ONLY THE BOUNDARY MATTERS: EFFICIENT MASK LEARNING FOR SPARSE NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse neural networks construct subnetworks by selectively retaining weight connections and are widely used for model compression, inference acceleration, and parameter-efficient adaptation. Binary masks implement connection selection in sparse networks, while learned masks use continuous scores to adapt this selection to data or tasks. However, discrete masks still rely on dense floating-point scores, gradients, and optimizer states, whose update overhead grows with the number of candidate weights. We identify a simple rank-stability phenomenon: under bounded score updates, score intervals that cannot cross the Top-k threshold correspond to fixed mask decisions, and only boundary scores can change their inclusion status. Based on this observation, we propose Boundary, an active-set optimization framework for learned binary masks. It stores determined decisions as fixed bits, maintains continuous optimization state only for boundary scores, and completes selection using the remaining quota; periodically committing state and rebuilding the active set allows previously fixed coordinates to re-enter optimization. Experiments on ImageNet and random-weight subnetworks show that Boundary substantially reduces controller state while maintaining accuracy close to the original mask optimization. On Pythia-1.4B, its language-model quality is close to Full-score; in an independent short-context timing benchmark, controller state and peak allocated memory decrease by 80.6% and 67.1%, respectively, with a measured training-loop speedup of 9.20×. Boundary turns Top-k rank stability into a concrete rule for allocating optimization resources according to whether mask decisions can change, providing a direct way to reduce mask-training cost.

1 INTRODUCTION

Binary masks are widely used in network pruning, sparse subnetwork search, and pretrained-model adaptation. By multiplying weights by zeros and ones, they select the connections that form a subnetwork while leaving the underlying weights fixed or separately trained. Piggyback, Edge-Popup, and MaskLLM apply this idea to frozen models, random-weight subnetworks, and large language models (Mallya et al., 2018; Ramanujan et al., 2020; Fang et al., 2024). Learning the discrete choices, however, typically adds a continuous score, gradient, and optimizer state for every candidate connection. These dense arrays make mask optimization itself a substantial part of the training cost. Figure 1(a) illustrates the mismatch between a one-bit executed mask and its dense training state.

Our starting point is classical hard thresholding in sparse learning (Blumensath & Davies, 2009). Top-k selection depends on the ranking of the score magnitudes, not on every score value separately. Under bounded score updates, a coordinate whose score interval cannot cross the Top-k threshold keeps the same mask decision throughout the interval; only scores near the threshold can change their inclusion status. In ResNet-18/CIFAR-10, the 10% of scores closest to the threshold cover about 90% of endpoint-mask changes late in training, with the concentration varying across training stages. Figure 1(b) visualizes this ranking-stability observation.

Building on this principle, we propose **Boundary**. As shown in Figure 1(c), Boundary stores continuous scores and optimizer state only for the coordinates that may still change the mask, represents the remaining decisions as fixed bits, and periodically rebuilds the active set so that previously fixed

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.

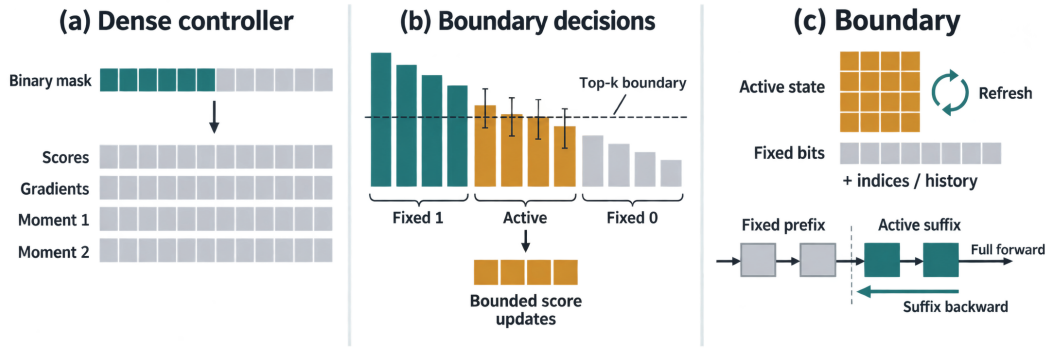

Figure 1: **Which mask scores need to remain trainable?** (a) Full-score learning stores scores, gradients, and optimizer moments for every eligible weight. (b) Allowed score changes distinguish mask entries that stay at 1 or 0 from scores retained for training; the bars are schematic, and Proposition 1 defines the checks. (c) Boundary stores trainable scores compactly and periodically restores saved scores to repeat these checks. Fixing the initial layers' masks additionally removes their backward computation; the full forward pass still runs and the number of selected weights is unchanged.

coordinates can re-enter. The same representation can be combined with a fixed network prefix to remove its backward computation.

Across three architectures on ImageNet-1K, Boundary with the frontier has a cross-architecture mean accuracy drop of 0.13 percentage points relative to Full-score while reducing mask-training state by up to 82.6% and reaching $1.243\times$ paired step speedup. On Pythia-1.4B with deterministic $2{:}4$ masks, it keeps language-model quality close to Full-score; an independent short-context benchmark reduces controller state by 80.6% and peak allocated memory by 67.1%, with a measured training-loop speedup of $9.20\times$. These results turn rank stability into a practical rule for allocating training state and computation according to which mask decisions can still change.

Our main contributions are:

- **A rank-stability view of mask optimization.** We connect bounded score updates to the mask decisions that can still change, giving a direct rule for allocating continuous optimization state.
- **Boundary active-set optimization.** Boundary stores active score state, refreshes that state during training, and supports a fixed prefix for reducing backward computation without changing the host mask objective.
- **Cross-model evaluation.** We measure quality, controller state, memory, and training speed across vision models, random-weight subnetworks, and deterministic language-model masks.

2 Related Work

Learning binary masks over fixed weights. Pruning, sparse subnetwork search, and frozen-model adaptation all use binary connectivity decisions to select subnetworks while keeping the underlying weights fixed or training them separately. Magnitude pruning (Han et al., 2015), the lottery-ticket line (Frankle & Carbin, 2019), and data- or gradient-based criteria such as SNIP, GraSP, and SynFlow (Lee et al., 2019; Wang et al., 2020; Tanaka et al., 2020) mainly answer which connections should be retained. Supermasks and Edge-Popup learn real-valued scores for candidate connections and threshold them to obtain the executed subnetwork (Zhou et al., 2019; Ramanujan et al., 2020); Piggyback applies the same idea to task-specific adaptation of a frozen network (Mallya et al., 2018). Score-based mask-learning methods such as Edge-Popup and Piggyback maintain a continuous score and its gradient for each candidate connection, with additional per-score state under momentum-based or adaptive optimizers. Boundary inherits the host method's mask objective, retention quota, and straight-through update, while changing how score state is allocated. Boundary therefore targets the controller cost of mask learning rather than redefining the mask objective.

Table 1: How Boundary differs from topology learning and optimizer-state compression. Boundary preserves the host mask rule and allocates continuous state only to boundary scores.

