50.4	65.2 / *55.2*	54.6 / 45.0	76.3 / 62.9
	Static	BlockPruner	49.2 / 23.8	68.0 / 56.7	41.2 / 31.0	75.2 / 69.3	*66.0* / 56.7	*71.0* / 58.0	61.8 / 49.2	86.3 / 68.8
	Dyn.	Router-Tuning	35.1 / 25.4	51.5 / 34.8	31.9 / 26.7	65.6 / 58.7	60.9 / 52.6	59.9 / 53.9	50.8 / 42.0	71.0 / 58.7
	Dyn.	SkipGPT	27.7 / 27.9	61.5 / 56.2	35.8 / *34.6*	70.9 / 71.4	57.1 / 54.3	40.7 / 50.9	49.0 / 49.2	68.4 / 68.7
	Dyn.	Dr.LLM	*51.7* / *30.5*	*68.7* / *57.3*	*44.8* / 30.8	73.0 / 66.5	64.0 / 60.5	**75.1 / 62.7**	*62.9* / 51.4	*87.8* / 71.7
		MoSO (ours)	**63.7 / 62.7**	**71.7 / 64.8**	**52.6 / 47.6**	*75.0* / *70.9*	**70.8 / 69.1**	71.8 / **62.7**	**67.6 / 63.0**	**94.4 / 88.0**
Qwen2.5-7B 0.857 / 0.750		Dense	71.8	78.8	51.0	79.9	72.6	84.5	73.1	100.0
	Static	ShortGPT	*44.6* / 26.4	*69.8* / *60.3*	*45.7* / *34.9*	78.1 / **74.1**	59.5 / 54.4	*66.6* / 45.8	60.7 / 49.3	83.1 / 67.5
	Static	SLEB	43.7 / 26.4	67.5 / *60.3*	45.8 / *34.9*	77.0 / **74.1**	*61.9* / 54.4	64.9 / 45.8	60.1 / 49.3	82.2 / 67.5
	Static	LaCo	36.6 / 24.1	64.4 / 57.0	42.2 / 34.3	75.5 / 73.5	55.9 / 53.0	62.8 / 55.2	56.2 / *49.5*	76.9 / *67.7*
	Static	BlockPruner	46.5 / 28.4	70.4 / 61.1	45.0 / 35.1	**78.4** / 73.5	62.6 / 56.0	60.2 / 53.2	*60.5* / 51.2	*82.8* / 70.1
	Dyn.	Router-Tuning	41.2 / *30.0*	48.2 / 39.4	32.3 / 29.5	62.9 / 59.1	56.4 / 53.4	67.2 / 60.3	51.4 / 45.3	70.3 / 61.9
	Dyn.	SkipGPT	26.1 / 26.3	36.0 / 32.8	30.0 / 24.9	62.3 / 58.1	51.5 / 51.0	42.1 / 47.4	41.3 / 40.1	56.5 / 54.8
	Dyn.	Dr.LLM	40.9 / 31.1	46.7 / 37.9	32.1 / 30.2	61.4 / 60.3	52.2 / *54.5*	61.2 / *58.8*	49.1 / 45.5	67.1 / 62.2
		MoSO (ours)	**60.0 / 52.1**	**71.6 / 63.4**	**50.4 / 41.1**	*77.5* / *73.4*	**64.9 / 60.1**	**75.0 / 66.9**	**66.6 / 59.5**	**91.0 / 81.4**
Mistral-7B 0.875 / 0.750		Dense	59.5	81.1	53.8	82.1	74.2	83.7	72.4	100.0
	Static	ShortGPT	59.0 / 52.9	75.4 / 59.7	47.3 / 37.3	77.9 / 69.3	70.9 / 67.1	80.4 / 71.7	68.5 / 59.7	94.6 / 82.4
	Static	SLEB	42.5 / 26.1	70.9 / *60.9*	41.0 / 33.5	77.4 / **74.4**	67.3 / 60.8	77.8 / 54.4	*62.8* / 51.7	*86.8* / 71.4
	Static	LaCo	23.5 / 25.0	68.0 / 52.6	34.7 / 30.6	77.3 / 67.7	60.0 / 56.4	64.1 / 54.4	54.6 / 47.8	75.4 / 66.0
	Static	BlockPruner	33.4 / 24.1	*72.3* / 57.2	*43.0* / 31.9	*77.7* / 70.6	*68.7* / *62.4*	*78.3* / *69.8*	62.2 / 52.7	86.0 / 72.8
	Dyn.	Router-Tuning	24.9 / 25.0	29.1 / 36.9	23.9 / 27.6	53.7 / 56.3	51.1 / 51.9	56.5 / 48.3	39.9 / 41.0	55.1 / 56.6
	Dyn.	SkipGPT	26.7 / 24.3	52.3 / 47.2	26.4 / 25.5	63.8 / 59.8	52.3 / 50.3	61.9 / 48.7	47.2 / 42.6	65.2 / 58.9
	Dyn.	Dr.LLM	*48.2* / *34.0*	62.3 / 63.5	29.5 / *35.6*	56.7 / *72.7*	52.2 / 59.0	74.7 / 68.9	53.9 / *55.6*	74.5 / *76.8*
		MoSO (ours)	**59.2 / 58.7**	**75.5 / 66.4**	**48.9 / 42.7**	**78.4** / 73.0	**71.8 / 70.8**	**84.4 / 82.0**	**69.7 / 65.6**	**96.3 / 90.6**

Baselines. We compare against four static pruners – ShortGPT (Men et al., 2025), SLEB (Song et al., 2024), LaCo (Yang et al., 2024) and BlockPruner (Zhong et al., 2025) – and three learned routers retrofitted to a frozen model, Router-Tuning (He et al., 2025), SkipGPT (Zhao et al., 2025) and Dr.LLM (Heakl et al., 2026), each evaluated at its own measured compute. Dr.LLM is run with its public code, supervised by paths sampled at the target budget (Appendix D.1).

4.2 Main results

MoSO attains the highest Avg_6 in every setting, and its lead grows as the budget tightens. At the reported operating points of Table 1, MoSO achieves the highest average accuracy in all



(a) Frontier on LLaMA-3-8B-I. (b) Retention over five LLMs. (c) Lead over the best baseline.

Figure 3: **MoSO degrades gracefully, and its lead grows as the budget tightens**. **(a)** Accuracy against measured compute on LLaMA-3-8B-Instruct. **(b)** Share of dense accuracy kept, averaged over the five LLMs: at the tight budget MoSO keeps 88.6%, against 73.5% for the strongest baseline, ShortGPT. **(c)** Margin of MoSO over the best baseline in each setting of Table 1.

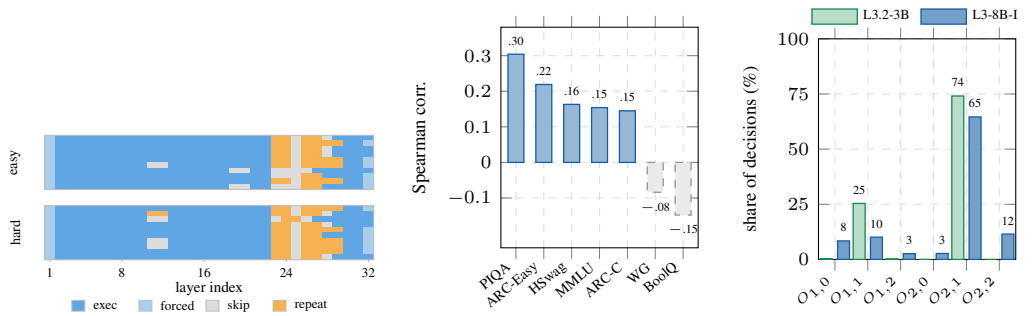
ten model–budget settings, and the best score on most individual benchmarks. Its margin over the strongest baseline is about 1 to 6 points at the loose budget and grows at the tight budget to between about 5 and 12 points on every model (Figure 3c). Averaged over the five models, MoSO keeps close to 90% of the original accuracy at the tight budget, about 15 points more than the best baseline (Figure 3b), and on LLaMA-3-8B-Instruct its accuracy declines gently with compute while the baselines fall off sharply (Figure 3a). Per model, MoSO keeps at least 91% of dense accuracy at the loose budget and at least 81% at the tight one. The training data do not by themselves raise accuracy: with routing switched off, so that every layer is executed, the trained model stays within about one point of the original (Appendix E.1).

The advantage holds against the official Dr.LLM and comes with wall-clock savings. A stronger test for a learned router is a comparison with a published router trained with its authors' code and default settings. We train Dr.LLM this way on three models: on LLaMA-3.1-8B-Instruct and Qwen2.5-7B-Instruct, MoSO is more accurate at every budget while executing fewer layers, and on LLaMA-3.2-3B it is more accurate than Dr.LLM at its default setting while using substantially less compute (Table 7, Appendix E.3). Because MoSO removes entire layers from the forward pass, its compute savings also translate directly into lower latency when a batch shares one schedule: on LLaMA-3-8B-Instruct, skipping a quarter of the layers speeds up both prefill and decoding by about 1.3×, and the router's own cost is negligible (Appendix E.5). Training takes about an hour on a single GPU and needs no per-budget calibration search or search-generated supervision.

4.3 The learned schedule matters

Other methods' layer choices cost accuracy under the same training. We isolate the contribution of the routing schedule by holding everything else fixed and changing only *which layers are skipped* (Figure 5, Appendix E.2). When the identical training on LLaMA-3.2-3B is given the layers that SLEB or LaCo would remove instead of MoSO's learned schedule, it loses 2.9 and 4.1 points at $K{=}4$ and 6.2 and 9.5 points at $K{=}7$ (Figure 5a): the cost of a poor layer choice grows as the budget tightens. At $K{=}7$ they retain only 80% and 75% of dense accuracy, against 89% for the learned schedule. Under this identical training setup, retraining does not compensate for these alternative layer choices of SLEB and LaCo.

On Qwen2.5-7B, the learned schedule improves on the Block Influence schedule. The learned schedule recovers the layer choice of ShortGPT's Block Influence (BI) criterion where that criterion is sound: on LLaMA-3.2-3B the router converges to the layers BI selects. It departs from BI where BI is known to fail. On Qwen2.5-7B, where BI ranks knowledge-bearing mid-network layers as least important, the learned schedule overlaps BI least (0.60 and 0.40 at $K{=}4$ and 7) and is more accurate under the identical training (66.6 vs. 65.2 and 59.5 vs. 57.7 Avg_6; Figure 5b), while on Mistral-7B, where BI is sound, the two schedules coincide. Deciding jointly which computation to keep thus matters most where layer-wise criteria fail.



(a) Per-prompt schedules (MMLU). (b) Difficulty vs. executed depth. (c) Operator usage.

Figure 4: **Dynamic routing over segment operators** (Appendix E.4). **(a)** Operator per layer for the 10 easiest (top) and 10 hardest (bottom) MMLU prompts on LLaMA-3-8B-Instruct (*forced*: outside the router). **(b)** Spearman correlation of prompt difficulty (dense gold loss) with executed depth. **(c)** Operator usage. The LLaMA-3-8B-Instruct router (λ_{compute}=0.10) runs 1.08× the dense layers.

4.4 Dynamic routing and the mixture of segments

The router spends computation where inputs need it. For this routing analysis, we train the six-operator MoSO router with the analysis recipe of Appendix E.4 (Figure 4). Within a benchmark, harder prompts receive more computation: on MMLU the router replaces skips with repeats for the hardest prompts (Figure 4a; 50 repeated and 20 skipped cells against 40 and 26 for the easiest, 35.0 vs. 33.4 layers on average), and executed depth rises with prompt difficulty on five of seven benchmarks (Figure 4b). No difficulty signal is used in training; the per-prompt allocation emerges from the routing objective, so executed depth follows the input rather than a fixed share of the budget.

The router routes over segments. The router's decisions are expressed mostly through two-layer segment operators (Figure 4c): treating a pair of layers as one unit is its dominant action on both models, accounting for 74% of decisions on LLaMA-3.2-3B and 79% on LLaMA-3-8B-Instruct. On LLaMA-3-8B-Instruct the mix also depends on the task: BoolQ and MMLU are skipped most and the two ARC tasks are repeated most (Appendix E.4; χ^2=355.9, 30 degrees of freedom). Repeats (14.1% of decisions) slightly outnumber skips (11.1%) on this model, and the segment repeat $O_{2,2}$ alone accounts for 11.5%, so the router uses the operator that Proposition 3 separates from the ℓ=1 slice. As Theorem 1 prescribes, skipped segments mostly change the hidden state little, and since the bound also weighs each change by its amplification A_t, the schedules agree with BI where that criterion is sound and depart from it where it fails (§4.3). Routing over segments thus lets a single router skip, execute and reuse layer pairs within one schedule.

5 Conclusion

We have recast depth reduction in large language models as learning a routing schedule over segment operators – actions that skip, run or repeat short groups of consecutive layers. This view places static pruning, layer skipping, skip-or-repeat routing and looped computation in one action space, which we prove, under explicit linear-block conditions, to be strictly richer than per-layer skip, execute and repeat; MoSO learns this schedule on a frozen model with a router that makes all decisions jointly. Across five LLMs from three families, MoSO attains the highest average accuracy at every reported operating point, and its lead widens as the budget tightens. Controlled experiments confirm that the learned schedule contributes to it: other methods' layer choices lose accuracy under the same training, and the router recovers a sound layer-importance criterion and improves on a failing one.

Several directions follow. Combining MoSO, which reduces computation, with width pruning or quantization, which also reduce memory, is a natural route to methods that save both. Serving per-input schedules efficiently in large batches calls for grouping requests that share a schedule. Finally, extending the evaluation from log-likelihood benchmarks to long-form generation, and the segment-operator view to other architectures, would test how far learned routing schedules carry. The operator family also extends to longer segments and more repetitions (Appendix B.2).

Ethics Statement

This study does not involve human subjects, personally identifiable information, or the collection of any new data. All experiments are run on publicly released pretrained checkpoints and publicly available evaluation benchmarks, used under their respective licenses and for their intended purpose of model evaluation. Our method reduces the inference cost of existing models rather than extending their capabilities, so it neither introduces new harmful capabilities nor removes existing safety behaviour by design. We identified no new ethical risks; because altering model computation may affect safety-relevant behaviour, we recommend re-running safety evaluations on routed models before deployment. We have complied with the ICLR Code of Ethics.

Reproducibility Statement

We have made the following efforts to ensure this work is reproducible. Complete proofs of every theoretical claim in §3 are given in Appendices C.1, C.2, C.3, C.4, C.5 and C.6, each stating its assumptions explicitly. The method is specified in full in Appendix B, with the router architecture in Appendix B.3 and the two-phase training recipe, hyperparameters and optimizer schedule in Appendix B.3.1. The experimental protocol – backbones and their pinned revisions, evaluation harness and version, task list, the compute-accounting protocol and BlockPruner's FLOP matching, and the numerical precision and provenance of each measurement – is in Appendix D.1. Per-task scores underlying every main-text table are reported in Appendix E.1 rather than only as aggregates. We additionally supply a provenance manifest mapping every number printed in this paper to the result file it was computed from. **Our code is provided as supplementary material for the review process and will be released publicly upon acceptance**, together with the configuration files and evaluation scripts needed to reproduce the reported cells.

AI Use Statement

We used generative AI tools in the ways declared in our submission form. *First, to aid and polish writing*: grammar, wording, phrasing and the tightening of prose we had already written. *Second, to draft sections of the paper*: initial drafts of some passages of the manuscript, which the authors then verified, edited and rewrote as needed. We also used AI coding assistants for part of the implementation and the experiments – writing and debugging parts of the training and evaluation code, running experiment scripts, aggregating results and producing figures and tables from measured results. We also used generative AI tools to provide critical feedback on manuscript presentation and experimental methodology; all such feedback was reviewed by the authors, who decided which of its suggestions to incorporate into the paper.

We did *not* use generative AI tools to generate the research idea or to design the method. Every number in this manuscript comes from a logged run of our code, and reported values were checked against their source artifacts. All AI-assisted text and code were reviewed by the authors, who take full responsibility for the final content of this work, including all text, claims and artifacts.

References

Mohammad Ali Alomrani, Yingxue Zhang, Derek Li, Qianyi Sun, Soumyasundar Pal, Zhanguang Zhang, Yaochen Hu, Rohan Deepak Ajwani, Antonios Valkanas, Raika Karimi, et al. Reasoning on a budget: A survey of adaptive and controllable test-time compute in LLMs. *arXiv preprint arXiv:2507.02076*, 2025.

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 10865–10873, 2024.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *International Conference on Learning Representations*, 2024.

Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, and Se-Young Yun. Mixture-of-recursions: Learning

dynamic recursive depths for adaptive token-level computation. In *Advances in Neural Information Processing Systems*, 2025.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *AAAI Conference on Artificial Intelligence*, 2020.

Vladimír Boža. Fast and effective weight update for pruned large language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=1hcpXd9Jir.

Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. Flextron: Many-in-one flexible large language model. In *International Conference on Machine Learning*, 2024.