Method family	Main object optimized	Topology rule	Mask-controller state	Relation to Boundary
Pruning / topology search	weights or topology	method-specific	method-dependent	defines retained connections
Learned binary masks	mask scores	Top-k or threshold	dense scores, gradients, optimizer state	direct mask-learning host
Structured N:M masks	weights or mask scores	fixed group quota	dense or groupwise score state	compatible structured host
Optimizer-state efficiency	optimizer statistics	topology unchanged	compressed per-coordinate state	complementary state axis
Boundary (ours)	active mask scores	host rule unchanged	active scores/state + fixed bits	proposed state allocation

Dynamic sparse training and structured sparsity. Dynamic sparse training studies how the active topology changes during optimization. RigL periodically replaces low-magnitude connections using gradient information (Evci et al., 2020), while Top-KAST uses a larger backward support than forward support (Jayakumar et al., 2020). BiDST formulates dynamic sparse training as a bi-level optimization problem over weights and masks (Ji et al., 2024). Structured N:M methods introduce group constraints and specialized mask updates, including SR-STE, S-STE, and BAME (Zhou et al., 2021; Hu et al., 2024; Yang et al., 2025). SERo explores and re-optimizes channel-pruned ViT structures (An et al., 2025). One-shot LLM pruning methods such as SparseGPT and Wanda, together with learnable N:M mask training such as MaskLLM, target sparse topology or model quality at scale (Frantar & Alistarh, 2023; Sun et al., 2024; Fang et al., 2024). These works focus on how a topology is discovered, replaced, or constrained. Boundary can be placed on top of these host rules: it preserves the host's group structure and retention quota while reducing the score and optimizer state maintained as the topology is learned. Boundary thereby forms a compatible efficiency layer for Top-k, N:M, and fixed-weight mask methods. Within each benchmark, we retain the declared sparsity setting and compare model quality with training resources.

Optimizer and training-state efficiency. Adam and AdamW maintain per-coordinate first- and second-moment state (Kingma & Ba, 2015; Loshchilov & Hutter, 2019), while quantized and factored optimizers compress the numerical representation of these statistics. These methods reduce the bytes used by each optimizer state. Boundary is complementary: it reduces the number of score states that must be maintained while leaving the host optimizer unchanged. The experiments report the two effects separately through controller-state size, checkpoint size, and runtime.

Hard thresholding, screening, and rank stability. Iterative hard thresholding alternates gradient updates with selection of the largest coordinates (Blumensath & Davies, 2009); Gap Safe screening uses a safe region to remove variables that cannot enter the optimal active set (Fercoq et al., 2015). These methods reason about which coordinates need to remain in an optimization support, but they do not address the dense controller state maintained while binary mask scores are learned. Boundary instead leaves the host mask objective, sparsity quota, and update rule unchanged, and changes only how continuous score state is allocated: state is kept for mask decisions that may change while determined decisions use fixed bits. Sections 3–4 give the derivation and implementation; Table 1 summarizes the distinction.

3 Where Does Learnable Mask Optimization Actually Happen?

Mask learning searches discrete connections through continuous scores. Multiple score states represent the same masked network; connectivity changes require rank exchanges between selected and excluded positions. We use rank stability to allocate score state, then examine how fixing the initial layers' masks reduces backward computation.

3.1 Rank Stability under Bounded Updates

For frozen weights w and trainable scores s, the binary mask $m(s) = \mathrm{TopK}(|s|, k)$ retains k of n positions independently within each group. Mask learning optimizes

$$\min_s F(s), \qquad F(s) = \mathbb{E}_{(x,y)}\mathcal{L}(f(x; w \odot m(s)), y)\,. \tag{1}$$

Ties follow original-index order. Scores affect the model only through the mask, so an unchanged selected set preserves $F(s)$ with other model state fixed. A straight-through estimator (STE) can still update scores and optimizer state without changing the mask (Bengio et al., 2013).

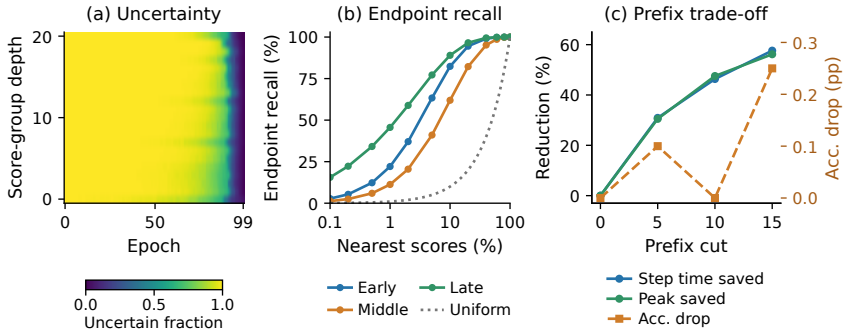

Figure 2: **Training-phase structure and fixed-prefix efficiency.** ResNet-18/CIFAR-10. (a) Boundary fractions from Proposition 1, using each group's largest recorded cumulative score displacement over an epoch; means over three seeds. (b) Coverage of epoch-endpoint mask changes by scores nearest the threshold at the epoch's start, in early (0–32), middle (33–65), and late (66–99) training; the dotted line is uniform-selection expectation. (c) Step-time and peak-allocated-memory reductions and development accuracy drops relative to cut 0 after one epoch of continuation from epoch 90; medians of paired changes over two seeds.

Rank changes depend on the initial gap and the allowed displacement. Let a be the scores at the start of an update interval, with $|s_i - a_i| \le r_i$ and $r_i \ge 0$. The reverse triangle inequality gives $\big||s_i| - |a_i|\big| \le r_i$, hence

$$|s_i| - |s_j| \ge |a_i| - |a_j| - (r_i + r_j). \tag{2}$$

Thus $|a_i| - |a_j| > r_i + r_j$ ensures that i always precedes j. For common radius r and $1 \le k < n$, a gap above $2r$ between the kth and $(k+1)$th largest initial magnitudes preserves the selected set.

Coordinate-level stability follows from the magnitude intervals $\ell_i = \max(0, |a_i| - r_i)$ and $u_i = |a_i| + r_i$. Define

$$P_i = \#\{j \ne i : u_j \ge \ell_i\}, \qquad C_i = \#\{j \ne i : \ell_j > u_i\}. \tag{3}$$

Here P_i counts positions that could reach or exceed i under these intervals; C_i counts those guaranteed to exceed it strictly. The fixed-on and fixed-off sets are $\mathcal{F}^+ = \{i : P_i < k\}$ and $\mathcal{F}^- = \{i : C_i \ge k\}$. Positions not determined by these rules form the *boundary active set* $\mathcal{A} = [n] \setminus (\mathcal{F}^+ \cup \mathcal{F}^-)$.

Proposition 1 (Interval rank stability)**.** *For every $s \in \mathcal{I}(a, r) = \prod_i [a_i - r_i, a_i + r_i]$, positions in $\mathcal{F}^+$ remain selected and those in $\mathcal{F}^-$ remain excluded. Consequently,*

$$\{i : m_i(s) \ne m_i(s')\} \subseteq \mathcal{A}, \qquad s, s' \in \mathcal{I}(a, r). \tag{4}$$

Proof. At most P_i coordinates precede i, so $P_i < k$ guarantees selection. At least C_i coordinates strictly precede i, so $C_i \ge k$ guarantees exclusion. Both statements hold throughout the interval, confining all mask changes to $\mathcal{A}$. Appendix A gives the full proof. □

Within the interval, $\mathcal{F}^+$ and $\mathcal{F}^-$ supply fixed bits to the forward mask, while continuous optimization focuses on $\mathcal{A}$. Boundary enforces the update range by projection and revisits the partition at training events.