Yilong Chen, Junyuan Shang, Zhenyu Zhang, Yanxi Xie, Jiawei Sheng, Tingwen Liu, Shuohuan Wang, Yu Sun, Hua Wu, and Haifeng Wang. Inner thinking transformer: Leveraging dynamic depth scaling to foster adaptive internal thinking. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28241–28259, 2025.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Aditya Cowsik, Tianyu He, and Andrey Gromov. Towards distributed neural architectures. *arXiv preprint arXiv:2506.22389*, 2025.

Róbert Csordás, Christopher D. Manning, and Christopher Potts. Do language models use their depth efficiently? In *Advances in Neural Information Processing Systems*, 2025.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019.

Devvrit, Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham Kakade, Ali Farhadi, and Prateek Jain. MatFormer: Nested transformer for elastic inference. In *Advances in Neural Information Processing Systems*, volume 37, pp. 140535–140564, 2024.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. LayerSkip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12622–12642, 2024.

Siqi Fan, Xin Jiang, Xiang Li, Xuying Meng, Peng Han, Shuo Shang, Aixin Sun, and Yequan Wang. Not all layers of LLMs are necessary during inference. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, 2025a. doi: 10.24963/ijcai.2025/566.

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. In *International Conference on Learning Representations*, 2025b.

Ying Fan, Anej Svete, and Kangwook Lee. Bridging the gap between latent and explicit reasoning with looped transformers. *arXiv preprint arXiv:2606.31779*, 2026.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Ruibo Liu, Baochen Sun, Yawen Zhang, Daiyi Peng, Xiaoyuan Guo, and V. S. Subrahmanian. MoLA: MoE LoRA with layer-wise expert allocation. In *Findings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2025.

Zitian Gao, Yilong Chen, Yihao Xiao, Xinyu Yang, Ran Tao, Joey Zhou, and Bryan Dai. Loop the loopies! *arXiv preprint arXiv:2607.16051*, 2026.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *Advances in Neural Information Processing Systems*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. The unreasonable ineffectiveness of the deeper layers. In *International Conference on Learning Representations*, 2025.

Shwai He, Tao Ge, Guoheng Sun, Bowei Tian, Xiaoyang Wang, and Dong Yu. Router-tuning: A simple and effective approach for dynamic depth. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1925–1938, 2025.

Ahmed Heakl, Martin Gubri, Salman Khan, Sangdoo Yun, and Seong Joon Oh. Dr.LLM: Dynamic layer routing in LLMs. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=3ow7tq0O3l.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems*, 2022.

Benhao Huang, Chufan Shi, Junlin Chen, Shicheng Wen, Zhengzhong Liu, Eric Xing, and Xuezhe Ma. Towards looped models done right – part I: Topology, input injection, recurrent-state design. IFM Research blog, https://ifm-research.notion.site/Towards-Looped-Models-Done-Right-3ade511912ec8128987dfeb7a5580043, 2026.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, 2016.

Gagan Jain, Nidhi Hegde, Aditya Kusupati, Arsha Nagrani, Shyamal Buch, Prateek Jain, Anurag Arnab, and Sujoy Paul. Mixture of nested experts: Adaptive processing of visual tokens. *Advances in Neural Information Processing Systems*, 37:58480–58497, 2024.

Ajay Kumar Jaiswal, Bodun Hu, Lu Yin, Yeonju Ro, Tianlong Chen, Shiwei Liu, and Aditya Akella. FFN-SkipLLM: A hidden gem for autoregressive decoding with adaptive feed forward skipping. In *Conference on Empirical Methods in Natural Language Processing*, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Junmo Kang, Leonid Karlinsky, Hongyin Luo, Zhen Wang, Jacob A. Hansen, James R. Glass, David Daniel Cox, Rameswar Panda, Rogerio Feris, and Alan Ritter. Self-MoE: Towards compositional large language models with self-specialized experts. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=IDJUscOjM3.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. Shortened LLaMA: Depth pruning for large language models with comparison of retraining methods. *arXiv preprint arXiv:2402.02834*, 2024.

Steven Kolawole, Lucio Dery, Jean-François Kagy, Virginia Smith, Graham Neubig, and Ameet Talwalkar. Everybody prune now: Structured pruning of LLMs with only forward passes, 2026. URL https://arxiv.org/abs/2402.05406.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Conference on Language Modeling*, 2025.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021.

Shuzhen Li, Yifan Zhang, Jiacheng Guo, Quanquan Gu, and Mengdi Wang. DeepLoop: Depth scaling for looped transformers. *arXiv preprint arXiv:2607.13491*, 2026a.

Ziyue Li, Yang Li, and Tianyi Zhou. Skip a layer or loop it? Test-time depth adaptation of pretrained LLMs. *arXiv preprint arXiv:2507.07996*, 2025.

Ziyue Li, Yang Li, and Tianyi Zhou. Skip a layer or loop it? Learning program-of-layers in LLMs. In *International Conference on Machine Learning*, 2026b. URL https://arxiv.org/abs/2606.06574.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision*, 2017.

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. In *Annual Meeting of the Association for Computational Linguistics*, 2024.

Xuan Luo, Weizhi Wang, and Xifeng Yan. Adaptive layer-skipping in pre-trained LLMs. In *Second Conference on Language Modeling*, 2025.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-Pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems*, 2023.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. ShortGPT: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20192–20204, 2025.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Fabrizio Sandri, Elia Cunegatti, and Giovanni Iacca. 2SSP: A two-stage framework for structured pruning of LLMs. *Transactions on Machine Learning Research*, 2025.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, 2025.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems*, 2022.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

Shoaib Ahmed Siddiqui, Xin Dong, Greg Heinrich, Thomas Breuel, Jan Kautz, David Krueger, and Pavlo Molchanov. A deeper look at depth pruning of LLMs. In *International Conference on Machine Learning Workshop on Theoretical Foundations of Foundation Models*, 2024.

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. SLEB: Streamlining LLMs through redundancy verification and elimination of transformer blocks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 46136–46155, 2024.

Qi Sun, Marc Pickett, Aakash Kumar Nain, and Llion Jones. Transformer layers as painters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25219–25227, 2025.

Rachel Teo and Tan Minh Nguyen. MoLEx: Mixture of layer experts for fine-tuning with sparse upcycling. In *International Conference on Learning Representations*, 2025.

Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. DART-Math: Difficulty-aware rejection tuning for mathematical problem-solving. In *Advances in Neural Information Processing Systems*, volume 37, pp. 7821–7846, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Neeraj Varshney, Agneet Chatterjee, Mihir Parmar, and Chitta Baral. Investigating acceleration of LLaMA inference by enabling intermediate layer decoding via instruction tuning with 'LITE'. In *Findings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2024.

Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *European Conference on Computer Vision*, 2018.

Andreas Veit, Michael J. Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems*, 2016.

Zihan Wang, Rui Pan, Jiarui Yao, Robert Csordas, Linjie Li, Lu Yin, Jiajun Wu, Tong Zhang, Manling Li, and Shiwei Liu. Chain-of-experts: Unlocking the communication power of mixture-of-experts models. *arXiv preprint arXiv:2506.18945*, 2025.

Qiong Wu, Zhaoxi Ke, Yiyi Zhou, Xiaoshuai Sun, and Rongrong Ji. Routing experts: Learning to route dynamic experts in existing multi-modal large language models. In *International Conference on Learning Representations*, 2025.

Xun Wu, Shaohan Huang, and Furu Wei. Mixture of LoRA experts. In *International Conference on Learning Representations*, 2024.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared LLaMA: Accelerating language model pre-training via structured pruning. In *International Conference on Learning Representations*, 2024.

Ziyang Xiao, Dongxiang Zhang, Yangjun Wu, Lilin Xu, Yuan Jessica Wang, Xiongwei Han, Xiaojin Fu, Tao Zhong, Jia Zeng, Mingli Song, and Gang Chen. Chain-of-experts: When LLMs meet complex operations research problems. In *International Conference on Learning Representations*, 2024.

Yifei Yang, Zouying Cao, and Hai Zhao. LaCo: Large language model pruning via layer collapse. In *Findings of the Conference on Empirical Methods in Natural Language Processing*, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Annual Meeting of the Association for Computational Linguistics*, 2019.

Yang Zhang, Yawei Li, Xinpeng Wang, Qianli Shen, Barbara Plank, Bernd Bischl, Mina Rezaei, and Kenji Kawaguchi. FinerCut: Finer-grained interpretable layer pruning for large language models. In *Neural Information Processing Systems Workshop on Machine Learning and Compression*, 2024.

Yifan Zhang, Jichen Feng, and Shihan Qin. Recurrent looped transformer. Technical report, `https://github.com/yifanzhang-pro/recurrent-looped-tranformer`, 2026.

Anhao Zhao, Fanghua Ye, Yingqi Fan, Junlong Tong, Jing Xiong, Zhiwei Fei, Hui Su, and Xiaoyu Shen. SkipGPT: Each token is one of a kind. In *International Conference on Machine Learning*, 2025.

Longguang Zhong, Fanqi Wan, Ruijun Chen, Xiaojun Quan, and Liangzhi Li. BlockPruner: Fine-grained pruning for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 5065–5080, 2025.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. Mixture-of-experts with expert choice routing. In *Advances in Neural Information Processing Systems*, 2022.

A Extended Related Work and Operator-Slice Correspondence

This appendix expands the related-work discussion of §2. Following the taxonomy of adaptive test-time compute of Alomrani et al. (2025), *controllable* methods operate under a compute budget fixed at inference, while *adaptive* methods scale compute with input difficulty or model confidence. MoSO is controllable through the number K of skipped layers, and adaptive through its per-input schedule, which allocates more computation to harder inputs (§4.4).

Depth redundancy. Residual networks behave like ensembles of relatively shallow paths (Veit et al., 2016), can be trained with stochastic layer dropping (Huang et al., 2016), and admit learned depth routing via Gumbel-softmax gates in convolutional networks (Veit & Belongie, 2018). For LLMs, a block of deeper layers can be pruned with little degradation after light finetuning (Gromov et al., 2025), layers in the second half contribute much less to the residual stream than those in the first (Csordás et al., 2025), and middle layers are robust to skipping, reordering and parallel execution (Sun et al., 2025). MoSO turns this redundancy into a learned, per-input schedule rather than an offline search.

Static depth pruning. ShortGPT (Men et al., 2025) removes layers with low Block Influence; SLEB (Song et al., 2024) greedily eliminates blocks by calibration perplexity; FinerCut (Zhang et al., 2024) and BlockPruner (Zhong et al., 2025) treat attention and MLP sub-blocks as separate candidates; attention tolerates more aggressive pruning than MLP layers (Siddiqui et al., 2024), and 2SSP (Sandri et al., 2025) combines neuron and attention-block pruning. LaCo (Yang et al., 2024) merges adjacent layers, and Shortened LLaMA (Kim et al., 2024) removes whole blocks ranked by an importance score. Width-wise and joint width–depth structured reductions (SliceGPT (Ashkboos et al., 2024), FLAP (An et al., 2024), LLM-Pruner (Ma et al., 2023), Sheared LLaMA (Xia et al., 2024), Bonsai (Kolawole et al., 2026)) are complementary, and weight recovery by an ADMM-based layer-wise update (Boža, 2024) is often applied on top. These methods apply one reduced architecture to every input and need a separate calibration per budget; in the operator view they occupy $\{O_{1,0}, O_{1,1}\}$ or $\{O_{\ell,0}\}_{\ell\geq 2} \cup \{O_{1,1}\}$ with an input-independent selection rule.

Dynamic depth routing. Early-exit methods stop at intermediate layers, by a confidence criterion in CALM (Schuster et al., 2022), LITE (Varshney et al., 2024) and AdaInfer (Fan et al., 2025a), or at a fixed exit layer in LayerSkip (Elhoushi et al., 2024), which combines layer dropout and an early-exit loss with self-speculative decoding. Per-input gates skip layers: Mixture-of-Depths (Raposo et al., 2024) makes per-token skip decisions during training, FFN-SkipLLM (Jaiswal et al., 2024) skips only MLP sub-blocks by a training-free similarity test, FlexiDepth (Luo et al., 2025) learns per-token skipping with a plug-in router and adapter, Router-Tuning (He et al., 2025) gates attention sub-blocks with a lightweight router, Dr.LLM (Heakl et al., 2026) learns skip, execute or repeat decisions per layer from Monte Carlo tree search (MCTS) supervision, CoLa (Li et al., 2025) searches layer compositions online, and PoLar (Li et al., 2026b) trains a predictor on such searches to emit per-input programs that skip, keep or repeat contiguous layer segments. Related designs route inputs through shortcut paths (Wu et al., 2025), nested experts (Jain et al., 2024), self-specialized experts (Kang et al., 2025), iterated expert selection (Wang et al., 2025) or arbitrary module sequences (Cowsik et al., 2025). Apart from Dr.LLM, CoLa and PoLar, which can also repeat layers, every method in this line is confined to the binary slice $\{O_{1,0}, O_{1,1}\}$, and all but PoLar decide one layer at a time.

Adaptive and looped computation. This line treats depth as a resource to be spent rather than a fixed property of the architecture. Adaptive computation time (Graves, 2016) and Universal Transformers (Dehghani et al., 2019) established the core idea by applying a shared block a variable number of times, and a recent and fast-moving line of *looped* transformers has revived it. Geiping et al. (2025) scale test-time compute by recurring in latent space rather than in tokens; Saunshi et al. (2025) argue that many reasoning problems demand depth rather than parameters, showing a k-layer model looped m times approaches a km-layer model on reasoning tasks; Fan et al. (2026) relate the latent computation a loop performs to explicit chain-of-thought reasoning; Li et al. (2026a) show that the residual scaling rule itself must change when depth comes from repetition rather than from distinct layers, since a shared update accumulates across visits; Gao et al. (2026) study where the loop should be placed – around a single layer or around the whole model – and report that the choice materially changes the compute–quality trade-off; Zhang et al. (2026) carry the recurrent state across

tokens rather than only across depth; and Huang et al. (2026) systematically ablate the topology, input-injection and recurrent-state choices of looped models under matched compute. Orthogonally, elastic-width models (Devvrit et al., 2024; Cai et al., 2024) and mixture-of-experts (Shazeer et al., 2017; Fedus et al., 2022) adapt along a width axis rather than a depth one. *Position of* MoSO. This literature and ours meet at the $\rho \geq 2$ operators and differ in what is repeated. Looped models repeat a *shared* block, so depth is decoupled from parameters and the loop count is typically scheduled externally or learned as a single global quantity; MoSO repeats a *segment of the backbone's own distinct pretrained layers*, chosen per input by the router alongside skip and execute decisions in the same action space. The distinction has a mathematical consequence rather than only a practical one: because adjacent pretrained layers f_p and f_{p+1} are different maps, repeating the pair interleaves them as $f_{p+1}f_pf_{p+1}f_p$, which, for linear blocks, no per-layer skip, execute or repeat reproduces when their Jacobians fail to commute and their absolute determinants lie strictly on the same side of 1 (Proposition 3) – whereas repeating a shared block is self-composition and raises no such question. Two further differences follow from our setting: we keep the pretrained backbone frozen and require no pretraining with the loop in place, and repetition in MoSO is one action among six rather than the architecture itself, which lets the same router trade a repeat here against a skip there under one compute budget. Mixture-of-Recursions (Bae et al., 2025) sits between the two lines, routing tokens to different recursion depths over a shared block. Width-wise Mixture-of-Experts (Shazeer et al., 2017; Fedus et al., 2022; Lepikhin et al., 2021), adaptive expert routing (Zhou et al., 2022; Lu et al., 2024) and PEFT mixtures (Wu et al., 2024; Gao et al., 2025; Teo & Nguyen, 2025) sparsify width rather than depth, and "chains of experts" have also been used as a prompting abstraction (Xiao et al., 2024); MoSO is compatible with dense and MoE backbones alike.

Positioning of MoSO. MoSO differs from these lines in three respects. First, its action space is the segment-operator family $\mathcal{O}$, which strictly extends the binary $\{O_{1,0}, O_{1,1}\}$ and ternary $\ell{=}1$ slices through the segment repeat $O_{2,2}$ (Proposition 3); among prior methods, only PoLar also acts on multi-layer segments. Second, depth reduction is learned as a *schedule*: the router scores all candidate segments jointly, so each decision is made in the context of the schedule being built rather than one layer at a time. Third, the schedule is learned end to end from language-modelling and distillation losses on a small corpus, without search-generated supervision (as in Dr.LLM, CoLa or PoLar) and without continued pretraining. Table 2 summarizes the correspondence between prior methods and operator slices of $\mathcal{O}$, the selection rule each uses, and the capabilities each forgoes relative to $\mathcal{O}$.

Table 2: Prior depth-pruning and depth-amplification methods as instances of the parameterized operator family $\{O_{\ell,\rho}\}$ (MoSO uses $\mathcal{O}$, with $\ell, \rho \leq 2$).

Category	Method	Operator slice	Selection rule	Capability forgone w.r.t. $\mathcal{O}$
Layer-wise static skipping	ShortGPT (Men et al., 2025)	$\{O_{1,0}, O_{1,1}\}$	static, Block Influence	input-conditional; repetition; segment-level
	SLEB (Song et al., 2024)	$\{O_{1,0}, O_{1,1}\}$	static, greedy PPL	input-conditional; repetition; segment-level
Sub-block static skipping	FinerCut (Zhang et al., 2024)	$\{O_{1,0}, O_{1,1}\}$ (sub-block)	static, greedy output divergence	input-conditional; repetition; segment-level
	BlockPruner (Zhong et al., 2025)	$\{O_{1,0}, O_{1,1}\}$ (sub-block)	static, offline block selection	input-conditional; repetition; segment-level
Layer merging	LaCo (Yang et al., 2024)	$\{O_{\ell,0}\}_{\ell\geq 2} \cup \{O_{1,1}\}$	static, cosine-similarity merge	input-conditional; repetition
Dynamic per-layer routing	Dr.LLM (Heakl et al., 2026)	$\{O_{1,0}, O_{1,1}, O_{1,2}\}$	learned, MCTS-supervised	segment-level ($\ell \geq 2$)
	FlexiDepth (Luo et al., 2025)	$\{O_{1,0}, O_{1,1}\}$	learned, per-token	repetition; segment-level
	Router-Tuning (He et al., 2025)	$\{O_{1,0}, O_{1,1}\}$	learned, per-sequence	repetition; segment-level
	LayerSkip (Elhoushi et al., 2024)	$\{O_{1,0}, O_{1,1}\}$	fixed early exit, self-speculative	repetition; segment-level
Dynamic segment routing	PoLar (Li et al., 2026b)	$\{O_{\ell,\rho}\}_{\ell\leq 4,\,\rho\leq 2}$	learned, MCTS-supervised	compute budget
Token-wise routing	MoD (Raposo et al., 2024)	$\{O_{1,0}, O_{1,1}\}$ (token-wise)	learned, per-token	repetition; segment-level
Layer repetition / looped computation	Universal Trans. (Dehghani et al., 2019)	$\{O_{1,\rho}\}_{\rho\geq 1}$	static / ACT-gated	skipping; segment-level
	Looped Trans. (Fan et al., 2025b)	$\{O_{\ell,\rho}\}_{\rho\geq 1}$	length-dependent loop count	skipping; learned joint choice
	Recurrent depth (Geiping et al., 2025)	$\{O_{1,\rho}\}_{\rho\gg 1}$	externally scheduled	skipping; segment-level
	ITT, MoR (Chen et al., 2025; Bae et al., 2025)	$\{O_{1,\rho}\}_{\rho\geq 1}$ (token-wise)	learned, per-token	skipping; segment-level
Ours (MoSO)	**MoSO**	$\mathcal{O} = \{O_{\ell,\rho} : \ell \in \{1,2\}, \rho \in \{0,1,2\}\}$	**learned, joint** (ℓ, ρ)	—

B Method Details

This appendix expands §3 in the same order: the operator family and its extensions, then the router, the training recipe, pseudocode and complexity.

B.1 Operator family: full formulation

This subsection gives the full formulation of the segment-operator routing of §3.1: the router, the soft training-time mixture, the training objective, the hard-inference rule and the output-reuse schedule.

B.1.1 Operator-space router

The router ingests the current execution state $s_t = (\mathbf{h}_t, p_t, r_t)$ through three inputs: (i) the attention output $\mathbf{a}_t \in \mathbb{R}^{d_v}$ produced by the self-attention of Eq. (4), (ii) the layer-pointer embedding $\mathbf{p}_{p_t} \in \mathbb{R}^{d_r}$ at the current pointer, and (iii) the scalar remaining budget $r_t \in \mathbb{R}$. These are concatenated along the feature axis and passed through a lightweight MLP that emits logits over the operator family $\mathcal{O}$:

$$\mathbf{u}_t \;=\; \mathrm{MLP}\big(\big[\,\mathbf{a}_t\,;\,\mathbf{p}_{p_t}\,;\,r_t\,\big]\big) \;\in\; \mathbb{R}^{d_h}, \tag{8}$$

$$\mathbf{z}_t \;=\; \mathbf{W}^{(\mathcal{O})}\mathbf{u}_t + \mathbf{b}^{(\mathcal{O})} \;\in\; \mathbb{R}^{|\mathcal{O}|}, \tag{9}$$

$$\boldsymbol{\pi}_t \;=\; \mathrm{softmax}\big(\mathbf{z}_t \,/\, \tau\big) \;\in\; \Delta^{|\mathcal{O}|-1}, \tag{10}$$

where $\mathbf{W}^{(\mathcal{O})} \in \mathbb{R}^{|\mathcal{O}|\times d_h}$ and the temperature τ follows the same annealing schedule used for every router in this paper. Equation (10) is a *masked* softmax: any operator $O_{\ell,\rho}$ with $p_t + \ell - 1 > L - 1$ has its logit set to $-\infty$, so the right-boundary validity constraint of §3.1 is enforced exactly and no segment reaches the protected layer L.

Joint head. A single categorical head over all valid operators (Eq. (9)) is used throughout. It places no factorization assumption on (ℓ, ρ); its capacity relative to a factorized (length $\times$ repetition) head is the classical moment-projection identity, restated here to make the design argument precise.

Proposition 4 (Joint-vs-factorized head gap)**.** *Fix a state s_t and a target distribution $\boldsymbol{\pi}^\star \in \Delta^{|\mathcal{O}|-1}$ over the valid operators. A joint softmax head with unrestricted logits has $\inf_{\boldsymbol{\pi}} \mathrm{KL}(\boldsymbol{\pi}^\star \| \boldsymbol{\pi}) = 0$, while a factorized head is confined to $\mathcal{I} = \{\pi_\ell \otimes \pi_\rho\}$ and*

$$\min_{\pi_\ell, \pi_\rho} \mathrm{KL}\big(\boldsymbol{\pi}^\star \,\big\|\, \pi_\ell \otimes \pi_\rho\big) \;=\; I_{\boldsymbol{\pi}^\star}(\ell; \rho) \;\le\; \log 2, \tag{11}$$

the mutual information of $(\ell, \rho) \sim \boldsymbol{\pi}^\star$, attained at the marginals of $\boldsymbol{\pi}^\star$ and equal to zero iff $\boldsymbol{\pi}^\star$ is a product distribution.

The proof is in Appendix C.4. The joint head therefore imposes no factorization on the training-time distribution over operators, whose tendency to concentrate early on a single operator is counteracted by the focal-balance regularizer of Eq. (6).

B.1.2 Soft mixture training

During training, operator selection is relaxed component-wise on the three parts of the execution state $s_t = (\mathbf{h}_t, p_t, r_t)$. The hidden state is updated by a convex mixture over operator outputs, while the integer pointer is advanced by a straight-through argmax and the scalar budget is decreased by the expected operator cost:

$$\mathbf{h}_{t+1} \;=\; \sum_{(\ell,\rho)\in\mathcal{O}_{\text{valid}}(p_t)} \boldsymbol{\pi}_t(\ell,\rho)\; F^{\rho}_{p_t,\ell}(\mathbf{h}_t), \tag{12}$$

$$p_{t+1} \;=\; p_t \;+\; \bar{\ell}_t, \qquad \bar{\ell}_t \;=\; \arg\max_{\ell\in\{1,2\}} \sum_{\rho} \boldsymbol{\pi}_t(\ell,\rho), \tag{13}$$

$$r_{t+1} \;=\; r_t \;-\; \sum_{(\ell,\rho)\in\mathcal{O}_{\text{valid}}(p_t)} \boldsymbol{\pi}_t(\ell,\rho)\,\ell\,\rho. \tag{14}$$

The hidden-state mixture in Eq. (12) is differentiable and carries the training signal to the router via $\boldsymbol{\pi}_t$; the pointer update in Eq. (13) is a straight-through surrogate that lets subsequent routing steps remain well-defined while gradients flow through the soft mixture. Routing terminates when the pointer passes the last routable layer. The training pointer uses the *marginal* argmax over ℓ and hard inference (§B.1.3) the ℓ-component of the *joint* argmax; the two coincide whenever $\max_{(\ell,\rho)} \boldsymbol{\pi}_t(\ell,\rho) > 1/2$, i.e. once the distribution has sharpened. Termination is $p_t \ge L$, which stops before layer L because layer L is always executed (§3.1); the routable range is $2, \ldots, L-1$ throughout.

B.1.3 TRAINING OBJECTIVE

The router is trained with the objective of Eq. (6): the language-modelling loss, an expected-compute regularizer, the focal-balance regularizer, and $\mathcal{L}_{\text{aux}} = \lambda_{\text{KL}}\mathcal{L}_{\text{KL}} - \beta_{\text{entropy}}\mathcal{L}_{\text{entropy}}$, which combines distillation from the unreduced model (Appendix B.3.1) with an entropy bonus. Written out without the focal and distillation terms,

$$\mathcal{L}_{\text{LM}} \;+\; \lambda_{\text{compute}}\,\mathcal{L}_{\text{compute}} \;-\; \beta_{\text{entropy}}\,\mathcal{L}_{\text{entropy}}. \tag{15}$$

Operator cost and expected-compute regularizer. The cost of an operator is the number of backbone-block evaluations it performs,

$$c(O_{\ell,\rho}) \;=\; \ell\,\rho, \tag{16}$$

so $c(O_{1,0}) = c(O_{2,0}) = 0$, $c(O_{1,1}) = 1$, $c(O_{1,2}) = c(O_{2,1}) = 2$, and $c(O_{2,2}) = 4$. The expected compute over the routing trajectory is

$$\mathcal{L}_{\text{compute}} \;=\; \mathbb{E}_{\mathbf{x}}\Big[\sum_{t}\;\sum_{(\ell,\rho)\in\mathcal{O}_{\text{valid}}(p_t)}\boldsymbol{\pi}_t(\ell,\rho)\;\ell\,\rho\Big], \tag{17}$$

which is differentiable through $\boldsymbol{\pi}_t$ and pushes the router toward cheaper operators unless more expensive segment-level computation is beneficial for the language-modelling loss.

Entropy regularizer. The entropy term

$$\mathcal{L}_{\text{entropy}} \;=\; \frac{1}{N}\sum_{t=1}^{N} H\big(\boldsymbol{\pi}_t\big) \;=\; -\frac{1}{N}\sum_{t=1}^{N}\sum_{(\ell,\rho)}\boldsymbol{\pi}_t(\ell,\rho)\log\boldsymbol{\pi}_t(\ell,\rho) \tag{18}$$

encourages exploration early in training. Combined with temperature annealing, it drives the routing distribution from smooth operator mixtures toward sharper operator decisions, narrowing the soft-vs-hard gap between training and inference.

Hard inference. At inference time we use the hard operator selection

$$(\ell_t^\star,\rho_t^\star) \;=\; \arg\max_{(\ell,\rho)\in\mathcal{O}_{\text{valid}}(p_t)}\boldsymbol{\pi}_t(\ell,\rho), \tag{19}$$

and apply the corresponding operator $O_{\ell_t^\star,\rho_t^\star}$ deterministically. The execution terminates when the pointer reaches the protected layer L, which is then executed, and the final prediction is produced by the frozen prediction head of the backbone.

Output reuse under the soft mixture. A naive evaluation of all six operators at a single routing step would issue $0+1+2+0+2+4 = 9$ backbone-block forward calls (summing $c(O_{\ell,\rho})$ over $\mathcal{O}$). We exploit the segment structure to reuse intermediate activations: $f_{p_t}(\mathbf{h}_t)$ is computed once and serves $O_{1,1}$ as well as the seeds of $O_{1,2}$ and $O_{2,1}$, and $f_{p_t+1}(f_{p_t}(\mathbf{h}_t))$ (the output of $O_{2,1}$) is reused as the seed of $O_{2,2}$. With this reuse schedule, the total number of distinct backbone-block applications per routing step drops to **five** when all six operators are valid, and to two at the right boundary where only $\ell = 1$ is legal. The reuse schedule is transparent to the router: the same Eq. (12) is evaluated with shared intermediate tensors.

B.2 EXTENDING THE FAMILY BEYOND $\ell \le 2$, $\rho \le 2$

Table 3 exhibits the operator family $\mathcal{O}$ extended to segment lengths $\ell \in \{1,2,3\}$ and repetition counts $\rho \in \{0,1,2,3\}$, yielding 12 operators. Each row lists the operator $O_{\ell,\rho}$, the traditional action name it subsumes, the hidden-state update $F^{\rho}_{p,\ell}(\mathbf{h})$, the full state transition $O_{\ell,\rho}(\mathbf{h},p,r)$, the bilinear cost $c = \ell\rho$, and whether the same behaviour is expressible in the standard ternary *skip*/*execute*/*repeat* action space. The ℓ=1, $\rho\le$2 slice recovers the ternary space exactly. With identity skips, ρ=0 operators equal compositions of single-layer skips; the (ℓ,ρ)=(2, 2) operator is a strict extension of the ternary space under the hypotheses of Proposition 3.

Table 3: Unified view of layer-wise and segment-level operators under the parameterized family $\mathcal{O}$ extended to $\ell \in \{1, 2, 3\}$, $\rho \in \{0, 1, 2, 3\}$.

(ℓ, ρ)	Operator	Traditional name	$F^{\rho}_{p,\ell}(\mathbf{h})$	State transition $O_{\ell,\rho}(\mathbf{h}, p, r)$	Cost c	Ternary?
$\ell = 1$ *(single-layer operators; $\rho \leq 2$ is the ternary action space)*						
$(1, 0)$	$O_{1,0}$	*skip*	$\mathbf{h}$	$(\mathbf{h},\ p{+}1,\ r)$	0	✓
$(1, 1)$	$O_{1,1}$	*execute*	$f_p(\mathbf{h})$	$(f_p(\mathbf{h}),\ p{+}1,\ r{-}1)$	1	✓
$(1, 2)$	$O_{1,2}$	*repeat*	$f_p \circ f_p(\mathbf{h})$	$(f_p^2(\mathbf{h}),\ p{+}1,\ r{-}2)$	2	✓
$(1, 3)$	$O_{1,3}$	*triple-repeat*	$f_p^3(\mathbf{h})$	$(f_p^3(\mathbf{h}),\ p{+}1,\ r{-}3)$	3	×
$\ell = 2$ *(2-layer segment operators — atomic segment skipping, execution, and reuse)*						
$(2, 0)$	$O_{2,0}$	segment-skip	$\mathbf{h}$	$(\mathbf{h},\ p{+}2,\ r)$	0	= (2×skip)
$(2, 1)$	$O_{2,1}$	segment-execute	$f_{p+1} \circ f_p(\mathbf{h})$	$(F_{p,2}(\mathbf{h}),\ p{+}2,\ r{-}2)$	2	= (2×exec)
$(2, 2)$	$O_{2,2}$	segment-repeat	$(f_{p+1} \circ f_p)^2(\mathbf{h})$	$(F^2_{p,2}(\mathbf{h}),\ p{+}2,\ r{-}4)$	4	×
$(2, 3)$	$O_{2,3}$	segment-triple-repeat	$F^3_{p,2}(\mathbf{h})$	$(F^3_{p,2}(\mathbf{h}),\ p{+}2,\ r{-}6)$	6	×
$\ell = 3$ *(3-layer segment operators — block-level skipping, execution, and reuse)*						
$(3, 0)$	$O_{3,0}$	block-skip	$\mathbf{h}$	$(\mathbf{h},\ p{+}3,\ r)$	0	= (3×skip)
$(3, 1)$	$O_{3,1}$	block-execute	$f_{p+2} \circ f_{p+1} \circ f_p(\mathbf{h})$	$(F_{p,3}(\mathbf{h}),\ p{+}3,\ r{-}3)$	3	= (3×exec)
$(3, 2)$	$O_{3,2}$	block-repeat	$F^2_{p,3}(\mathbf{h})$	$(F^2_{p,3}(\mathbf{h}),\ p{+}3,\ r{-}6)$	6	×
$(3, 3)$	$O_{3,3}$	block-triple-repeat	$F^3_{p,3}(\mathbf{h})$	$(F^3_{p,3}(\mathbf{h}),\ p{+}3,\ r{-}9)$	9	×

Table 4: Default router and training hyperparameters (two training phases; Appendix B.3.1).