3.2 Training-Phase Structure of Mask Changes

The boundary size depends jointly on score gaps and update magnitudes. Figure 2(a) shows that, under intervals constructed from recorded score motion, more mask entries can be fixed late in training, opening greater opportunities to compress score state.

Epoch-endpoint mask differences also concentrate near the ranking threshold (Figure 2(b)). Nearby scores cover substantially more endpoint changes than uniform selection, with concentration varying across training phases. This pattern and the interval analysis motivate periodic updates to the active set using current ranks and update ranges.

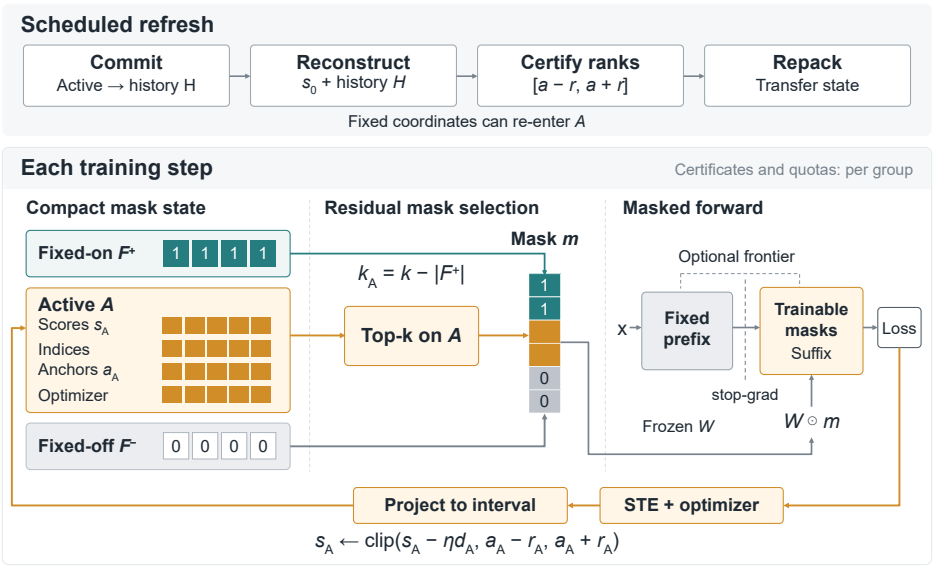

Figure 3: **Boundary optimization and scheduled refresh.** The compact mask state separates fixed-on, active, and fixed-off coordinates; the residual Top-k selection is performed on active scores. Projected updates modify only the active scores. At a scheduled refresh, active values are committed, scores are reconstructed, rank bounds are certified, and the compact state is repacked, allowing fixed coordinates to re-enter the active set. The optional backward frontier removes backward computation through a fixed prefix.

3.3 Computational Gains from a Fixed Prefix

Compressing score state leaves gradient dependencies intact. Even a frozen-weight layer must propagate input gradients when earlier scores remain trainable. Computational savings also depend on the network locations of active scores.

Let $h_c = f_{\le c}(x; w_{\le c}, m^{\text{fixed}}_{\le c})$ contain all tensors crossing the fixed-prefix cut c. For suffix mask multipliers u, write $G(h_c, u) = \mathcal{L}(f_{>c}(h_c; w_{>c} \odot u), y)$ and evaluate derivatives at $u = m_{>c}$. With prefix masks and model state fixed and no dependence of h_c on trainable suffix variables, $\partial h_c / \partial u = 0$. The chain rule gives

$$\nabla_u G(\text{stopgrad}(h_c), u) = \nabla_u G(h_c, u). \tag{5}$$

Detaching this fixed prefix preserves suffix-mask gradients and, through the same STE, surrogate score gradients (Baydin et al., 2018). The prefix still supplies forward features, while its backward computation and associated intermediate activations are removed.

Figure 2(c) separates the two execution effects: compact resident score state and removal of prefix backward work. The comparison shows how a fixed prefix changes step time and peak allocation beyond the controller-state measurement.

4 Boundary: Optimizing Only Boundary Scores

Boundary represents mask optimization with a boundary active set: it trains and stores scores that can still change the Top-k decision within the current update interval, refreshes that set at scheduled events, and optionally fixes an initial network prefix to remove its backward computation (Figure 3). A *score group* is one tensor for per-tensor Top-k masking in the vision host, or a four-weight block with quota two for deterministic 2:4 masking. All certificates and residual quotas are applied independently within each group.

4.1 Representing, Updating, and Refreshing Boundary Scores

At the start of each event interval, Boundary anchors the logical scores and sets a prescribed radius. It stores trainable scores and optimizer state inside the resulting boundary active set and fixed binary decisions outside it. These two parts form the full mask at every iteration, while projection keeps active updates inside the event's score intervals.

For a Top-k group, the fixed-on set $\mathcal{F}^+$ already occupies $|\mathcal{F}^+|$ selected positions. Thus only $k_{\mathcal{A}} = k - |\mathcal{F}^+|$ positions are selected from the active set:

$$m_i = \begin{cases} 1, & i \in \mathcal{F}^+, \\ [\mathrm{TopK}_{\mathrm{id}}(|s_{\mathcal{A}}|, k_{\mathcal{A}})]_i, & i \in \mathcal{A}, \\ 0, & i \in \mathcal{F}^-. \end{cases} \tag{6}$$

The active vector retains original-index labels, which id uses to break magnitude ties. Equation 6 implements the host Top-k rule with fixed bits and the residual active scores; fixed-on weights still contribute to the forward pass.

We pack active scores, indices, optimizer state, and anchors $a_{\mathcal{A}}$; the interval radius is set by the training protocol. The host STE supplies gradients. The optimizer direction $d^t_{\mathcal{A}}$ denotes the complete active-domain update before projection, including the optimizer's moment, bias-correction, and decoupled weight-decay terms when present. The projected update is

$$s^{t+1}_{\mathcal{A}} = \mathrm{clip}\big(s^t_{\mathcal{A}} - \eta_t d^t_{\mathcal{A}}, a_{\mathcal{A}} - r_{\mathcal{A}}, a_{\mathcal{A}} + r_{\mathcal{A}}\big)\,. \tag{7}$$

Anchors remain fixed within an interval and reset at the next event.

Packing is a representation choice for the active-restricted optimizer, not a second optimization rule; the host update can gather the active entries and scatter them back to their original indices when needed.

Scheduled events allow previously fixed coordinates to re-enter optimization. Each event commits active values to sparse history, reconstructs eligible scores from history and the initializer, starts a new interval, and recomputes the boundary. Optimizer state is transferred for coordinates that remain active and initialized according to the declared host-specific re-entry policy: the vision host initializes moments for new or re-entering coordinates, whereas the language host restores its retained sparse moments. The event-local equivalence uses the same policy in its full-layout reference.

4.2 Fixing a Network Prefix

At a scheduled event, Boundary can fold masks before a legal stage or block boundary c into the frozen weights and discard their score state and history. Subsequent refreshes and updates process only the suffix. This Backward Frontier is an optional execution optimization, separate from the active-set representation. The prefix still computes features, without recording a backward graph:

$$h_c = \mathrm{stopgrad}\big(f_{\le c}(x; w_{\le c}, m^{\mathrm{fixed}}_{\le c})\big)\,, \qquad \hat{y} = f_{>c}(h_c; w_{>c}, m_{>c}). \tag{8}$$

Here h_c includes all tensors crossing the cut, including residual branches. Section 3.3 shows that this preserves suffix-mask gradients for the same fixed-prefix model while removing prefix backward work and saved activations. Boundary-only uses $c = 0$; $c > 0$ additionally contracts the backward graph.