Parameter	**Value**
Router embedding dim (d_r)	128
Attention layers / heads	2 / 4
Per-head key / value dim (d_k=d_v)	32
Trunk MLP hidden dim (d_h)	128
Per-layer positional embedding	learnable, d_r-dim
Windowed pool count (N_w)	2
Operator family	$\ell \in \{1, 2\}, \rho \in \{0, 1, 2\}$
Output head	joint, $\lvert\mathcal{O}\rvert$=6
Total router params	0.93M (3B) / 1.09M (32-layer 8B)
Optimizer	AdamW
Learning rate	10^{-3} / $5{\times}10^{-4}$ (two phases), cosine, 5% warmup
Weight decay	0.01
Gradient clip	1.0
Training steps	2000 per phase (two phases)
Total training tokens seen	$\approx$ 8.3 M (3B) to 9.1 M (7B) per run
Sequence length	512
Effective batch size	8
Temperature schedule (τ)	$1.0 \rightarrow 0.2$ (linear)
λ_{compute}	target-K adjusted; cosine annealed over 30% of training
β_{entropy}	5×10^{-3}
λ_{focal}	0.2
Focal exponent (γ)	2.0
Distillation weight λ_{KL}	1.0 / 2.0 (two phases)

B.3 Router Architecture and Training Recipe

B.3.1 Training Recipe

All results of Table 1 use one recipe (hyperparameters in Table 4). Training data are 7.5K ARC+DART-Math examples for base backbones and 50K Tulu-3 examples for the instruction-tuned backbones (Appendix D.1). Training minimizes Eq. (6) in two phases of 2000 steps: all

trainable parameters are first learned jointly, and training then continues at a lower learning rate and a larger distillation weight to refine the model for the learned schedule.

Optimization details. We use AdamW (weight decay 0.01, gradient clip 1.0) with a cosine schedule and 5% linear warmup. Micro-batch size is 2 and gradient accumulation is 4 (effective batch 8). The Block Influence (BI) scores used as the router's layer-importance prior are calibrated once before training on 32 random 256-token chunks from WikiText-2. Two initialization settings – the scale ω of the BI warm start of the skip-operator logits and a monotone depth prior κ that biases these logits toward later layers – are selected per backbone by LM loss on a held-out split of the training pool, never on evaluation data.

B.4 Pseudocode and computational complexity

Algorithms 1 and 2 give the training step and the inference pass of MoSO.

B.4.1 Training and inference pseudocode

Algorithm 1 MoSO training step (one minibatch).

Require: Minibatch $\mathbf{x}$; frozen backbone $\{f_i\}_{i=1}^{L}$ and LM head; trainable parameters θ; operator family $\mathcal{O}$; temperature schedule for τ; initial budget $r_{\max}$ set by K; loss weights of Eq. (6).

1: $\mathbf{h} \leftarrow f_1(\mathrm{Embed}(\mathbf{x}))$; $p \leftarrow 2$; $r_1 \leftarrow r_{\max}$; $t \leftarrow 1$ ▷ *layer 1 always executed*
2: **while** $p \le L-1$ **do**
3: $\mathbf{e}_t \leftarrow$ step embedding of the pooled state $\mathbf{h}$, the position p and the layer-importance prior
4: $\mathbf{a}_t \leftarrow$ self-attention over $\mathbf{e}_1, \dots, \mathbf{e}_t$ ▷ *Eq.* (4)
5: $\boldsymbol{\pi}_t \leftarrow \mathrm{softmax}\big(\mathrm{MLP}([\mathbf{a}_t; \mathbf{p}_p; r_t])/\tau\big)$ over $\mathcal{O}_{\text{valid}}(p)$ ▷ *Eq.* (5)
6: $\mathbf{h} \leftarrow \sum_{(\ell,\rho)} \boldsymbol{\pi}_t(\ell,\rho)\, F^{\rho}_{p,\ell}(\mathbf{h})$ ▷ *soft mixture, Eq.* (7)
7: $p \leftarrow p + \arg\max_{\ell} \sum_{\rho} \boldsymbol{\pi}_t(\ell,\rho)$; $r_{t+1} \leftarrow r_t - \sum_{(\ell,\rho)} \boldsymbol{\pi}_t(\ell,\rho)\, \ell\rho$; $t \leftarrow t+1$ ▷ *straight-through pointer update*
8: **end while**
9: $\mathbf{y} \leftarrow \mathrm{LMHead}(f_L(\mathbf{h}))$ ▷ *layer L always executed*
10: Compute $\mathcal{L}_{\mathrm{MoSO}}$ of Eq. (6) from $\mathbf{y}$ and $\{\boldsymbol{\pi}_t\}_t$
11: $\theta \leftarrow \mathrm{AdamW}\big(\theta,\, \nabla_\theta \mathcal{L}_{\mathrm{MoSO}}\big)$ ▷ *backbone frozen*

Algorithm 2 MoSO inference (deterministic argmax routing).

Require: Tokens $\mathbf{x}$; frozen backbone $\{f_i\}$ and LM head; trained parameters θ; inference temperature τ (default 0.2); initial budget $r_{\max}$ set by K.

1: $\mathbf{h} \leftarrow f_1(\mathrm{Embed}(\mathbf{x}))$; $p \leftarrow 2$; $r \leftarrow r_{\max}$; $t \leftarrow 1$
2: **while** $p \le L-1$ **do**
3: Compute $\mathbf{e}_t$, $\mathbf{a}_t$ and the operator logits as in Algorithm 1 (lines 3–5)
4: Mask invalid operators ($p+\ell-1 > L-1$); $\boldsymbol{\pi}_t \leftarrow \mathrm{softmax}(\mathrm{logits}_t/\tau)$
5: $(\ell_t^\star, \rho_t^\star) \leftarrow \arg\max_{(\ell,\rho)\in\mathcal{O}_{\text{valid}}(p)} \boldsymbol{\pi}_t(\ell,\rho)$
6: $\mathbf{h} \leftarrow F^{\rho_t^\star}_{p,\ell_t^\star}(\mathbf{h})$ ▷ *single operator; no soft mixing*
7: $p \leftarrow p + \ell_t^\star$; $r \leftarrow r - \ell_t^\star \rho_t^\star$; $t \leftarrow t+1$
8: **end while**
9:
10: **return** $\mathrm{LMHead}(f_L(\mathbf{h}))$

Training versus inference. The two algorithms share the router computation and differ in three places. Training updates the hidden state with the soft mixture over all valid operators, so gradients reach every operator, whereas inference applies the single argmax operator; training advances the pointer by a straight-through marginal-ℓ argmax, inference by the joint argmax; and training

accumulates the loss terms over the batch, whereas inference is a single forward pass. The validity mask guarantees that $\boldsymbol{\pi}_t$ is supported on $\mathcal{O}_{\text{valid}}(p)$ (at $p{=}L{-}1$ only $\{O_{1,0}, O_{1,1}, O_{1,2}\}$ remain, and the joint head reduces to a ternary head). Layers 1 and L are executed outside the loop, the pointer starts at $p{=}2$ and advances by at least one per step, so $N \le L-2$. The pretrained weights and the LM head receive no gradient updates.

B.4.2 Computational complexity

Let d be the hidden dimension, S the sequence length, $C_{\text{layer}} = \mathrm{O}(S^2 d + S d^2)$ the cost of one backbone block, and $N \le L-2$ the number of routing decisions. One router step costs $\mathrm{O}(S\,d\,d_r + t\,d_r d_k + d_h^2)$ (pooling and embedding, attention over the t previous steps, MLP trunk), which for $d_r{=}d_h{=}128$ and $d_k{=}32$ is negligible next to C_{layer}. Under the reuse schedule (§B.1.3) a training step performs at most 5 distinct block evaluations (2 at the right boundary), while inference takes one operator path with $\ell_t \rho_t \in \{0,1,2,4\}$ block evaluations. Hence

$$C_{\text{train}}(\mathbf{x}) \;\le\; \big(5(L-2)+2\big)\,C_{\text{layer}} + \sum_{t=1}^{N} C_{\text{router}}^{(t)} \;=\; \mathrm{O}\big(L\,(S^2 d + S d^2)\big), \tag{20}$$

$$C_{\text{infer}}(\mathbf{x}) \;=\; C(\mathbf{x})\,C_{\text{layer}} + \sum_{t=1}^{N} C_{\text{router}}^{(t)} \;=\; \mathrm{O}\big(C(\mathbf{x})\,(S^2 d + S d^2) + L \max_t C_{\text{router}}^{(t)}\big), \tag{21}$$

where $C(\mathbf{x}) = 2 + \sum_{t=1}^{N} \ell_t^\star \rho_t^\star$ counts the block evaluations actually performed, including the two protected layers (its mean over inputs is L_{eff}, Appendix D.1). Training therefore costs at most $5\times$ a dense forward pass ($2\times$ on the $\ell{=}1$ slice), and stores up to five segment outputs per step for the convex mixture; inference is single-path, with ideal speedup $L/C(\mathbf{x})$ (measured speedups in Appendix E.5). The router has $\mathrm{O}(d\,d_r + L\,d_r + d_r^2 + d_h^2 + d_h|\mathcal{O}|)$ parameters: 0.93M on LLaMA-3.2-3B and 1.09M on the 32-layer backbones, 0.01–0.03% of the backbone.

C Detailed Proofs of Operator-Family Theorems

This appendix gives the complete proofs of Theorem 1, Proposition 2, Proposition 3 with Corollary 5, Proposition 4, Lemma 6 and Proposition 7.

C.1 Proof of Theorem 1 (Operator-Trajectory Approximation)

Assumption 1 (Backbone regularity)**.** *There is a set $\mathcal{D}_h \subseteq \mathbb{R}^{S\times d}$ containing the embedding $\mathbf{h}_0$ and mapped into itself by every block, $f_i(\mathcal{D}_h) \subseteq \mathcal{D}_h$, such that for all $\mathbf{h}, \mathbf{h}' \in \mathcal{D}_h$*

$$\|f_i(\mathbf{h}) - f_i(\mathbf{h}')\| \le (1+\eta_i)\,\|\mathbf{h} - \mathbf{h}'\|$$

for constants $\eta_i \ge 0$. ($\mathcal{D}_h = \mathbb{R}^{S\times d}$ needs globally Lipschitz blocks; softmax attention is not, so $\mathcal{D}_h$ is then a bounded invariant set.) Forward invariance guarantees that every routed, full and hybrid state used below lies in $\mathcal{D}_h$. We write $\mathrm{Lip}(\cdot)$ for Lipschitz constants on $\mathcal{D}_h$.

Setup and notation. The segment composition is $F_{p,\ell} = f_{p+\ell-1} \circ \cdots \circ f_p$ and the operator is $O_{\ell,\rho}(\mathbf{h}, p, r) = (F_{p,\ell}^{\rho}(\mathbf{h}),\, p+\ell,\, r-\ell\rho)$, with the convention $F^0 = \mathrm{id}$. Layers 1 and L are protected, so a trajectory $(\ell_t, \rho_t)_{t=1}^{N}$ with $\sum_t \ell_t = L-2$ visits pointers $p_1{=}2$, $p_{t+1}{=}p_t{+}\ell_t$, $p_{N+1}{=}L$, and each routable layer index $2,\dots,L{-}1$ appears in exactly one segment. We write $\Lambda_t := \mathrm{Lip}(F_{p_t,\ell_t})$ for the segment Lipschitz constant at step t (the symbol used in Theorem 1). By submultiplicativity of Lipschitz constants under composition, the segment $F_{p,\ell}$ satisfies

$$\mathrm{Lip}(F_{p,\ell}) \;\le\; \prod_{j=0}^{\ell-1}(1+\eta_{p+j}) \;\le\; e^{\sum_{j=0}^{\ell-1}\eta_{p+j}}, \tag{22}$$

the last bound following from $1+x \le e^x$. The proof below uses Λ_t symbolically; any of the upper bounds in Eq. (22) can be substituted to obtain a closed-form result in terms of the per-block Lipschitz excesses $\{\eta_i\}$.

Define the routed hidden states $\mathbf{y}_0 := f_1(\mathbf{h}_0)$, $\mathbf{y}_t = F^{\rho_t}_{p_t,\ell_t}(\mathbf{y}_{t-1})$, so that $\mathbf{h}_L^{\text{routed}} = f_L(\mathbf{y}_N)$. The unrouted backbone corresponds to $\rho_t \equiv 1$:

$$\mathbf{h}_L^{\text{full}} = f_L \circ F_{p_N,\ell_N} \circ \cdots \circ F_{p_1,\ell_1}(\mathbf{y}_0),$$

since the segments partition $\{2, \ldots, L-1\}$ and composition in the listed order recovers $f_L \circ \cdots \circ f_1$.

Step 1: Per-step operator deviation. At step t, define the deviation

$$\delta_t \;:=\; F^{\rho_t}_{p_t,\ell_t}(\mathbf{y}_{t-1}) - F_{p_t,\ell_t}(\mathbf{y}_{t-1}). \tag{23}$$

We bound $\|\delta_t\|$ case by case for $\rho_t \in \{0, 1, 2\}$.

Case $\rho_t = 0$. By convention $F^0 = \mathrm{id}$, so

$$\delta_t \;=\; \mathbf{y}_{t-1} - F_{p_t,\ell_t}(\mathbf{y}_{t-1}), \qquad \|\delta_t\| \;=\; \|F_{p_t,\ell_t}(\mathbf{y}_{t-1}) - \mathbf{y}_{t-1}\|. \tag{24}$$

This identity is *exact* with coefficient $m_t(0) = 1$.

Case $\rho_t = 1$. Trivially $\delta_t = 0$, with coefficient $m_t(1) = 0$.

Case $\rho_t = 2$. Write $\mathbf{u} := F_{p_t,\ell_t}(\mathbf{y}_{t-1})$ and $\mathbf{v} := \mathbf{y}_{t-1}$. Then

$$\delta_t \;=\; F^2_{p_t,\ell_t}(\mathbf{v}) - F_{p_t,\ell_t}(\mathbf{v}) \;=\; F_{p_t,\ell_t}(\mathbf{u}) - F_{p_t,\ell_t}(\mathbf{v}). \tag{25}$$

Applying the definition of the segment Lipschitz constant gives

$$\|\delta_t\| \;=\; \|F_{p_t,\ell_t}(\mathbf{u}) - F_{p_t,\ell_t}(\mathbf{v})\| \;\le\; \mathrm{Lip}(F_{p_t,\ell_t}) \cdot \|\mathbf{u} - \mathbf{v}\| \;=\; \Lambda_t \, \|F_{p_t,\ell_t}(\mathbf{y}_{t-1}) - \mathbf{y}_{t-1}\|. \tag{26}$$

The coefficient is $m_t(2) = \Lambda_t$, and the bound is non-asymptotic.

Combining the three cases,

$$\|\delta_t\| \;\le\; m_t(\rho_t) \, \|F_{p_t,\ell_t}(\mathbf{y}_{t-1}) - \mathbf{y}_{t-1}\|, \tag{27}$$

with $m_t(\rho_t) \in \{1, 0, \Lambda_t\}$ for $\rho_t \in \{0, 1, 2\}$ as defined in Theorem 1.

Step 2: Hybrid-sequence telescope. For $0 \le j \le N$ define the hybrid sequence $\mathbf{H}^{(j)}$ by applying the routed operator $F^{\rho_t}_{p_t,\ell_t}$ for $t \le j$ and the full operator F_{p_t,ℓ_t} (i.e., $\rho_t = 1$) for $t > j$, all starting from $\mathbf{y}_0$ and followed by the protected layer f_L. In particular $\mathbf{H}^{(0)} = \mathbf{h}_L^{\text{full}}$, $\mathbf{H}^{(N)} = \mathbf{h}_L^{\text{routed}}$, and consecutive hybrids differ only at step j:

$$\mathbf{H}^{(j)} - \mathbf{H}^{(j-1)} \;=\; \Phi_{>j}\Big(F^{\rho_j}_{p_j,\ell_j}(\mathbf{y}_{j-1})\Big) \;-\; \Phi_{>j}\big(F_{p_j,\ell_j}(\mathbf{y}_{j-1})\big)\,, \tag{28}$$

where $\Phi_{>j} := f_L \circ F_{p_N,\ell_N} \circ \cdots \circ F_{p_{j+1},\ell_{j+1}}$ is the composition of the segments *after* step j followed by the protected last layer (so $\Phi_{>N} = f_L$).