4.3 Algorithm and Resource Cost

Algorithm 1 combines minibatch updates and scheduled events. The right-column loop calls the left-column RecomputeBoundary procedure at each event, updating shared state in place. Here $s_{\mathcal{A}}$ is the packed score vector, and $c_{\mathrm{live}} = 0$ denotes an unfixed prefix, with $f_{\le 0}(x) = x$.

Algorithm 1 Eventwise Boundary mask optimization with active-state refresh.

(A) Initialization and event subroutine

Input: $w, s^0, k, r, \mathcal{E}$; optimizer $\mathcal{O}$; optional (c, e_c)
State: $(\mathcal{H}, \mathcal{A}, \mathcal{F}^+, \mathcal{F}^-, a, \Omega)$

$s \leftarrow s^0, a \leftarrow s^0, \mathcal{H} \leftarrow \emptyset, c_{\text{live}} \leftarrow 0$
$(\mathcal{F}^+, \mathcal{F}^-, \mathcal{A}) \leftarrow \text{RefreshBoundary}(s, a, r, k)$
$(s_{\mathcal{A}}, \Omega) \leftarrow \text{Pack}(s, \mathcal{A}, \mathcal{O})$
procedure RECOMPUTEBOUNDARY(t)
  $\mathcal{H} \leftarrow \text{Commit}(\mathcal{H}, s_{\mathcal{A}}, \Omega)$
  $s \leftarrow \text{Reconstruct}(s^0, \mathcal{H}); a \leftarrow s$
  $(\mathcal{F}^+, \mathcal{F}^-, \mathcal{A}) \leftarrow \text{RefreshBoundary}(s, a, r, k)$
  $(s_{\mathcal{A}}, \Omega) \leftarrow \text{TransferPack}(s, \Omega, \mathcal{H}, \mathcal{A})$
  if $t = e_c$ **then**
    form current m by Eq. 6; fold prefix masks into w
    remove prefix state/history; $c_{\text{live}} \leftarrow c$
  end if
end procedure

(B) Main minibatch loop

for minibatch (x, y) at step $t = 0, \ldots, T-1$ **do**
  if $t \in \mathcal{E}$ **then**
    RECOMPUTEBOUNDARY(t) ▷ remaining groups
  end if
  $k_{\mathcal{A}} \leftarrow k - |\mathcal{F}^+|$
  $m \leftarrow \text{ResidualTopK}_{\text{id}}(s_{\mathcal{A}}, k_{\mathcal{A}}, \mathcal{F}^+, \mathcal{F}^-)$
  $h \leftarrow \text{stopgrad}(f_{\le c_{\text{live}}}(x; w, m))$
  $\hat{y} \leftarrow f_{>c_{\text{live}}}(h; w, m); \ell \leftarrow \mathcal{L}(\hat{y}, y)$
  $g_{\mathcal{A}} \leftarrow \text{STEGrad}(\ell, s_{\mathcal{A}})$
  $(d_{\mathcal{A}}, \Omega) \leftarrow \mathcal{O}(\Omega, g_{\mathcal{A}})$
  $s_{\mathcal{A}} \leftarrow \text{clip}(s_{\mathcal{A}} - \eta_t d_{\mathcal{A}}, a_{\mathcal{A}} - r_{\mathcal{A}}, a_{\mathcal{A}} + r_{\mathcal{A}})$
end for
return $(w, s_{\mathcal{A}}, \mathcal{H}, \mathcal{F}^{\pm}, a, \Omega, c_{\text{live}})$

For N scores, a dense FP32 Adam mask controller uses approximately $16N$ bytes for scores, gradients, and two moments; this is a logical controller-state convention, not a claim about total device peak memory. With $A = |\mathcal{A}|$ active positions and H historical positions not already counted in $\mathcal{A}$, the compact representation requires

$$M_{\text{Boundary}} \simeq b_{\mathcal{A}} A + b_{\mathcal{H}} H + M_{\text{bits}} + M_{\text{group}}. \tag{9}$$

$b_{\mathcal{A}}$ covers active values, optimizer state, indices, and anchor data; under FP32 scores and gradients, two FP32 moments, an int32 index, and an FP32 anchor this is 24 bytes per active entry before metadata. $b_{\mathcal{H}}$ covers stored history outside the current active set. Fixed decisions use $O(N)$ bits, while floating-point state and indices scale as $O(A + H)$.

Our dense-kernel path materializes expanded masks; prefix fixing additionally reduces backward work and saved activations. Sorting interval endpoints costs $O(\sum_g n_g \log n_g)$, or $O(N)$ for fixed-size 2:4 groups. For refresh cost C every J steps, amortized step cost is $T_{\mathcal{A},c} + C/J$.

5 EXPERIMENTS

This chapter follows one question: when most mask scores no longer compete for the Top-k decision, can their continuous optimization state be removed without changing mask quality, and when does that logical reduction become a systems gain? We therefore separate three measurements throughout the chapter: mask quality, logical controller state, and execution cost. The first two are the direct consequence of the active-set representation; the last depends on whether the host can exploit the representation during the forward and backward passes.

5.1 EXPERIMENTAL SETUP

We use frozen ResNet-50 (He et al., 2016), ConvNeXt-T (Liu et al., 2022), and DeiT-B (Touvron et al., 2021) on ImageNet-1K (Russakovsky et al., 2015) with 50% per-tensor Top-k masks. We compare Full-score, Hard-freeze, Boundary-only, and Boundary + frontier, using the same training stream and development-based model-selection protocol for all methods. Main quality results use the official validation set; mechanism ablations use the development split. Controller state, peak allocation, and training time are measured separately.

For transfer, Pythia-1.4B (Biderman et al., 2023) uses deterministic 2:4 groups. We report WikiText-2 perplexity, six zero-shot tasks, controller state, allocator peak, and with-data training-loop time. Timing comparisons use the same host, warm-up, and repeated-measurement protocol. Appendix B collects implementation details, radius and refresh schedules, optimizer-state policies, external-method adaptations, and supplementary experiments.

5.2 MASK QUALITY IS PRESERVED WHILE THE CONTROLLER BECOMES COMPACT

Table 2 gives the main ImageNet result. Boundary + frontier has a cross-architecture mean accuracy drop of 0.13 percentage points: ResNet-50 and DeiT-B remain essentially unchanged, while ConvNeXt-T drops by 0.38 points. It substantially reduces logical controller state (Table 3). Thus

the same active-set representation transfers across a CNN, a modern CNN, and a Transformer without a separate score layout. The active-set representation also transfers to a frozen-random-weight Edge-Popup-style host on CIFAR-10 (Table 13).

Table 2: **ImageNet-1K quality and controller state at 50% masks.** Top-1 is mean $\pm$ standard deviation over three seeds on the official validation set; controller state is the logical ratio relative to Full-score. Bold marks the best learned-mask accuracy and smallest state.