By submultiplicativity of Lipschitz constants under composition,

$$\mathrm{Lip}(\Phi_{>j}) \;\le\; (1+\eta_L) \prod_{t'=j+1}^{N} \mathrm{Lip}(F_{p_{t'},\ell_{t'}}) \;=\; (1+\eta_L) \prod_{t'=j+1}^{N} \Lambda_{t'} \;=:\; A_j. \tag{29}$$

Applying this to Eq. (28) and using the per-step bound Eq. (27) gives

$$\|\mathbf{H}^{(j)} - \mathbf{H}^{(j-1)}\| \;\le\; A_j \, \|\delta_j\| \;\le\; A_j \, m_j(\rho_j) \, \|F_{p_j,\ell_j}(\mathbf{y}_{j-1}) - \mathbf{y}_{j-1}\|. \tag{30}$$

Step 3: Triangle inequality. Summing Eq. (30) over $j = 1, \ldots, N$ and applying the triangle inequality,

$$\begin{aligned}\|\mathbf{h}_L^{\text{routed}} - \mathbf{h}_L^{\text{full}}\| = \big\|\mathbf{H}^{(N)} - \mathbf{H}^{(0)}\big\| = \Big\|\sum_{j=1}^{N}(\mathbf{H}^{(j)} - \mathbf{H}^{(j-1)})\Big\| \;\le\; \sum_{j=1}^{N} \|\mathbf{H}^{(j)} - \mathbf{H}^{(j-1)}\| \\ \le\; \sum_{t=1}^{N} A_t \, m_t(\rho_t) \, \|F_{p_t,\ell_t}(\mathbf{y}_{t-1}) - \mathbf{y}_{t-1}\|, \end{aligned} \tag{31}$$

which is Eq. (3). The bound is *non-asymptotic*: each step uses the segment Lipschitz constant Λ_t directly rather than a first-order expansion, and the amplification $A_t = (1+\eta_L)\prod_{t'>t} \Lambda_{t'}$ is the submultiplicative bound on $\mathrm{Lip}(\Phi_{>t})$; the example below shows that it is attained. □

Tight example. The bound is attained: take translation blocks $f_i(\mathbf{h}) = \mathbf{h} + \mathbf{b}_i$ with all $\mathbf{b}_i$ positively collinear (so every η_i=0, Λ_t=1, A_t=1) and a trajectory that skips every routable segment ($\rho_t \equiv 0$). Both sides of Eq. (3) then equal $\sum_{i=2}^{L-1} \|\mathbf{b}_i\|$.

C.2 Proof of Proposition 2 (Pareto dominance)

Setup. For a routing rule $\phi : \mathbf{x} \mapsto (\ell_t(\mathbf{x}), \rho_t(\mathbf{x}))_t$ in a class $\mathcal{F}$, let $R(\phi) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[\mathcal{L}_{\mathrm{LM}}(\phi, \mathbf{x})]$ be the expected task loss and $\bar{C}(\phi) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[c(\phi, \mathbf{x})]$ the expected compute, where $c(\phi, \mathbf{x}) := \sum_t \ell_t(\mathbf{x})\rho_t(\mathbf{x})$ is the per-input operator cost ($C(\mathbf{x})$ of §3.1 up to the additive constant 2 for the protected layers, which shifts every budget equally and changes nothing below). We assume $\mathbb{E}_{\mathbf{x}}|\mathcal{L}_{\mathrm{LM}}(u, \mathbf{x})| < \infty$ for every trajectory u, and that each class is *decomposable*: with $\mathcal{F}(\mathbf{x})$ the finite set of trajectories admissible at $\mathbf{x}$, $\mathcal{F}$ contains every measurable selection $\mathbf{x} \mapsto u(\mathbf{x}) \in \mathcal{F}(\mathbf{x})$. (The static class is the exception and is treated separately at the end.) The optimal loss at compute budget B is $R^\star_{\mathcal{F}}(B) := \inf\{R(\phi) : \phi \in \mathcal{F},\ \bar{C}(\phi) \le B\}$, and the penalized optimum is

$$J^\star_{\mathcal{F}}(\lambda) \;:=\; \inf_{\phi\in\mathcal{F}} \big[R(\phi) + \lambda\, \bar{C}(\phi)\big], \qquad \lambda \ge 0. \tag{32}$$

Duality. For every $\lambda \ge 0$ and every $\phi \in \mathcal{F}$ with $\bar{C}(\phi) \le B$, $R(\phi) \ge R(\phi) + \lambda(\bar{C}(\phi) - B) \ge J^\star_{\mathcal{F}}(\lambda) - \lambda B$, so $R^\star_{\mathcal{F}}(B) \ge \sup_{\lambda\ge 0}[J^\star_{\mathcal{F}}(\lambda) - \lambda B]$ (weak duality). Only this inequality is used below: the non-strict constrained chain needs no duality at all, and the strict statement follows from weak duality alone. The penalized form is also the form of the compute penalty in Eq. (6), with $\lambda = \lambda_{\mathrm{compute}}$.

Per-input decomposition. For a decomposable class, $R(\phi)+\lambda\bar{C}(\phi) = \mathbb{E}_{\mathbf{x}}[\mathcal{L}_{\mathrm{LM}}(\phi, \mathbf{x}) + \lambda c(\phi, \mathbf{x})]$, and because $\mathcal{F}(\mathbf{x})$ is a finite set and each $\mathbf{x} \mapsto \mathcal{L}_{\mathrm{LM}}(u, \mathbf{x})$ is measurable, the pointwise argmin admits a measurable selection, so

$$J^\star_{\mathcal{F}}(\lambda) \;=\; \mathbb{E}_{\mathbf{x}}\big[J^\star_{\mathcal{F}}(\mathbf{x}, \lambda)\big], \qquad J^\star_{\mathcal{F}}(\mathbf{x}, \lambda) \;:=\; \min_{u\in\mathcal{F}(\mathbf{x})} \big[\mathcal{L}_{\mathrm{LM}}(u, \mathbf{x}) + \lambda\, c(u, \mathbf{x})\big],$$

and the infimum defining $J^\star_{\mathcal{F}}(\lambda)$ is attained.

Monotonicity under class inclusion. $\mathcal{F}_{\mathrm{static}} \subseteq \mathcal{F}_{\mathrm{bin}}$ replaces an input-independent skip mask by an input-conditional one over the same operator support $\{O_{1,0}, O_{1,1}\}$; $\mathcal{F}_{\mathrm{bin}} \subseteq \mathcal{F}_{\ell=1}$ adds $O_{1,2}$; and $\mathcal{F}_{\ell=1} \subseteq \mathcal{F}_{\mathcal{O}}$ adds the ℓ=2 operators. Since $\mathcal{F}_{\mathrm{bin}}(\mathbf{x}) \subseteq \mathcal{F}_{\ell=1}(\mathbf{x}) \subseteq \mathcal{F}_{\mathcal{O}}(\mathbf{x})$ for every $\mathbf{x}$, a minimum over a smaller set is no smaller:

$$J^\star_{\mathcal{O}}(\mathbf{x}, \lambda) \;\le\; J^\star_{\ell=1}(\mathbf{x}, \lambda) \;\le\; J^\star_{\mathrm{bin}}(\mathbf{x}, \lambda) \tag{33}$$

for every $(\mathbf{x}, \lambda)$, and taking $\mathbb{E}_{\mathbf{x}}$ gives the penalized chain; $J^\star_{\mathrm{bin}}(\lambda) \le J^\star_{\mathrm{static}}(\lambda)$ holds because $\mathcal{F}_{\mathrm{static}} \subseteq \mathcal{F}_{\mathrm{bin}}$. The constrained chain $R^\star_{\mathcal{O}}(B) \le R^\star_{\ell=1}(B) \le R^\star_{\mathrm{bin}}(B) \le R^\star_{\mathrm{static}}(B)$ holds at every B for the simplest reason: the budget-feasible sets are nested, $\{\phi \in \mathcal{F}' : \bar{C}(\phi) \le B\} \subseteq \{\phi \in \mathcal{F} : \bar{C}(\phi) \le B\}$ whenever $\mathcal{F}' \subseteq \mathcal{F}$.

Strict inequality via positive-probability separation. Consider $\mathcal{F}_{\ell=1} \subseteq \mathcal{F}_{\mathcal{O}}$. The hypothesis must exclude ties: if some ℓ=1 trajectory *also* attains the $\mathcal{O}$-optimum at $\mathbf{x}$, nothing is gained there. We therefore assume

$$q \;:=\; \Pr_{\mathbf{x}\sim\mathcal{D}}\big[\,\text{no } \ell{=}1 \text{ trajectory attains } J^\star_{\mathcal{O}}(\mathbf{x}, \lambda)\,\big] \;>\; 0, \tag{34}$$

i.e. on a set S of positive probability *every* $\mathcal{O}$-minimizer uses an ℓ=2 operator. (The weaker reading "some minimizer uses ℓ=2" is not sufficient: it is compatible with an ℓ=1 trajectory tying for the optimum at every such $\mathbf{x}$.) For $\mathbf{x} \in S$ the per-input gap

$$\mathrm{gap}(\mathbf{x}, \lambda) \;:=\; J^\star_{\ell=1}(\mathbf{x}, \lambda) - J^\star_{\mathcal{O}}(\mathbf{x}, \lambda)$$

is strictly positive, because both are minima over finite sets and no minimizer of the second lies in the first. By continuity of measure there is $\varepsilon > 0$ with $\Pr(\{\mathbf{x} \in S : \mathrm{gap}(\mathbf{x}, \lambda) \ge \varepsilon\}) \ge q/2$, so

$$J^\star_{\ell=1}(\lambda) - J^\star_{\mathcal{O}}(\lambda) \;=\; \mathbb{E}_{\mathbf{x}}[\mathrm{gap}(\mathbf{x}, \lambda)] \;\ge\; \varepsilon\, q/2 \;>\; 0.$$

The constrained frontier inherits the strict gap by weak duality alone. Let $\phi^\star_\lambda$ attain $J^\star_{\mathcal{O}}(\lambda)$ and set $B := \bar{C}(\phi^\star_\lambda)$. Then

$$R^\star_{\ell=1}(B) \;\geq\; J^\star_{\ell=1}(\lambda) - \lambda B \;\geq\; J^\star_{\mathcal{O}}(\lambda) + \varepsilon q/2 - \lambda B \;=\; R(\phi^\star_\lambda) + \varepsilon q/2 \;\geq\; R^\star_{\mathcal{O}}(B) + \varepsilon q/2,$$

so the richer class is strictly better at that compute level, with no atomlessness assumption.

Which segment operators can witness strictness. If skips emit the identity, $O_{2,0} = O_{1,0} \circ O_{1,0}$ and $O_{2,1} = O_{1,1} \circ O_{1,1}$ as maps, at equal cost. Every trajectory that avoids $O_{2,2}$ therefore has an ℓ=1 twin with identical $\mathcal{L}_{\mathrm{LM}} + \lambda c$, so $J^\star_{\mathcal{O}\setminus\{O_{2,2}\}}(\mathbf{x}, \lambda) = J^\star_{\ell=1}(\mathbf{x}, \lambda)$ for every $(\mathbf{x}, \lambda)$, and Eq. (34) can hold only where every $\mathcal{O}$-minimizer uses $O_{2,2}$.

The same argument gives strict gaps for the remaining input-conditional inclusions, with the correspondingly strengthened hypotheses: $J^\star_{\ell=1}(\lambda) < J^\star_{\mathrm{bin}}(\lambda)$ whenever $\Pr_{\mathbf{x}}\big[\text{no trajectory in } \mathcal{F}_{\mathrm{bin}}(\mathbf{x}) \text{ attains } J^\star_{\ell=1}(\mathbf{x}, \lambda)\big] > 0$; and $J^\star_{\mathcal{O}}(\lambda) < J^\star_{\rho\in\{0,1\}}(\lambda)$ whenever $\Pr_{\mathbf{x}}\big[\text{no trajectory in } \mathcal{F}_{\rho\in\{0,1\}}(\mathbf{x}) \text{ attains } J^\star_{\mathcal{O}}(\mathbf{x}, \lambda)\big] > 0$. In both cases the weaker condition "some minimizer uses ρ=2" is *not* sufficient: an idempotent block ($f \circ f = f$) at λ=0 makes $O_{1,2}$ and $O_{1,1}$ tie on every input.

The gap $J^\star_{\mathrm{bin}}(\lambda) < J^\star_{\mathrm{static}}(\lambda)$ is not of this form, because $\mathcal{F}_{\mathrm{static}}$ is not decomposable: the rule must be the same at every input. Here $J^\star_{\mathrm{static}}(\lambda) = \min_m \mathbb{E}_{\mathbf{x}}[\mathcal{L}_{\mathrm{LM}}(m, \mathbf{x}) + \lambda c(m, \mathbf{x})]$, a minimum over fixed skip masks m, while $J^\star_{\mathrm{bin}}(\lambda) = \mathbb{E}_{\mathbf{x}}[\min_m(\cdot)]$. The inequality $\mathbb{E}[\min] \leq \min \mathbb{E}$ is immediate, and it is strict exactly when no single mask attains the pointwise minimum $\mathcal{D}$-almost everywhere – the precise form of "the optimal skip pattern varies across inputs". □

C.3 Proof of Proposition 3 (strict expressive gap)

Setup. Let $f_p(\mathbf{h}) = \mathbf{h} + \mathbf{P}\mathbf{h}$ and $f_{p+1}(\mathbf{h}) = \mathbf{h} + \mathbf{Q}\mathbf{h}$ be linear residual blocks with identity skips, $J_p := I + \mathbf{P}$ and $J_{p+1} := I + \mathbf{Q}$ both invertible, $|\det J_p|$ and $|\det J_{p+1}|$ both strictly above or both strictly below 1, and $[\mathbf{P}, \mathbf{Q}] = [J_p, J_{p+1}] \neq 0$. We show the segment-repeat operator $O_{2,2} = (f_{p+1} \circ f_p)^2$ is not an element of the ℓ=1 function class

$$\mathcal{M}_{\ell=1}(f_p, f_{p+1}) \;=\; \big\{\, f_{p+1}^{\rho_{p+1}} \circ f_p^{\rho_p} : \rho_p, \rho_{p+1} \in \{0, 1, 2\} \,\big\}, \tag{35}$$

the finite family of nine maps that apply f_p before f_{p+1} without interleaving (using $f^0 = \mathrm{id}$).

Step 1: Jacobian of a composition. For any composition $g = h_k \circ \cdots \circ h_1$ of smooth maps, the chain rule gives, at any point $\mathbf{h}$,

$$\nabla g(\mathbf{h}) \;=\; \nabla h_k(h_{k-1} \circ \cdots \circ h_1(\mathbf{h})) \cdots \nabla h_2(h_1(\mathbf{h}))\, \nabla h_1(\mathbf{h}). \tag{36}$$

For *linear* blocks every chain-rule factor is state-independent, so the products collapse and equality of the maps is equivalent to equality of these matrix products. (This is precisely why we state the proposition for linear blocks: for nonlinear f, the factors in Eq. (36) are evaluated at distinct intermediate states, and a commutator at a single reference point constrains nothing on its own. The nonlinear case is treated separately in Corollary 5 below.) Hence, at every $\mathbf{h}$,

$$\nabla O_{2,2}(\mathbf{h}) \;=\; J_{p+1}\, J_p\, J_{p+1}\, J_p, \tag{37}$$

and the Jacobian of any ℓ=1 element $g = f_{p+1}^{\rho_{p+1}} \circ f_p^{\rho_p}$ is

$$\nabla g(\mathbf{h}) \;=\; J_{p+1}^{\rho_{p+1}}\, J_p^{\rho_p}. \tag{38}$$

Step 2: Commutator obstruction. Setting Eqs. (37) and (38) equal gives

$$J_{p+1}\, J_p\, J_{p+1}\, J_p \;=\; J_{p+1}^{\rho_{p+1}}\, J_p^{\rho_p} \qquad \text{for some } (\rho_p, \rho_{p+1}) \in \{0, 1, 2\}^2. \tag{39}$$

We show this equation forces $[J_p, J_{p+1}] = 0$. Rewrite the left-hand side by moving all J_p factors to the right, accumulating commutator corrections:

$$J_{p+1}\, J_p\, J_{p+1}\, J_p = J_{p+1}^2\, J_p^2 \;+\; J_{p+1}\, [J_p, J_{p+1}]\, J_p. \tag{40}$$

The identity Eq. (40) follows from $J_p\, J_{p+1} = J_{p+1}\, J_p + [J_p, J_{p+1}]$ applied once to the middle pair $J_p\, J_{p+1}$. Substituting into Eq. (39),

$$J_{p+1}^2\, J_p^2 + J_{p+1}\, [J_p, J_{p+1}]\, J_p \;=\; J_{p+1}^{\rho_{p+1}}\, J_p^{\rho_p}. \tag{41}$$

Case analysis on $(\rho_p, \rho_{p+1}) \in \{0,1,2\}^2$ proceeds in two steps. First, taking determinants of both sides of Eq. (39) gives

$$\det(J_{p+1})^2 \det(J_p)^2 \;=\; \det(J_{p+1})^{\rho_{p+1}} \det(J_p)^{\rho_p}.$$

Writing $a := 2 - \rho_p$ and $b := 2 - \rho_{p+1}$, both in $\{0,1,2\}$, this reads $|\det J_p|^a \, |\det J_{p+1}|^b = 1$. If $|\det J_p| > 1$ and $|\det J_{p+1}| > 1$ then the left-hand side is ≥ 1 with equality iff $a = b = 0$; if both are < 1 it is ≤ 1 with equality iff $a = b = 0$. Either way $\rho_p = \rho_{p+1} = 2$, ruling out the eight cases $(\rho_p, \rho_{p+1}) \neq (2,2)$. This is the only place the determinant hypothesis is used. Second, in the remaining case $(\rho_p, \rho_{p+1})=(2,2)$, Eq. (39) reduces via Eq. (40) to $J_{p+1}\,[J_p, J_{p+1}]\,J_p = 0$, which forces $[J_p, J_{p+1}] = 0$ since J_p, J_{p+1} are invertible.

Therefore, under the assumption $[J_p, J_{p+1}] \neq 0$, Eq. (39) has no solution and $O_{2,2} \notin \mathcal{M}_{\ell=1}(f_p, f_{p+1})$ as a map on all of $\mathbb{R}^d$. □

The determinant hypothesis is necessary. Take any invertible J_p and any involution $R \neq \pm I$ (say $R = \mathrm{diag}(1,-1,1)$), and set $J_{p+1} := R\,J_p^{-1}$. Then $O_{2,2} = (J_{p+1}J_p)^2 = R^2 = I$, which is the $\ell{=}1$ element with $(\rho_p, \rho_{p+1}) = (0,0)$, even though $[J_p, J_{p+1}] \neq 0$ in general. Such a pair necessarily has $|\det J_p|\,|\det J_{p+1}| = 1$, i.e. $|\det J_p|$ and $|\det J_{p+1}|$ sit on opposite sides of 1 or both equal 1 – exactly the configurations the hypothesis excludes.

The nonlinear case. For nonlinear blocks the four chain-rule factors of $\nabla O_{2,2}(\mathbf{h}^\star)$ are evaluated at four *different* states $\mathbf{h}^\star$, $f_p(\mathbf{h}^\star)$, $f_{p+1}f_p(\mathbf{h}^\star)$, $f_pf_{p+1}f_p(\mathbf{h}^\star)$, and the argument extends to a local statement with an explicit error budget.

Corollary 5 (Local expressive separation, per-layer repeat case)**.** *Let f_p, f_{p+1} be C^1 with $\|f_i(\mathbf{h}) - \mathbf{h}\| \leq \Delta$ and $\|\nabla f_i(\mathbf{h}) - \nabla f_i(\mathbf{h}')\| \leq M\|\mathbf{h} - \mathbf{h}'\|$ on the ball $\mathcal{B} := \{\mathbf{h} : \|\mathbf{h} - \mathbf{h}^\star\| \leq 3\Delta\}$, $i \in \{p, p+1\}$. Write $J_p := \nabla f_p(\mathbf{h}^\star)$, $J_{p+1} := \nabla f_{p+1}(\mathbf{h}^\star)$ and $G := \max\{1, \sup_{\mathbf{h}\in\mathcal{B}} \|\nabla f_p(\mathbf{h})\|, \sup_{\mathbf{h}\in\mathcal{B}} \|\nabla f_{p+1}(\mathbf{h})\|\}$ – a supremum over the neighbourhood in which the chain-rule factors are evaluated. Then for $g \in \{O_{2,2},\, f_{p+1}^2 \circ f_p^2\}$ with idealized products $\hat{J}_g \in \{J_{p+1}J_pJ_{p+1}J_p,\; J_{p+1}^2J_p^2\}$,*

$$\big\|\nabla g(\mathbf{h}^\star) - \hat{J}_g\big\| \;\leq\; 12\,G^3 M\Delta \;=:\; \varepsilon_{\mathrm{lin}}. \tag{42}$$

Consequently, if

$$\big\|J_{p+1}\,[J_p, J_{p+1}]\,J_p\big\| \;>\; 2\,\varepsilon_{\mathrm{lin}}, \tag{43}$$

then $\nabla O_{2,2}(\mathbf{h}^\star) \neq \nabla(f_{p+1}^2 \circ f_p^2)(\mathbf{h}^\star)$; in particular $O_{2,2}$ and the per-layer repeat trajectory $O_{1,2} \circ O_{1,2}$ do not coincide on any neighbourhood of $\mathbf{h}^\star$.

Proof. $O_{2,2}$ applies four blocks, so its Jacobian at $\mathbf{h}^\star$ is a product of four factors evaluated at the states $\mathbf{h}^\star$, $f_p(\mathbf{h}^\star)$, $f_{p+1}f_p(\mathbf{h}^\star)$, $f_pf_{p+1}f_p(\mathbf{h}^\star)$. By the residual bound these lie within $0, \Delta, 2\Delta$ and 3Δ of $\mathbf{h}^\star$ respectively (all inside $\mathcal{B}$), so each factor differs from its value at $\mathbf{h}^\star$ by at most $3M\Delta$ in norm. For products of four matrices with all factors bounded by G, $\|\prod_i A_i - \prod_i B_i\| \leq \sum_{i=1}^4 \|A_1 \cdots A_{i-1}\|\,\|A_i - B_i\|\,\|B_{i+1} \cdots B_4\| \leq 4G^3 \max_i \|A_i - B_i\|$, which gives $4G^3 \cdot 3M\Delta = 12G^3M\Delta$. The per-layer repeat map $f_{p+1}^2 \circ f_p^2$ also applies four blocks at states within 3Δ of $\mathbf{h}^\star$ and is bounded identically. The two idealized products differ by exactly $J_{p+1}[J_p, J_{p+1}]J_p$ (Eq. (40)), so by the triangle inequality $\|\nabla O_{2,2}(\mathbf{h}^\star) - \nabla(f_{p+1}^2 \circ f_p^2)(\mathbf{h}^\star)\| \geq \|J_{p+1}[J_p, J_{p+1}]J_p\| - 2\varepsilon_{\mathrm{lin}} > 0$. Two C^1 maps that coincide on a neighbourhood of $\mathbf{h}^\star$ have equal Jacobians there, which proves the last claim of the corollary. □

The remaining eight cases $(\rho_p, \rho_{p+1}) \neq (2,2)$ are excluded in the linear proposition by the determinant identity $|\det J_p|^a |\det J_{p+1}|^b = 1$; the corollary concerns the per-layer repeat trajectory, which is the case closest to $O_{2,2}$ in cost and structure.

C.4 Proof of Proposition 4 (joint vs. factorized head gap)

Setup. Let $\mathcal{S}_\ell = \{1,2\}$ and $\mathcal{S}_\rho = \{0,1,2\}$, so $\mathcal{O} = \mathcal{S}_\ell \times \mathcal{S}_\rho$ with $|\mathcal{O}| = |\mathcal{S}_\ell| \cdot |\mathcal{S}_\rho| = 6$. Let $\Delta^{|\mathcal{O}|-1}$ denote the probability simplex on $\mathcal{O}$. The *independence manifold* is

$$\mathcal{I} \;=\; \big\{\pi_\ell \otimes \pi_\rho : \pi_\ell \in \Delta^{|\mathcal{S}_\ell|-1},\, \pi_\rho \in \Delta^{|\mathcal{S}_\rho|-1}\big\} \;\subset\; \Delta^{|\mathcal{O}|-1}, \tag{44}$$

where $(\pi_\ell \otimes \pi_\rho)(\ell, \rho) = \pi_\ell(\ell)\,\pi_\rho(\rho)$. This set has dimension $(|\mathcal{S}_\ell| - 1) + (|\mathcal{S}_\rho| - 1) = 3$, whereas the ambient simplex has dimension $|\mathcal{O}| - 1 = 5$; the *approximation gap* of the factorized head is the distance from $\boldsymbol{\pi}^\star$ to the manifold $\mathcal{I}$ in the KL sense.

Given any target $\boldsymbol{\pi}^\star \in \Delta^{|\mathcal{O}|-1}$, the joint head approximates $\boldsymbol{\pi}^\star$ arbitrarily well (exactly, if $\boldsymbol{\pi}^\star$ has full support; zero entries are reached only in the limit of unbounded logits, except for masked operators, whose logits are exactly $-\infty$), while the factorized head is restricted to $\mathcal{I}$. Define the *marginal distributions*

$$\pi^\star_\ell(\ell) \;:=\; \sum_\rho \boldsymbol{\pi}^\star(\ell, \rho), \qquad \pi^\star_\rho(\rho) \;:=\; \sum_\ell \boldsymbol{\pi}^\star(\ell, \rho), \tag{45}$$

and the mutual information between ℓ and ρ under $\boldsymbol{\pi}^\star = \boldsymbol{\pi}^\star(\cdot \mid s_t)$,

$$I_{\boldsymbol{\pi}^\star}(\ell; \rho) \;:=\; \sum_{\ell,\rho} \boldsymbol{\pi}^\star(\ell, \rho) \log \frac{\boldsymbol{\pi}^\star(\ell, \rho)}{\pi^\star_\ell(\ell)\pi^\star_\rho(\rho)} \;=\; \mathrm{KL}(\boldsymbol{\pi}^\star \| \pi^\star_\ell \otimes \pi^\star_\rho). \tag{46}$$

The second equality is by definition of KL. We must show that this quantity is the minimum of $\mathrm{KL}(\boldsymbol{\pi}^\star \| \cdot)$ over $\mathcal{I}$.

Step 1: KL decomposition onto the independence manifold. For any $q = \pi_\ell \otimes \pi_\rho \in \mathcal{I}$, expand

$$\begin{aligned}
\mathrm{KL}(\boldsymbol{\pi}^\star \| q) &= \sum_{\ell,\rho} \boldsymbol{\pi}^\star(\ell, \rho) \log \frac{\boldsymbol{\pi}^\star(\ell, \rho)}{\pi_\ell(\ell)\pi_\rho(\rho)} \\
&= \sum_{\ell,\rho} \boldsymbol{\pi}^\star(\ell, \rho) \log \frac{\boldsymbol{\pi}^\star(\ell, \rho)}{\pi^\star_\ell(\ell)\pi^\star_\rho(\rho)} \;+\; \sum_{\ell,\rho} \boldsymbol{\pi}^\star(\ell, \rho) \log \frac{\pi^\star_\ell(\ell)\pi^\star_\rho(\rho)}{\pi_\ell(\ell)\pi_\rho(\rho)} \\
&= I_{\boldsymbol{\pi}^\star}(\ell; \rho) \;+\; \underbrace{\sum_\ell \pi^\star_\ell(\ell) \log \frac{\pi^\star_\ell(\ell)}{\pi_\ell(\ell)}}_{=\mathrm{KL}(\pi^\star_\ell \| \pi_\ell)} \;+\; \underbrace{\sum_\rho \pi^\star_\rho(\rho) \log \frac{\pi^\star_\rho(\rho)}{\pi_\rho(\rho)}}_{=\mathrm{KL}(\pi^\star_\rho \| \pi_\rho)},
\end{aligned} \tag{47}$$

where the last equality uses $\sum_\rho \boldsymbol{\pi}^\star(\ell, \rho) = \pi^\star_\ell(\ell)$ and $\sum_\ell \boldsymbol{\pi}^\star(\ell, \rho) = \pi^\star_\rho(\rho)$.

Step 2: Pythagorean minimum. All three terms in Eq. (47) are non-negative (KL is non-negative), so

$$\mathrm{KL}(\boldsymbol{\pi}^\star \| q) \;\geq\; I_{\boldsymbol{\pi}^\star}(\ell; \rho) \qquad \text{for every } q \in \mathcal{I}. \tag{48}$$

The right-hand side is independent of q, so equality is achieved iff $\mathrm{KL}(\pi^\star_\ell \| \pi_\ell) = 0$ and $\mathrm{KL}(\pi^\star_\rho \| \pi_\rho) = 0$, i.e. iff $\pi_\ell = \pi^\star_\ell$ and $\pi_\rho = \pi^\star_\rho$. Therefore

$$\min_{q \in \mathcal{I}} \mathrm{KL}(\boldsymbol{\pi}^\star \| q) \;=\; \mathrm{KL}(\boldsymbol{\pi}^\star \| \pi^\star_\ell \otimes \pi^\star_\rho) \;=\; I_{\boldsymbol{\pi}^\star}(\ell; \rho), \tag{49}$$

which is the Pythagorean identity in Eq. (11).

Step 3: Consequence. The joint head's infimal error is 0. The factorized head is restricted to $\mathcal{I}$ and attains the minimum above, so its error equals $I_{\boldsymbol{\pi}^\star}(\ell; \rho)$, which is at most $\min\{H(\ell), H(\rho)\} \leq \log |\mathcal{S}_\ell| = \log 2$. The two heads match iff $\boldsymbol{\pi}^\star$ is a product distribution; in particular every point mass $\delta_{(\ell_0, \rho_0)} = \delta_{\ell_0} \otimes \delta_{\rho_0}$ is a product, so for deterministic targets – including the per-state minimizer of the penalized objective, an argmin over a finite set – the gap is exactly zero. The advantage of the joint head therefore lies in training, where the soft distribution over operators need not be a product, and the focal-balance regularizer keeps every operator explored. □

C.5 Proof of Lemma 6 (soft-to-hard discrepancy)

We first restate Lemma 6 formally (referenced from §3.4). Define the *training-time soft-mixture next hidden state*

$$\tilde{\mathbf{h}}_{t+1} \;:=\; \sum_{(\ell,\rho) \in \mathcal{O}_{\text{valid}}(p_t)} \boldsymbol{\pi}_t(\ell, \rho)\, F^\rho_{p_t, \ell}(\mathbf{h}_t). \tag{50}$$

Only the hidden state is mixed: the pointer is an integer that the algorithm advances by a straight-through argmax (Eq. (13)), not by a convex combination. Let $\mathbf{h}^\star_{t+1} := F^{\rho^\star}_{p_t, \ell^\star}(\mathbf{h}_t)$ denote the inference-time hard-argmax next hidden state.

Lemma 6 (Soft-to-hard discrepancy)**.** *Let $(\ell^\star, \rho^\star)$ be the argmax operator at state s_t and let $M_t := \max_{(\ell,\rho)\in\mathcal{O}_{valid}(p_t)\setminus\{(\ell^\star,\rho^\star)\}} \|F^{\rho}_{p_t,\ell}(\mathbf{h}_t) - F^{\rho^\star}_{p_t,\ell^\star}(\mathbf{h}_t)\|$ be the maximum operator disagreement. Then*

$$\big\|\tilde{\mathbf{h}}_{t+1} - \mathbf{h}^\star_{t+1}\big\| \;\le\; M_t\big(1 - \boldsymbol{\pi}_t(\ell^\star,\rho^\star)\big) \;\le\; M_t\big(1 - e^{-H(\boldsymbol{\pi}_t)}\big), \tag{51}$$

where $H(\boldsymbol{\pi}_t)$ is the Shannon entropy of $\boldsymbol{\pi}_t$. If the maximal logit is unique, then as $\tau \to 0$ the entropy vanishes and the bound tends to zero.

The lemma is a single-step statement at a fixed routing state s_t, which is how it is used in §3.4: it shows that the soft update of Eq. (7) approaches the hard one as the distribution sharpens. The second inequality is the ordering of Rényi entropies, $-\log\max_o \boldsymbol{\pi}(o) = H_\infty(\boldsymbol{\pi}) \le H(\boldsymbol{\pi})$.

The proof is a convex-combination bound followed by a monotonicity argument on $\log$.

Step 1: Convex-combination bound. Let $\boldsymbol{\pi}_t \in \Delta^{|\mathcal{O}|-1}$ be the routing distribution at step t and let $(\ell^\star, \rho^\star)$ be the argmax operator. Let $\mathbf{o}_{\ell,\rho} := F^{\rho}_{p_t,\ell}(\mathbf{h}_t)$ denote the operator outputs and $\mathbf{o}^\star := \mathbf{o}_{\ell^\star,\rho^\star}$ the hard selection. The soft mixture is $\tilde{\mathbf{h}}_{t+1} = \sum_{(\ell,\rho)} \boldsymbol{\pi}_t(\ell,\rho)\,\mathbf{o}_{\ell,\rho}$, so

$$\begin{aligned}\tilde{\mathbf{h}}_{t+1} - \mathbf{o}^\star &= \sum_{(\ell,\rho)} \boldsymbol{\pi}_t(\ell,\rho)\,\mathbf{o}_{\ell,\rho} - \mathbf{o}^\star \\ &= \sum_{(\ell,\rho)\neq(\ell^\star,\rho^\star)} \boldsymbol{\pi}_t(\ell,\rho)\,(\mathbf{o}_{\ell,\rho} - \mathbf{o}^\star),\end{aligned} \tag{52}$$

where we used the normalization $\sum_{(\ell,\rho)} \boldsymbol{\pi}_t(\ell,\rho) = 1$ to replace $\mathbf{o}^\star$ with the sum $\sum_{(\ell,\rho)} \boldsymbol{\pi}_t(\ell,\rho)\,\mathbf{o}^\star$ (so the $(\ell^\star,\rho^\star)$ term cancels).

Applying the triangle inequality and the definition $M_t := \max_{(\ell,\rho)\neq(\ell^\star,\rho^\star)} \|\mathbf{o}_{\ell,\rho} - \mathbf{o}^\star\|$,

$$\begin{aligned}\|\tilde{\mathbf{h}}_{t+1} - \mathbf{o}^\star\| &\le \sum_{(\ell,\rho)\neq(\ell^\star,\rho^\star)} \boldsymbol{\pi}_t(\ell,\rho)\,\|\mathbf{o}_{\ell,\rho} - \mathbf{o}^\star\| \\ &\le M_t \sum_{(\ell,\rho)\neq(\ell^\star,\rho^\star)} \boldsymbol{\pi}_t(\ell,\rho) \;=\; M_t\,\big(1 - \boldsymbol{\pi}_t(\ell^\star,\rho^\star)\big).\end{aligned} \tag{53}$$

This is the first inequality in Eq. (51).