	ResNet-50		**ConvNeXt-T**		**DeiT-B**	
Method	Top-1 (% ↑)	Ctrl. state (% ↓)	Top-1 (% ↑)	Ctrl. state (% ↓)	Top-1 (% ↑)	Ctrl. state (% ↓)
Dense pretrained	80.85	–	82.52	–	81.80	–
Magnitude mask	66.58	–	46.77	–	74.18	–
Full-score STE	$\mathbf{79.72} \pm 0.03$	100.0	$\mathbf{80.94} \pm 0.09$	100.0	80.20 ± 0.06	100.0
Hard-freeze	79.48 ± 0.10	–	80.60 ± 0.03	–	80.11 ± 0.05	–
Boundary	79.71 ± 0.07	23.7	80.58 ± 0.08	22.9	$\mathbf{80.21} \pm 0.03$	23.1
Boundary + frontier	79.71 ± 0.03	**23.5**	80.55 ± 0.03	**22.7**	80.20 ± 0.03	**17.4**

The longer-training comparison in Table 12 shows the same pattern after the checkpoints are fixed: the active set remains useful beyond a single short recovery interval.

5.3 Logical state and GPU peak measure different bottlenecks

Controller compression does not proportionally reduce backbone computation or GPU peak. Vision execution still uses full-shaped masked weights, backbone activations, and temporary workspaces. Consequently, Boundary-only substantially reduces controller state but changes step time and peak allocation little (Table 3).

The frontier reduces prefix backward computation, but setup and reconstruction overhead can offset steady-state savings: total time improves modestly on ResNet-50 and ConvNeXt-T but increases on DeiT-B. In contrast, Pythia's compact state directly participates in 2:4 mask construction, with the largest measured savings in the forward region containing it. Substantial speedup thus requires not only storing fewer score states, but eliminating their associated work.

5.4 Pythia exposes where the execution gain comes from

Table 4 shows that Boundary preserves Pythia's language-model quality while reducing controller state, allocator peak, and with-data loop time.

The component profile in Table 5 locates the largest time reduction in the full forward region, which includes mask construction; optimizer and projection costs also fall. Full-score uses stable four-score sorting, whereas Boundary uses direct rank comparisons on active groups, so the measured gain combines active-set reduction with its execution path. The context-2,048 run in Table 8 also achieves $7.98\times$ loop speedup and 55.4% lower peak allocation relative to the same Full-score implementation, extending the gain beyond short contexts.

External structured-sparsity methods provide context for the Pythia host. Table 8 reports the corresponding one-shot and learnable-mask operating costs; the matched quality comparison remains the Full-score–Boundary pair in Table 4.

5.5 Recompilation recovers mask quality

Table 6 compares the design choices on the development set. A fixed band can compress state but degrade quality, whereas eventwise recompilation refreshes the candidate set as scores move and recovers the quality gap.

Adding the frontier changes execution rather than the mask policy; it improves step efficiency without changing recovered mask quality. Count-matched, storage-matched, projected Full-score, and prefix controls are reported in Table 11.

Table 3: **ImageNet-1K systems cost across backbones.** Candidate/Full-score ratios (↓); controller and checkpoint columns are logical state ratios, and the remaining columns are paired execution measurements.

Backbone	Variant	Candidate / Full-score ratio (↓)				
		Ctrl. state	Ckpt. state	GPU peak	Median step	Setup + training
ResNet-50	Boundary-only	0.237	0.280	1.000	1.023	1.029
	Boundary + frontier	0.235	0.277	1.000	0.846	0.980
ConvNeXt-T	Boundary-only	0.229	0.270	0.992	1.007	1.136
	Boundary + frontier	0.227	0.268	0.992	0.804	0.945
DeiT-B	Boundary-only	0.231	0.272	0.980	0.974	1.367
	Boundary + frontier	0.174	0.205	0.980	0.819	1.309

Table 4: **Pythia-1.4B 2:4 mask learning.** Quality uses three-seed C4 runs; resource columns use the separate five-repeat same-host timing profile.

	Quality		Training resources		
Method	PPL↓	Acc (%)↑	Controller saving (%)↑	GPU peak saving (%)↑	Training speedup (×)↑
Full-score	**20.64**	**43.29**	0.0	0.0	1.00
Boundary	20.70	43.28	**80.6**	**67.1**	**9.20**

Quality references (PPL / 6-task accuracy): dense pretrained, 10.94 / 52.65%; magnitude 2:4, 724.82 / 32.17%.

Table 5: **Pythia component profile.** Component means (s) over all 50 steps; median step excludes 10 warm-up steps.

Component	Full-score	Boundary	Boundary / Full
Data	0.0000749	0.0000678	0.906
Full forward region	1.46318	0.08463	0.058
Backward	0.04112	0.03256	0.792
Optimizer	0.05759	0.00661	0.115
Projection	0.02346	0.00367	0.156
Median step	1.58279	0.118807	0.075

Table 6: **Development-set mechanism ablation.** Top-1 changes are relative to Full-score; controller saving is the final logical state.

	Top-1 change (pp) ↑			Controller saving (%) ↑		
Method	RN50	CN-T	DeiT-B	RN50	CN-T	DeiT-B
Full-score STE	0.00	0.00	0.00	0.0	0.0	0.0
Random-active	−0.23	−0.58	−0.28	30.9	31.6	31.8
Fixed-band	−0.11	−1.48	−0.15	89.7	90.0	90.0
Stability-active	−0.12	−1.29	−0.17	74.0	74.9	74.7
Static Boundary	−0.08	−1.17	−0.16	80.1	80.7	80.8
Eventwise Boundary	−0.01	−0.30	−0.04	76.3	77.1	76.9
+ frontier	0.00	−0.26	−0.10	76.5	77.3	82.6

Together, these experiments show that mask learning need not maintain full optimization state for every score throughout training: boundary scores handle the current selection competition, while periodic refresh adapts the active set as training progresses, preserving mask quality with less state. Whether this compression yields systems gains depends on how execution uses the compact representation: vision models mainly reduce backward computation by fixing a prefix, whereas Pythia's packed mask path reduces both state-maintenance and mask-computation overhead. Boundary's efficiency thus comes from two complementary changes: reducing the scores that require optimization and reducing the computation needed to execute their updates.

6 Conclusion

We introduced Boundary, an active-set optimization framework that allocates mask-learning resources according to which discrete decisions can still change. Under bounded score updates, Top-k rank stability identifies fixed mask decisions, allowing continuous optimization state to be concentrated on boundary scores. Periodic refresh lets previously fixed coordinates re-enter optimization, while compact mask execution and prefix freezing provide routes to computational savings. Experiments on vision models, random-weight subnetworks, and Pythia show substantial controller-state reductions with quality close to Full-score. The systems results further show that translating these savings into lower peak memory and faster training depends on how the host executes the compact representation. Boundary thus makes mask-decision stability a practical basis for reducing training cost, beyond sparsity in the final network. Future work can extend this principle through adaptive update intervals and specialized kernels for active-score selection and mask construction.

AI USE STATEMENT

Generative AI tools assisted with manuscript drafting, language editing, and schematic-figure design. The authors selected the methods and experiments, verified all data, numbers, and citations, and take responsibility for the final scientific content.

REFERENCES

Sangho An, Jinwoo Kim, Keonho Lee, Jingang Huh, Chanwoong Kwak, Yujin Lee, Moonsub Jin, and Jangho Kim. Sparse structure exploration and re-optimization for vision transformer. In *Proceedings of the Forty-first Conference on Uncertainty in Artificial Intelligence*, volume 286 of *Proceedings of Machine Learning Research*, pp. 111–131, 2025. URL https://proceedings.mlr.press/v286/an25a.html.

Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18(153):1–43, 2018. URL https://jmlr.org/papers/v18/17-468.html.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. arXiv:1308.3432, 2013. URL https://arxiv.org/abs/1308.3432.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 2397–2430. PMLR, 2023. URL https://proceedings.mlr.press/v202/biderman23a.html.

Thomas Blumensath and Mike E. Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3):265–274, 2009. doi: 10.1016/j.acha.2009.04.002. URL https://doi.org/10.1016/j.acha.2009.04.002.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2943–2952. PMLR, 2020. URL https://proceedings.mlr.press/v119/evci20a.html.

Gongfan Fang, Hongxu Yin, Saurav Muralidharan, Greg Heinrich, Jeff Pool, Jan Kautz, Pavlo Molchanov, and Xinchao Wang. MaskLLM: Learnable semi-structured sparsity for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 7736–7758. Curran Associates, Inc., 2024. doi: 10.52202/079017-0248. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/0e9a05f5ce62284c91e4a33498899124-Paper-Conference.pdf.

Olivier Fercoq, Alexandre Gramfort, and Joseph Salmon. Mind the duality gap: safer rules for the lasso. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 333–342, Lille, France, 2015. PMLR. URL https://proceedings.mlr.press/v37/fercoq15.html.

Jonathan Frankle and Michael Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10323–10337. PMLR, 2023. URL https://proceedings.mlr.press/v202/frantar23a.html.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, volume 28, pp. 1135–1143. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/ae0eb3eed39d2bcef4622b2499a05fe6-Paper.pdf.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Yuezhou Hu, Jun Zhu, and Jianfei Chen. S-STE: Continuous pruning function for efficient 2:4 sparse pre-training. In *Advances in Neural Information Processing Systems*, volume 37, pp. 33756–33778. Curran Associates, Inc., 2024. doi: 10.52202/079017-1063. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/3b576711b12ab036b45130fc8eb78504-Paper-Conference.pdf.

Siddhant Jayakumar, Razvan Pascanu, Jack Rae, Simon Osindero, and Erich Elsen. Top-KAST: Top-k always sparse training. In *Advances in Neural Information Processing Systems*, volume 33, pp. 20744–20754. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/ee76626ee11ada502d5dbf1fb5aae4d2-Paper.pdf.

Jie Ji, Gen Li, Lu Yin, Minghai Qin, Geng Yuan, Linke Guo, Shiwei Liu, and Xiaolong Ma. Advancing dynamic sparse training by exploring optimization opportunities. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 21606–21619, 2024. URL https://proceedings.mlr.press/v235/ji24a.html.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. SNIP: Single-shot Network Pruning based on Connection Sensitivity. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1810.02340.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11976–11986, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 67–82, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pp. 8026–8037. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11893–11902, 2020.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115 (3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A Simple and Effective Pruning Approach for Large Language Models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2306.11695.

Hidenori Tanaka, Daniel Kunin, Daniel Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6377–6389. Curran Associates, Inc.,

2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/46a4378f835dc8040c8057beb6a2da52-Paper.pdf.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10347–10357. PMLR, 2021. URL https://proceedings.mlr.press/v139/touvron21a.html.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking Winning Tickets Before Training by Preserving Gradient Flow. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/2002.07376.

Chenyi Yang, Wenjie Nie, Yuxin Zhang, Yuhang Wu, Xiawu Zheng, Guannan Jiang, and Rongrong Ji. BAME: Block-aware mask evolution for efficient N:M sparse training. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 70943–70952, 2025. URL https://proceedings.mlr.press/v267/yang25v.html.

Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. Learning N:M fine-grained structured sparse neural networks from scratch. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2102.04010.

Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *Advances in Neural Information Processing Systems*, volume 32, pp. 3597–3607. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/1113d7a76ffceca1bb350bfe145467c6-Paper.pdf.

A Rank Certificates and Execution Guarantees

Objects and assumptions. Consider one group of n coordinates, a quota $0 \le k \le n$, finite anchors $a_i \in \mathbb{R}$, and finite radii $r_i \ge 0$. The admissible score box is $\mathcal{I}(a, r) = \prod_{i=1}^{n} [a_i - r_i, a_i + r_i]$. Magnitude Top-k orders coordinates by descending $|s_i|$ and then ascending original index i. Its ranking threshold τ is a scalar order statistic; at a tie, membership also depends on the index, not on a scalar comparison with τ alone. The empirical near-threshold band $\mathcal{B}_q$ contains the fraction q of scores closest in magnitude to the threshold. The boundary active set $\mathcal{A}$ is defined by the interval rank rules, while a network-level Backward Frontier fixes an initial network prefix. The rank rules are applied independently within each score group: a vision group is one tensor for per-tensor Top-k, and a structured 2:4 group has four coordinates and quota two.

The image of $[a_i - r_i, a_i + r_i]$ under absolute value has endpoints $\ell_i = \max(0, |a_i| - r_i)$ and $u_i = |a_i| + r_i$. Thus signed anchors and intervals crossing zero are covered without assuming nonnegative learned scores. Intervals describe allowed values; their Cartesian product gives the coordinatewise enclosure used by the certificate, including when actual updates are correlated.

Proof of Proposition 1. For any $s \in \mathcal{I}(a, r)$, write $x_i = |s_i|$, so that $\ell_i \le x_i \le u_i$. Every coordinate j preceding i in the magnitude/index order satisfies $u_j \ge x_j \ge x_i \ge \ell_i$. The number of predecessors is therefore at most P_i. If $P_i < k$, coordinate i must be selected. Conversely, each of the C_i coordinates with $\ell_j > u_i$ satisfies $x_j \ge \ell_j > u_i \ge x_i$ and strictly precedes i. If $C_i \ge k$, coordinate i cannot be selected. The two sets are disjoint: every certain strict outranker is also counted among the potential competitors, hence $C_i \le P_i$. The decisions in $\mathcal{F}^+$ and $\mathcal{F}^-$ hold for every admissible score vector. Their bits therefore agree for any two $s, t \in \mathcal{I}(a, r)$, which proves Equation 4. The certificate is interval-local: at the next refresh the anchors and envelope are recomputed, so a previously fixed coordinate may re-enter. The active set is conservative and can include coordinates that do not actually flip before that event.

The inequalities deliberately treat ties conservatively. The non-strict comparison in P_i includes a coordinate that could tie with i, even if its index would lose that tie. The strict comparison in C_i counts only guaranteed predecessors independent of index. This can leave stable coordinates in $\mathcal{A}$, including when $r_i = 0$ and magnitudes tie. The active set contains every coordinate whose membership is not certified by these interval inequalities.

Residual-quota identity and boundary cases. An admissible full mask selects exactly k coordinates, contains every member of $\mathcal{F}^+$, and contains none of $\mathcal{F}^-$. It must consequently select $k_{\mathcal{A}} = k - |\mathcal{F}^+|$ members of $\mathcal{A}$, with $0 \le k_{\mathcal{A}} \le |\mathcal{A}|$. Restricting the original magnitude/index order to $\mathcal{A}$ preserves their relative ranks. Selecting its first $k_{\mathcal{A}}$ coordinates therefore gives Equation 6, with no approximation of Top-k by independent thresholding. Packed indices are stored in ascending original order, so a stable sort implements the same tie rule.