Step 2: Entropy lower bound on the maximum. Let $p^\star := \boldsymbol{\pi}_t(\ell^\star,\rho^\star)$ be the largest component of $\boldsymbol{\pi}_t$, and let $H(\boldsymbol{\pi}_t) = -\sum_{(\ell,\rho)} \boldsymbol{\pi}_t(\ell,\rho)\log\boldsymbol{\pi}_t(\ell,\rho)$ be its Shannon entropy. For any probability distribution $p_1,\dots,p_n$ with $p^\star = \max_k p_k$,

$$H(\boldsymbol{\pi}_t) \;=\; -\sum_k p_k \log p_k \;\ge\; -\sum_k p_k \log p^\star \;=\; -\log p^\star, \tag{54}$$

where the inequality uses the monotonicity of $-\log$: since $p_k \le p^\star$, we have $-\log p_k \ge -\log p^\star$, and the weighted sum (with weights $p_k \ge 0$) preserves the direction. Rearranging, $-H(\boldsymbol{\pi}_t) \le \log p^\star$, hence $p^\star \ge e^{-H(\boldsymbol{\pi}_t)}$ and $1 - p^\star \le 1 - e^{-H(\boldsymbol{\pi}_t)}$. Substituting into Eq. (53) gives the second inequality in Eq. (51):

$$\|\tilde{\mathbf{h}}_{t+1} - \mathbf{o}^\star\| \;\le\; M_t\,(1 - p^\star) \;\le\; M_t\big(1 - e^{-H(\boldsymbol{\pi}_t)}\big). \tag{55}$$

Step 3: Temperature and uniform-distribution limits. If the maximal logit is unique, then as $\tau \to 0$ the (temperature-annealed) softmax concentrates on the argmax and $H(\boldsymbol{\pi}_t) \to 0$, so the right-hand side vanishes; with k tied maximal logits the entropy tends to $\log k$ instead. Conversely, the entropy of any distribution on $|\mathcal{O}|$ outcomes is bounded by $\log|\mathcal{O}|$, with equality only at the uniform distribution. In that worst case $p^\star = 1/|\mathcal{O}|$ and the bound reduces to $\|\tilde{\mathbf{h}}_{t+1} - \mathbf{o}^\star\| \le M_t(1 - 1/|\mathcal{O}|)$. □

C.6 Unique minimizer of the focal-balance regularizer

Proposition 7. *For every $\gamma \ge 0$, the map $\bar{\boldsymbol{\pi}} \mapsto -\frac{1}{|\mathcal{O}|}\sum_{o\in\mathcal{O}}(1-\bar{\pi}_o)^\gamma \log\bar{\pi}_o$ on the interior of the simplex $\Delta^{|\mathcal{O}|-1}$ has the uniform distribution as its unique minimizer.*

Proof. Write $\psi(u) = -(1-u)^\gamma \log u$ on $(0,1)$, so the regularizer is $\frac{1}{|\mathcal{O}|}\sum_o \psi(\bar\pi_o)$. Differentiating,

$$\psi'(u) = \gamma(1-u)^{\gamma-1}\log u - \frac{(1-u)^\gamma}{u},$$
$$\psi''(u) = -\gamma(\gamma-1)(1-u)^{\gamma-2}\log u \;+\; \frac{2\gamma(1-u)^{\gamma-1}}{u} \;+\; \frac{(1-u)^\gamma}{u^2}.$$

On $(0,1)$ we have $\log u < 0$ and $(1-u) > 0$, so for every $\gamma \geq 1$ all three terms of ψ'' are non-negative and the last is strictly positive; hence $\psi'' > 0$ and ψ is strictly convex on $(0,1)$. (At γ=2, the value used throughout, this reduces to $\psi''(u) = -2\log u + 2/u + 1/u^2 - 3 > 0$. At γ=0, $\psi = -\log u$ is strictly convex outright.) For $\gamma \in (0,1)$ the first term of ψ'' is negative, but writing $\psi''(u) = (1-u)^{\gamma-2}u^{-2}\big[\gamma(\gamma-1)(-u^2\log u) + 2\gamma u(1-u) + (1-u)^2\big]$ and using $-\log u \leq (1-u)/u$ gives $\psi''(u) \geq (1-u)^{\gamma-2}u^{-2}\big[\gamma(1+\gamma)u(1-u) + (1-u)^2\big] > 0$, so ψ is strictly convex for every $\gamma \geq 0$. The regularizer is a symmetric separable sum of a strictly convex function, so Jensen gives, for any $\bar{\boldsymbol{\pi}} \in \Delta^{|\mathcal{O}|-1}$,

$$\tfrac{1}{|\mathcal{O}|}\sum_o \psi(\bar\pi_o) \;\geq\; \psi\Big(\tfrac{1}{|\mathcal{O}|}\sum_o \bar\pi_o\Big) \;=\; \psi\big(1/|\mathcal{O}|\big),$$

using $\sum_o \bar\pi_o = 1$, with equality iff all $\bar\pi_o$ coincide – that is, iff $\bar{\boldsymbol{\pi}}$ is uniform. □

Remark. The uniform minimizer acts as a training-time pressure. In the total objective (6) it competes with $\mathcal{L}_{\text{compute}}$, which prices each operator at $\ell\rho$, so the stationary mix trades balance against cost. The regularizer keeps every operator receiving gradient long enough to be *evaluated*, and the final policy retains the operators that serve the objective.

D Experimental Protocol and Baselines

D.1 Full experimental protocol

Backbones. The main comparison (Table 1) uses five frozen pretrained backbones: LLaMA-2-7B (Touvron et al., 2023) (L=32), LLaMA-3.2-3B (L=28), LLaMA-3-8B-Instruct (Grattafiori et al., 2024) (L=32), Qwen2.5-7B (Qwen Team, 2025) (L=28) and Mistral-7B (Jiang et al., 2023) (L=32). Appendix E additionally reports LLaMA-3.1-8B-Instruct (L=32) and Qwen2.5-7B-Instruct (L=28) for the comparison with Dr.LLM. Every backbone is pinned to a single public revision, and each job runs in `bfloat16` on one NVIDIA RTX PRO 6000 Blackwell GPU (96 GB).

Operator instantiation. Operators act on whole transformer layers. The cost of MoSO's additional trained components (Appendix B.3.1; 0.2–0.8% of dense compute) is included in every reported compute figure. The router has 0.93M parameters on LLaMA-3.2-3B and 1.09M on the 32-layer backbones, at most 0.03% of the backbone parameters.

Training data. Base backbones are trained on a pool of 7,466 examples: all 3,370 items of the ARC-Easy and ARC-Challenge *train* splits (Clark et al., 2018) and 4,096 from DART-Math (Tong et al., 2024); evaluation uses the ARC-Challenge *test* split, which is disjoint from the training items. The instruction-tuned backbones are trained on 50K Tulu-3 examples (Lambert et al., 2025). No other data are used for router training; the BI prior is calibrated on 32 WikiText-2 chunks (Appendix B.3.1).

Evaluation. All accuracies are zero-shot under `lm-evaluation-harness` v0.4.4 on the full standard evaluation splits of ARC-Challenge, HellaSwag (Zellers et al., 2019), MMLU (Hendrycks et al., 2021), PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2021) and BoolQ (Clark et al., 2019). We report per-task accuracy, the unweighted mean Avg_6 and $\%\text{Dense} := \text{Avg}_6/\text{Avg}_6^{\text{dense}}$. MoSO rows of Table 1 are means over three seeds, and the standard deviations of MoSO and Router-Tuning are summarized in Appendix E.1.

Compute accounting. Compute is the FLOP ratio against the dense model, computed from each backbone's configuration. For methods that skip whole layers it equals the fraction of executed layers $C(\mathbf{x})/L$, up to the small cost of MoSO's additional trained components, which is counted;

$L_{\text{eff}} = \mathbb{E}_{\mathbf{x}}[C(\mathbf{x})]$ denotes the mean number of executed layers, including the two protected layers. Compute is always *measured*: for input-dependent methods it is averaged over the evaluation inputs, and for Dr.LLM it is read from its own per-layer decisions (0=skip, 1=execute, 2=repeat). BlockPruner removes attention and MLP sub-blocks as separate units, so a budget of K removed units leaves it 6–9 points of dense compute above the column; we therefore remove units until its realized FLOP ratio approximately matches the column (matched FLOPs).

Static baselines. ShortGPT (Men et al., 2025) removes the layers with the lowest Block Influence, SLEB (Song et al., 2024) removes blocks greedily by calibration perplexity, LaCo (Yang et al., 2024) merges adjacent layers, and BlockPruner (Zhong et al., 2025) searches over sub-block removals by perplexity; each is calibrated per budget with its published procedure.

Router-Tuning. Router-Tuning (He et al., 2025) is trained with its published router objective plus KL distillation from the dense model (1000 steps). We gate whole layers rather than attention sub-blocks only, since gating attention alone cannot reach the tight budget on most backbones and is less accurate where both variants can be compared. As for MoSO, the first and last layers are never routed, and each cell reports the mean over three training runs.

Dr.LLM. Dr.LLM (Heakl et al., 2026) learns per-layer skip, execute or repeat decisions – the ℓ=1 slice $\{O_{1,0}, O_{1,1}, O_{1,2}\}$ of $\mathcal{O}$ – from routers supervised by offline tree search. All Dr.LLM numbers use its public code (commit `14e8837`). For the base models we add a 3-shot prefix to the prompt, without which the released reward never scores the dense LLaMA-2-7B or Qwen2.5-7B as correct (0 of 50 inputs each). For the ten cells of Table 1 we train its router on the frozen backbones with this code, supervised by execution paths sampled at each column's target budget in place of its tree search, which at these reduced budgets finds no accuracy-preserving path, whereas 8.1% of sampled paths preserve accuracy. On LLaMA-3.2-3B, Table 5 also lists Dr.LLM at its default (native) setting, and MoSO is more accurate than both. Appendix E.3 compares Dr.LLM at its default settings on three backbones and at several compute budgets.

SkipGPT. SkipGPT (Zhao et al., 2025) learns per-token skip decisions for attention and MLP sub-blocks. We train its router on the frozen backbones with MoSO's training data and token budget, and match its compute to MoSO's in each cell with a routing offset fitted on the training data; without this offset it runs at nearly full compute (0.974 on LLaMA-3.2-3B at the loose budget). Its compute counts the FLOPs actually executed.

Other methods. We do not report Mixture-of-Depths (Raposo et al., 2024): its router is trained jointly with the backbone from the start of pretraining, so a retrofit onto frozen weights would measure the retrofit rather than the method. FlexiDepth (Luo et al., 2025) is designed for training budgets far beyond ours, so we do not report it.

E Additional Results

E.1 Per-task scores for the measured-compute comparison

Tables 5 and 6 give the per-task scores behind Table 1, with each method's realized compute and %Dense retention. MoSO rows are means over three seeds; the standard deviation of MoSO's Avg_6 is at most 1.81 points in every setting and below one point (0.05–0.98) in nine of the ten. Router-Tuning rows are means over three training runs, whose standard deviation ranges from 0.50 to 12.43 Avg_6. The static baselines are deterministic. Running the same training with routing switched off, so that every layer is executed, gives 65.85 Avg_6 on LLaMA-3.2-3B, 66.00 on LLaMA-2-7B and 71.61 on LLaMA-3-8B-Instruct, against 65.60, 64.80 and 71.58 for the original models.

E.2 Schedule substitution

Figure 5 gives the schedule-substitution experiments of §4.3: the training is held fixed and only the set of skipped layers changes.

Table 5: **Per-task scores for Table 1** (Qwen2.5-7B, Mistral-7B, LLaMA-3.2-3B). **Compute**: FLOP ratio against dense, including the 0.2–0.8% cost of MoSO's additional trained components. Best non-dense Avg_6 per block in **bold**, second best underlined. ‡ShortGPT and SLEB select the same layers on Qwen2.5-7B at 0.750. †Router-Tuning: mean of three training runs. ♯SkipGPT and §Dr.LLM: see Appendix D.1; Dr.LLM (native), at its default setting, is not ranked.

	Method	Compute	MMLU	HSwag	ARC-C	PIQA	WG	BoolQ	Avg_6	%Dense
Qwen2.5-7B (L=28)	Dense	1.000	71.8	78.8	51.0	79.9	72.6	84.5	73.1	100.0
	ShortGPT	0.857	44.6	69.8	45.7	78.1	59.5	66.6	60.7	83.1
	SLEB	0.857	43.7	67.5	45.8	77.0	61.9	64.9	60.1	82.2
	LaCo	0.857	36.6	64.4	42.2	75.5	55.9	62.8	56.2	76.9
	BlockPruner	0.852	46.5	70.4	45.0	78.4	62.6	60.2	60.5	82.8
	Router-Tuning†	0.868	41.2	48.2	32.3	62.9	56.4	67.2	51.4	70.3
	SkipGPT♯	0.866	26.1	36.0	30.0	62.3	51.5	42.1	41.3	56.5
	Dr.LLM§	0.815	40.9	46.7	32.1	61.4	52.2	61.2	49.1	67.1
	MoSO	0.859	60.0	71.6	50.4	77.5	64.9	75.0	**66.6**	**91.0**
	ShortGPT‡	0.750	26.4	60.3	34.9	74.1	54.4	45.8	49.3	67.5
	SLEB‡	0.750	26.4	60.3	34.9	74.1	54.4	45.8	49.3	67.5
	LaCo	0.750	24.1	57.0	34.3	73.5	53.0	55.2	49.5	67.7
	BlockPruner	0.759	28.4	61.1	35.1	73.5	56.0	53.2	51.2	70.1
	Router-Tuning†	0.768	30.0	39.4	29.5	59.1	53.4	60.3	45.3	61.9
	SkipGPT♯	0.773	26.3	32.8	24.9	58.1	51.0	47.4	40.1	54.8
	Dr.LLM§	0.716	31.1	37.9	30.2	60.3	54.5	58.8	45.5	62.2
	MoSO	0.754	52.1	63.4	41.1	73.4	60.1	66.9	**59.5**	**81.4**
Mistral-7B (L=32)	Dense	1.000	59.5	81.1	53.8	82.1	74.2	83.7	72.4	100.0
	ShortGPT	0.875	59.0	75.4	47.3	77.9	70.9	80.4	68.5	94.6
	SLEB	0.875	42.5	70.9	41.0	77.4	67.3	77.8	62.8	86.8
	LaCo	0.875	23.5	68.0	34.7	77.3	60.0	64.1	54.6	75.4
	BlockPruner	0.877	33.4	72.3	43.0	77.7	68.7	78.3	62.2	86.0
	Router-Tuning†	0.877	24.9	29.1	23.9	53.7	51.1	56.5	39.9	55.1
	SkipGPT♯	0.888	26.7	52.3	26.4	63.8	52.3	61.9	47.2	65.2
	Dr.LLM§	0.894	48.2	62.3	29.5	56.7	52.2	74.7	53.9	74.5
	MoSO	0.877	59.2	75.5	48.9	78.4	71.8	84.4	**69.7**	**96.3**
	ShortGPT	0.750	52.9	59.7	37.3	69.3	67.1	71.7	59.7	82.4
	SLEB	0.750	26.1	60.9	33.5	74.4	60.8	54.4	51.7	71.4
	LaCo	0.750	25.0	52.6	30.6	67.7	56.4	54.4	47.8	66.0
	BlockPruner	0.762	24.1	57.2	31.9	70.6	62.4	69.8	52.7	72.8
	Router-Tuning†	0.754	25.0	36.9	27.6	56.3	51.9	48.3	41.0	56.6
	SkipGPT♯	0.792	24.3	47.2	25.5	59.8	50.3	48.7	42.6	58.9
	Dr.LLM§	0.806	34.0	63.5	35.6	72.7	59.0	68.9	55.6	76.8
	MoSO	0.755	58.7	66.4	42.7	73.0	70.8	82.0	**65.6**	**90.6**
LLaMA-3.2-3B (L=28)	Dense	1.000	54.1	73.6	45.9	77.4	69.9	72.9	65.6	100.0
	ShortGPT	0.857	49.8	63.5	40.8	69.8	65.8	61.7	58.6	89.2
	SLEB	0.857	34.4	61.3	33.8	72.3	60.2	53.5	52.6	80.1
	LaCo	0.857	23.5	51.8	30.2	71.4	54.0	58.7	48.3	73.5
	BlockPruner	0.862	26.2	54.8	28.5	70.7	57.1	60.8	49.7	75.7
	Router-Tuning†	0.873	28.8	56.1	29.0	68.6	54.1	54.2	48.5	73.9
	SkipGPT♯	0.867	46.0	64.9	42.1	73.8	63.2	67.1	59.5	90.7
	Dr.LLM (native)§	0.942	51.9	66.4	41.2	69.3	61.2	66.4	59.4	90.5
	Dr.LLM§	0.901	40.3	64.2	31.9	65.0	53.8	62.2	52.9	80.6
	MoSO	0.862	55.3	66.5	42.3	72.9	68.4	63.0	**61.4**	**93.5**
	ShortGPT	0.750	34.2	49.0	34.0	64.1	62.2	45.0	48.1	73.3
	SLEB	0.750	23.4	46.7	28.9	68.2	53.9	39.1	43.4	66.1
	LaCo	0.750	23.2	42.8	24.6	65.3	52.7	54.5	43.8	66.8
	BlockPruner	0.767	24.8	39.9	26.2	61.1	52.3	53.8	43.0	65.6
	Router-Tuning†	0.778	24.6	39.5	23.1	61.0	52.0	50.9	41.9	63.8
	SkipGPT♯	0.777	33.7	55.3	35.8	71.7	60.3	63.3	53.3	81.3
	Dr.LLM§	0.787	24.8	42.9	26.4	60.5	53.3	52.3	43.4	66.1
	MoSO	0.758	55.0	60.2	38.7	68.7	66.2	62.9	**58.6**	**89.3**

Table 6: **Per-task scores for Table 1, continued** (LLaMA-2-7B, LLaMA-3-8B-Instruct), same protocol and notation as Table 5. BlockPruner's compute column reports the FLOP ratio that each configuration actually executed.

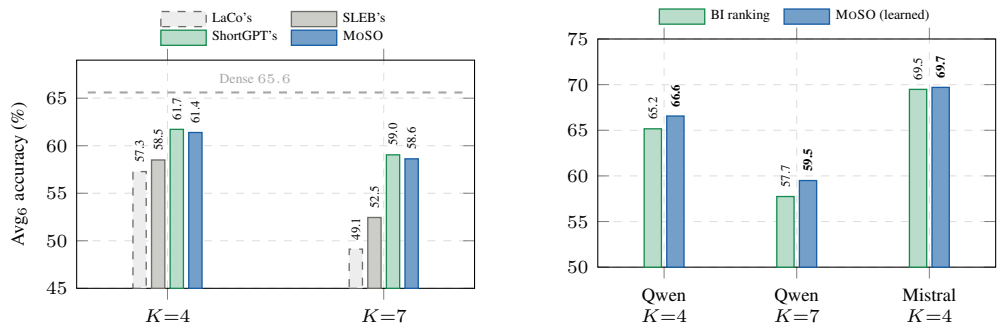
	Method	Compute	MMLU	HSwag	ARC-C	PIQA	WG	BoolQ	Avg_6	%Dense
LLaMA-2-7B (L=32)	Dense	1.000	40.8	75.9	46.3	78.8	69.2	77.7	64.8	100.0
	ShortGPT	0.844	34.1	68.4	39.1	73.2	67.1	74.9	59.5	91.8
	SLEB	0.844	26.5	65.5	37.5	75.4	60.4	63.2	54.7	84.5
	LaCo	0.844	26.0	62.3	34.5	73.9	57.1	59.8	52.3	80.7
	BlockPruner	0.849	25.5	61.5	33.7	72.3	61.6	66.6	53.5	82.6
	Router-Tuning†	0.847	31.0	67.1	37.7	74.2	64.5	67.2	56.9	87.9
	SkipGPT♯	0.839	23.8	31.9	22.5	56.1	50.6	54.6	39.9	61.6
	Dr.LLM§	0.875	28.6	56.8	31.5	64.4	56.0	62.1	49.9	77.0
	MoSO	0.847	45.4	70.9	41.4	75.3	68.2	74.7	**62.6**	**96.7**
	ShortGPT	0.750	23.5	60.1	35.3	68.1	65.6	62.2	52.5	81.0
	SLEB	0.750	25.1	60.2	33.3	72.8	57.9	62.1	51.9	80.1
	LaCo	0.750	24.5	54.1	30.3	69.5	53.8	50.4	47.1	72.7
	BlockPruner	0.749	23.0	48.5	27.2	68.0	54.1	62.1	47.2	72.8
	Router-Tuning†	0.755	25.1	51.0	28.5	61.0	57.0	57.7	46.7	72.1
	SkipGPT♯	0.744	23.8	30.0	25.7	52.7	50.9	39.2	37.1	57.2
	Dr.LLM§	0.812	25.1	34.0	27.4	58.9	55.6	57.9	43.1	66.6
	MoSO	0.755	43.8	65.9	39.6	71.3	67.2	75.5	**60.6**	**93.5**
LLaMA-3-8B-Instruct (L=32)	Dense	1.000	63.8	75.9	56.7	78.7	71.4	83.1	71.6	100.0
	ShortGPT	0.875	62.6	70.2	49.1	74.5	69.9	68.4	65.8	91.9
	SLEB	0.875	35.0	65.9	42.2	75.9	65.1	70.6	59.1	82.6
	LaCo	0.875	31.0	60.8	37.5	74.2	59.1	65.2	54.6	76.3
	BlockPruner	0.883	49.2	68.0	41.2	75.2	66.0	71.0	61.8	86.3
	Router-Tuning†	0.883	35.1	51.5	31.9	65.6	60.9	59.9	50.8	71.0
	SkipGPT♯	0.850	27.7	61.5	35.8	70.9	57.1	40.7	49.0	68.4
	Dr.LLM§	0.890	51.7	68.7	44.8	73.0	64.0	75.1	62.9	87.8
	MoSO	0.877	63.7	71.7	52.6	75.0	70.8	71.8	**67.6**	**94.4**
	ShortGPT	0.750	31.0	45.4	35.1	63.8	58.8	37.5	45.3	63.2
	SLEB	0.750	25.1	58.9	33.5	72.2	58.4	53.5	50.3	70.2
	LaCo	0.750	23.2	47.4	27.2	66.7	50.4	55.2	45.0	62.9
	BlockPruner	0.759	23.8	56.7	31.0	69.3	56.7	58.0	49.2	68.8
	Router-Tuning†	0.766	25.4	34.8	26.7	58.7	52.6	53.9	42.0	58.7
	SkipGPT♯	0.763	27.9	56.2	34.6	71.4	54.3	50.9	49.2	68.7
	Dr.LLM§	0.782	30.5	57.3	30.8	66.5	60.5	62.7	51.4	71.7
	MoSO	0.755	62.7	64.8	47.6	70.9	69.1	62.7	**63.0**	**88.0**

E.3 Comparison with Dr.LLM at its default settings

The Dr.LLM rows of Table 1 use its public code with target-budget supervision (Appendix D.1) and are reported at their own realized compute, which in eight of the ten cells is above MoSO's. We also train Dr.LLM with its public code and default settings on three models and compare against it directly in Table 7. Dr.LLM leaves the backbone weights unchanged, so both methods start from identical pretrained weights; we sweep its router penalty μ to approach MoSO's compute and read its realized compute from its own routing decisions. On LLaMA-3.1-8B-Instruct and Qwen2.5-7B-Instruct, MoSO is more accurate at every budget while executing fewer layers than the Dr.LLM point it is compared with. On LLaMA-3.2-3B it is more accurate than Dr.LLM at its default (native) setting, which executes more than two additional layers, and along its own frontier Dr.LLM loses accuracy quickly as it skips more. On LLaMA-3.1-8B-Instruct the margin is largest on MMLU, and on Qwen2.5-7B-Instruct on WinoGrande and BoolQ.

E.4 Routing behaviour and operator usage

Per-prompt routing. Figure 4 in the main text and Figure 6 here use routers over the full six-operator family, trained for this analysis on a 7,466-example ARC+DART pool (400 steps, KL distillation) with λ_{compute}=0.10 unless a sweep is shown. All LLaMA-3-8B-Instruct panels of Figure 4 come from one router, which executes $1.08\times$ the dense layers in its seven-task evaluation,



(a) Other methods' layers. (b) Learned vs. BI.

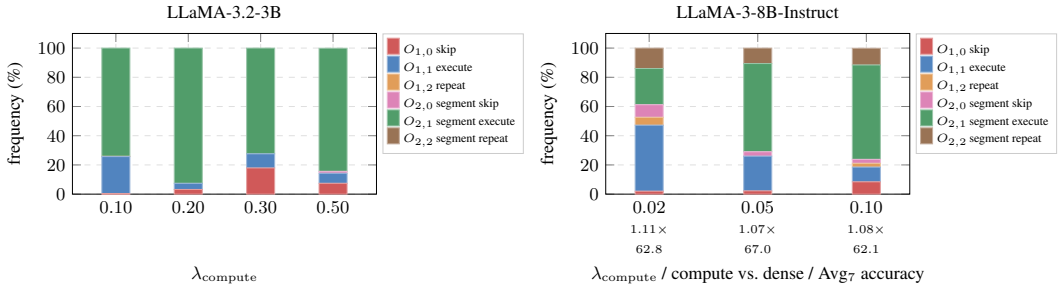
Figure 5: **Which layers are skipped matters under identical training.** All bars use identical training; only the set of skipped layers changes (dashed: dense). **(a)** On LLaMA-3.2-3B, given the layers SLEB or LaCo would remove, the recipe loses 2.9–9.5 points; MoSO's router converges to ShortGPT's layers on this model, and the two reach comparable accuracy. **(b)** Where BI mis-ranks layers (Qwen2.5-7B, overlap with MoSO's schedule 0.60 and 0.40) the learned schedule is better; on Mistral-7B the two coincide (overlap 1.00).

Table 7: **MoSO against Dr.LLM at its default settings** (public code), on identical backbone weights (native: its default setting). L_{eff} is the measured mean number of executed layers and μ is Dr.LLM's router penalty. The LLaMA-3.2-3B MoSO row is that of Table 1.

	Method	L_{eff}	MMLU	HSwag	ARC-C	PIQA	WG	BoolQ	Avg$_6$	%Dense
LLaMA-3.2-3B	Dense	28.00	54.1	73.6	45.9	77.4	69.9	72.9	65.6	100.0
	Dr.LLM (native, μ=−0.50)	26.17	51.9	66.4	41.2	69.3	61.2	66.4	59.4	90.5
	MoSO (K=4)	24.00	55.3	66.5	42.3	72.9	68.4	63.0	**61.4**	**93.5**
	Dr.LLM (μ=−0.640)	22.36	45.1	48.0	37.9	65.3	53.0	58.5	51.3	78.2
	Dr.LLM (μ=−0.663)	18.88	41.2	36.9	35.2	60.0	52.5	54.7	46.7	71.2
LLaMA-3.1-8B-I	Dense	32.00	68.1	79.1	55.1	80.8	74.2	84.0	73.6	100.0
	Dr.LLM (μ=−0.68)	28.71	51.2	66.1	44.8	72.9	68.6	76.8	63.4	86.2
	MoSO (K=4)	28.00	68.4	74.6	53.5	78.2	73.4	63.0	**68.5**	**93.1**
	Dr.LLM (μ=−0.69)	24.69	38.9	55.0	38.8	70.9	66.6	73.1	57.2	77.8
	MoSO (K=8)	24.00	67.7	68.2	48.6	72.1	72.1	63.4	**65.4**	**88.8**
Qwen2.5-7B-I	Dense	28.00	71.8	80.5	55.4	80.2	70.8	86.4	74.2	100.0
	Dr.LLM (μ=−0.70)	24.57	66.2	62.4	47.7	70.6	57.5	73.8	63.0	85.0
	MoSO (K=4)	24.00	65.9	70.8	49.9	75.7	65.9	82.3	**68.4**	**92.2**
	Dr.LLM (μ=−0.71)	23.62	65.2	60.1	45.5	68.1	56.7	72.3	61.3	82.7
	MoSO (K=7)	21.00	64.4	60.1	40.6	70.1	62.7	80.1	**63.0**	**84.9**

so these panels isolate where depth is moved. Figure 4(a) shows per-prompt routing on the 10 easiest and 10 hardest MMLU prompts (by dense gold-NLL): harder prompts trade skips for repeats within the same benchmark, without any per-difficulty supervision (26 skipped and 40 repeated layer cells, 33.4 layer applications on average, for the easy prompts; 20 and 50, 35.0 layer applications, for the hard ones). Over 350 prompts from seven tasks (5,927 routing decisions), this router uses two-layer segment operators for 78.9% of its decisions and single-layer operators for 21.1%, and repeats (14.1%) slightly outnumber skips (11.1%). Its operator mix depends on the workload (χ^2=355.9, 30 degrees of freedom): BoolQ and MMLU are skipped the most (16.2% of decisions each) and WinoGrande the least (4.8%), while ARC-Easy and ARC-Challenge receive the most repeats (about 18% of their decisions).

Operator-usage histograms. Figure 6 reports the frequency of each operator at the router's hard argmax, with one training recipe per backbone. On LLaMA-3.2-3B (200 held-out prompts, $\lambda_{\text{compute}} \in \{0.10, 0.20, 0.30, 0.50\}$), two-layer segment operators account for 72–93% of decisions at every λ_{compute}, and the router shifts between segment execution and segment skipping as the compute penalty changes. On LLaMA-3-8B-Instruct (350 prompts, the recipe of the Figure 4 router,



(a) LLaMA-3.2-3B. (b) LLaMA-3-8B-Instruct.

Figure 6: **Operator-usage histograms.** Stacked frequency of the six operators $O_{\ell,\rho}$ at the router's hard arg max. **(a)** LLaMA-3.2-3B, $\lambda_{\text{compute}} \in \{0.10, 0.20, 0.30, 0.50\}$; two-layer segment operators account for 72–93% of decisions. **(b)** LLaMA-3-8B-Instruct trained with the recipe of the Figure 4 router, $\lambda_{\text{compute}} \in \{0.02, 0.05, 0.10\}$ (λ=0.10 is the Figure 4 router); labels give realized compute relative to dense and seven-task average accuracy. Raising λ_{compute} shifts the router from single-layer execution toward segment execution, and these routers spend 1.07–1.11× the dense compute.

trained at $\lambda_{\text{compute}} \in \{0.02, 0.05, 0.10\}$), raising the penalty moves the router from single-layer execution (52.6% single-layer share at 0.02) toward segment execution (74% and 79% segment operators at 0.05 and 0.10); these routers spend 1.07–1.11× the dense compute.

Within-task difficulty and depth. At a fixed compute penalty, does the router spend more layers on harder *individual* prompts? For 200 held-out prompts per task we correlate the dense model's gold-continuation NLL with the number of layer applications chosen by the six-operator router at λ_{compute}=0.10 on LLaMA-3-8B-Instruct, whose compute varies across inputs. Stratifying within each task removes prompt length and format as confounds. Five of seven tasks show a significant positive within-task Spearman correlation (PIQA $r_s = +0.304$, ARC-E +0.219, HellaSwag +0.163, MMLU +0.154, ARC-C +0.145; all p<0.05), and pooled over all 1,400 prompts $r_s = +0.171$ (p=1.1 × 10^{-10}). The two exceptions are WinoGrande (r_s=−0.084) and BoolQ (r_s=−0.148), whose prompts follow a fixed template.

E.5 Efficiency

Wall-clock speedup. Table 8 reports measured speedups of skipping K layers on LLaMA-3-8B-Instruct in `bfloat16` with one shared schedule per batch, measured on the GPU of Appendix D.1 with PyTorch's scaled-dot-product attention; latency is the median over repeated timed runs. The speedup closely tracks the fraction of layers removed.

Table 8: **Measured speedup over dense** of skipping K layers on LLaMA-3-8B-Instruct (L=32, `bfloat16`), one shared schedule per batch, median of timed repetitions.

Layers skipped K	Executed layers	Prefill (batch 1)	Decode (batch 32)
0 (dense)	32	1.00×	1.00×
4	28	1.13×	1.12×
8	24	1.32×	1.30×
12	20	1.57×	1.55×

Router overhead. The router adds a small constant cost per forward pass, independent of sequence length: about 0.6% of a batch-8 prefill, and less for longer sequences.

Training cost. Table 9 compares training and preparation costs. One MoSO training run takes between 0.76 and 1.45 h on a single GPU and sees fewer than 10M training tokens. Static pruners need only a calibration pass per budget, and Dr.LLM additionally needs generated supervision before its router can be trained.

Table 9: **Training cost** of one training or calibration run: trainable parameters, optimization steps, wall-clock, peak memory, and the supervision each method needs before training. Baseline rows are measured on an idle GPU; the MOSO row, which trains the router alone, is measured on LLaMA-3.2-3B on a shared GPU and is an upper bound. Dr.LLM's wall-clock covers router training only.

Method	Trainable	Steps	Wall-clock	Peak	Supervision
ShortGPT	0	—	0.6 s	15.7 GiB	calibration set
LaCo	0	—	66.8 s	16.4 GiB	calibration set
SLEB	0	—	68.8 s	16.0 GiB	calibration set
BlockPruner	0	—	143.1 s	16.0 GiB	calibration set
Router-Tuning	$L\times(d{+}1)$	1000	894 s	30.4 GiB	none
Dr.LLM	11.02M	$\sim$2500	1592 s	6.2 GiB	sampled trajectories
MOSO	0.93M	2$\times$2000	0.79 h	21.4 GiB	none