The boundary refresh can resolve trivial quotas before applying the inequalities: for $k = 0$, all coordinates are fixed off; for $k = n$, all are fixed on. These cases also follow from the definitions of $\mathcal{F}^{\pm}$ in Section 3.1, since $C_i \ge 0$ and $P_i \le n - 1$. If $\mathcal{A} = \emptyset$, its residual quota is zero and the bitset is the complete mask. If $\mathcal{A} = [n]$, the residual operation is the original Top-k. More generally, residual quota zero or $|\mathcal{A}|$ selects none or all of the active group. Zero scores and signed zeros follow the same original-index tie rule. Deterministic 2:4 uses this argument independently on each group of four coordinates, with $k = 2$ and a separate residual quota. It is not global half-density Top-k.

What the certificate controls. Projection keeps scores inside the prescribed box throughout an event. The certificate therefore fixes membership for certified positions throughout that box, while fixed-on weights remain in the forward network and continue to contribute to active-score gradients. If two observed endpoints belong to the same box, their endpoint mask differences are supported on $\mathcal{A}$; re-anchoring creates the next certificate and the next active competition. The retrospective path envelopes used in the analysis quantify the recorded trajectories that motivate this eventwise schedule.

Remark 1 (Correctness of packed coordinate updates). For a fixed active set and projected interval, packed and full layouts with the same active-coordinate restriction give identical masks, active gradients, and SGD-momentum or AdamW updates when scores, anchors, optimizer states and counters, forward computation, STE, and projection agree.

Conditions for the packed-update remark. Let G gather a fixed active index set and $G^\top$ scatter it. Packed and full-layout states must have matching partitions, quotas, index ordering, active values, anchors, optimizer tensors, scalar step counters, and optimizer hyperparameters. They must use the same minibatch, frozen weights, other model state, loss, precision rules, and stochastic forward realization. The full-layout reference permits score updates only on the specified active coordinates; inactive scores remain at their admissible event values. In particular, zeroing inactive gradients alone is insufficient: momentum and weight decay can change a coordinate with a zero gradient.

Both STEs must obey $g_{\mathcal{A}}^{\text{packed}} = Gg^{\text{reference}}$. The pretrained hosts implement the identity surrogate through the complete magnitude-TopK operation, so no additional $\operatorname{sign}(s)$ factor is introduced in packed backward. A host using a different STE must establish the gather condition for that STE. Gradient accumulation, clipping, loss scaling, overflow decisions, and any other operation aggregating coordinates must act on the same active domain with the same settings. Score-dependent auxiliary losses, if present in a host, must likewise preserve this condition.

Verification of the packed-update remark. The residual-quota identity gives the same hard mask, hence the same logits and loss under the matching forward realization. The STE assumption then gives the same active gradients. Coordinatewise momentum SGD updates commute with gathering. For example, $v_i^+ = \mu v_i + g_i$ and $s_i^+ = s_i - \eta v_i^+$ produce the same values whether $i \in \mathcal{A}$ is stored densely or packed; matched dampening, Nesterov, or coordinatewise decay settings do not change this argument. AdamW's first and second moment updates also act coordinatewise, and matching step counters give identical bias-correction factors. Its active-coordinate weight decay and parameter update therefore commute with G. Finally, projection onto the coordinate intervals commutes with gathering:

$$G\Pi_{\mathcal{I}(a,r)}(s) = \Pi_{\mathcal{I}(Ga,Gr)}(Gs).$$

These facts prove equality of one mathematical update. Induction establishes the result through a fixed event interval. With matched numerical execution, the packed and full-layout implementations therefore follow the same update through that interval.

At each boundary-refresh event, the same check applies when the reconstructed logical scores, resulting active sets, and state-transfer policy match. The remark characterizes the event-local bounded active-coordinate optimizer; the empirical vision Full-score baseline without interval projection and the language Full-score baseline with matched bounds provide the corresponding quality references.

Reconstruction and re-entry policy. Pretrained hosts reconstruct never-modified scores from frozen weights using $s_i^0 = |w_i| / \operatorname{mean}(|w|)$, with a numerical floor for the denominator. Sparse score history records the most recently committed values of previously active coordinates; current active values override those entries before a boundary refresh. The eligible full logical scores, not just the previous active coordinates, determine the new envelope. A fixed coordinate can re-enter when the newly anchored intervals no longer exclude its competition. This does not require predicting that it will flip.

The vision host retains the current moments for coordinates active on both sides of an event and assigns zero moments to newly active or re-entering coordinates. It retains sparse score history but not dormant moment history. The language host retains sparse score, first-moment, and second-moment history. It restores the last stored moments on re-entry; dormant moments do not undergo intervening decay. Both hosts inherit the parameter-tensor Adam step counter, including for zero-initialized moments. These are distinct specified optimizers. A resumable controller checkpoint needs the current active scores, original indices, anchors, optimizer tensors and counters, fixed decisions, group quotas, event position, initializer/backbone identity, and the history required by its host policy. Exact training replay additionally requires the matching data stream, random state, and precision-control state.

Backward Frontier guarantee. Write the fixed-prefix loss as $G(h_c, u) = \mathcal{L}(f_{>c}(h_c; w_{>c} \odot u), y)$, where u denotes suffix mask multipliers and derivatives are evaluated at their current binary values. The representation h_c includes every tensor crossing the graph boundary. It has no dependence on suffix trainable variables, including through parameters shared across the cut. Thus $\partial h_c / \partial u = 0$, and detaching h_c removes no term from the chain-rule expression for $\nabla_u G$. Applying the same host STE to these matching mask gradients produces the same surrogate gradients for

suffix scores. The compared forward executions retain the same prefix masks, normalization state, stochastic realization, and cross-boundary tensors. The entire forward computation still executes. Folding a permanent prefix removes its controller and may discard its now-unneeded history; suffix gradients follow the same chain-rule expression under the fixed-prefix protocol. This chain-rule statement is independent of packed score storage and does not assert that freezing the prefix has zero quality cost relative to an unfrozen model.

Storage and boundary-refresh accounting. The $O(A + H)$ bound counts persistent scalar controller entries and indices, where H denotes historical coordinates outside the current active set so that overlap is not double-counted; $O(N)$ fixed-decision bits and group metadata are additional. History can extend to the full coordinate count before removing overlap with the active set. The coefficients in Equation 9 depend on whether dormant moments are retained. The approximate $16N$ FP32 Adam baseline counts one score, one gradient, and two moments per coordinate. Anchors belong to the logical controller and are included in the packed vision-controller totals; backbone weights, activations, allocator overhead, and temporary arrays are reported as separate regions.

Vision logical tensor accounting charges the serialized bitset size, while the language live-controller measurement includes expanded buffers and retained history. Reconstruction can temporarily materialize full score/moment arrays on the host or device, and interval sorting uses additional refresh workspace; these costs are included in peak memory and total runtime. Sorting per-group endpoints has cost $O(\sum_g n_g \log n_g)$; constant-size 2:4 comparisons have cost $O(N)$. The average $T_{\mathcal{A},c}+C/J$ includes boundary-refresh cost C amortized over J updates; whole-run measurements additionally include their stated setup, loading, and preprocessing regions.

B Implementation and Experimental Details

Score representation and stored state. Scores and optimizer moments are FP32. Active indices use int32 in the packed representation and are promoted when a gather/scatter kernel requires it. Fixed decisions admit bit-packed serialization. The vision and language adapters also materialize expanded masks and execution buffers for dense kernels. Vision's logical controller metric includes scores, gradients, moments, indices, anchors, a logical bitset, sparse score history, and the associated quota and shape metadata. The language-system metric includes live buffers and retained history. CUDA allocation, serialized controller size, and logical controller storage are reported as separate quantities.

Boundary refresh and optimizer policy. Pretrained hosts initialize each tensor as $s_i^0 = |w_i|/\operatorname{mean}(|w|)$. At an event, active values are committed to sparse history; never-modified scores are recovered from this initializer. Vision retains sparse score history on the host and transfers moments for coordinates active immediately before and after the event. New or re-entering coordinates receive zero moments. The language host restores sparse score and moment history on re-entry; inactive moments remain unchanged, without intervening decay. Both hosts inherit the parameter-tensor Adam step counter, including for coordinates whose moments reset. Re-anchoring precedes the next projected interval. Prefix coordinates permanently fixed by a frontier are excluded from subsequent reconstruction. Temporary full-layout score, endpoint, and optimizer arrays used at events contribute to compilation time and peak memory. Recoverable state therefore includes initialization metadata, active scores, history, optimizer policy/state, anchors, quotas, and the event position in addition to the fixed bits.

ImageNet quality protocol. The vision host uses PyTorch (Paszke et al., 2019). Scores are trained for one epoch with AdamW (Loshchilov & Hutter, 2019) (learning rate 3×10^{-4}, zero weight decay), FP32 scores, and mixed-precision forward computation. Fifty training images per class are reserved for development, leaving 1,231,167 update images. One epoch has 4,810 batches at batch size 256; seeds are 17/29/43. The original Full-score and frontier runs share initialization, data order, augmentations, and the update budget. The supplementary Hard-freeze and Boundary-only runs use the same weights, initial masks, splits, and training recipe in a separate runner. They provide supplementary quality and A40 cost records. Hard-freeze fixes masks after 2,400 updates and skips the unused tail; Boundary-only performs all 4,810 updates. Training uses random resized 224×224 crops, horizontal flips, and ImageNet normalization. Pretrained model and normalization

state remain in evaluation mode; classifier weights are dense and frozen. Official validation uses the pretrained recipe's architecture-specific transforms on all 50,000 images. Eventwise Boundary recompiles every 160 steps with radius 0.06. The combined method fixes through stage 1 for the CNNs or block 3 for DeiT-B at the event nearest the midpoint. Each architecture uses the same cut for all seeds.

Reference implementations and mechanism controls. Remark 1 uses an *active-restricted full-layout reference*, including identical intervals, projection, and optimizer-domain rules. The vision quality baseline instead updates every score without interval projection. The language quality baseline updates all scores under matching bounds and anchor resets. Fixed-band selects an initial distance band; Static Boundary uses the initial interval certificate and anchors throughout training. Eventwise Boundary refreshes both the active set and its anchors. Random-active and Stability-active use the selected/excluded quotas of a hypothetical boundary computed from their own current scores at each event. Random-active samples these competitors; Stability-active uses eight bounded perturbation probes initially and subsequent absolute score changes as its stability signal. Together, these controls compare initial bands, refreshed certificates, and alternative competitor-selection policies.

Resource regions and aggregation. Accuracy uses the mean and sample standard deviation over seeds. Main tables report storage and peak savings as $100(1 - R)$ and speedups as $1/R$, where R is the candidate/Full-score cost ratio. ImageNet resources come from the dedicated one-epoch profiling group, independently of the checkpoint- producing official-quality runs. Logical controller storage is the final trainable state, including gradients and score history, over three seeds. Timing and peak ratios pair matching GPU hosts and seeds: 29/43 for ResNet-50 and 17/29/43 for ConvNeXt-T and DeiT-B; speedup is the reciprocal of the mean paired ratio. Table 3 retains the underlying ratios.

Vision training-region time includes initial controller setup, data delivery, updates, and event compilation, and excludes model construction, final checkpoint serialization, and evaluation. The training allocation peak is reset after initial setup; median-step timing excludes event compilation and data-loader waiting; controller checkpoints exclude the shared frozen backbone. These regions distinguish controller compression from dense execution cost.

Supplementary A40 costs. Table 14 reports the existing Hard-freeze and Boundary-only records. A40 loop time includes data delivery and event compilation, but excludes initial setup and checkpoint serialization. The peak is measured during learning, before Hard-freeze discards its optimizer. The Hard-freeze row counts its full-score learning controller up to the stopping point; Boundary-only reports its final logical trainable state including history.

Language quality protocol. Pythia-1.4B uses frozen BF16 weights, FP32 scores, and deterministic Top-2 within each eligible four-weight group in attention and MLP linear layers. A fixed pool contains 4,096 length-2,048 samples from the first English C4 training shard, selected with seed 20260918. Seeds 17/29/43 each perform 1,024 batch-size-one updates, consuming 2,097,152 input tokens per run. AdamW uses learning rate 3×10^{-4}, betas $(0.9, 0.999)$, and zero weight decay. Both methods project scores within radius 0.03 and reset anchors at steps 256/512/768; paired methods share the sample stream. The final mask is fixed before independent evaluation and zero-shot export. Full WikiText-2 validation joins all rows with two newlines and no special tokens. Context 2,048 and stride 1,024 cover all 252,480 shifted targets exactly once, masking previously scored overlap and including the final partial window. All reported language quality uses WikiText-2 validation; the test split is withheld.

Language resource measurement. Table 4 reports quality from the three-seed C4 runs and resources from the five-repeat same-host primary timing. Total controller storage includes GPU buffers and retained CPU score/moment history. The reported with-data wall region includes in-memory token slicing and host-to-device transfer, but excludes parquet I/O and tokenization; paired cost ratios are arithmetic means of Boundary/Full-score ratios over the five alternating-GPU repeats. Savings equal one minus the mean ratio; speedup is the reciprocal of the mean time ratio. Inference masks have essentially identical sizes. The allocator peak covers the live buffers observed during the timed loop and its eventwise refreshes. A separate 10-warmup/40-step profile and per-seed records are provided below.

Table 7: Per-seed official ImageNet-1K top-1 accuracy (%) under the nominal one-epoch budget. Full-score and Boundary + frontier use A800 runs; Hard-freeze and Boundary-only use A40 runs with the same splits, weights, and training recipe.

Backbone	Method	Seed 17	Seed 29	Seed 43
ResNet-50	Hard-freeze	79.552	79.372	79.518
ResNet-50	Full-score	79.682	79.738	79.734
ResNet-50	Boundary-only	79.712	79.784	79.640
ResNet-50	Boundary + frontier	79.680	79.728	79.724
ConvNeXt-T	Hard-freeze	80.628	80.590	80.578
ConvNeXt-T	Full-score	80.970	80.836	81.002
ConvNeXt-T	Boundary-only	80.512	80.580	80.662
ConvNeXt-T	Boundary + frontier	80.594	80.538	80.530
DeiT-B	Hard-freeze	80.128	80.064	80.152
DeiT-B	Full-score	80.184	80.264	80.158
DeiT-B	Boundary-only	80.224	80.228	80.168
DeiT-B	Boundary + frontier	80.202	80.232	80.172

Component timing profile. The local Pythia profile executes 10 warm-up and 40 subsequent steps. Component means include all 50 steps; the step median uses only the final 40. The forward region includes gradient clearing, mask construction, and the backbone forward pass. Mean per-step times for Full-score versus Boundary are 1.46318 versus 0.08463 s for the forward path, 0.04112 versus 0.03256 s for backward, 0.05759 versus 0.00661 s for the optimizer, and 0.02346 versus 0.00367 s for score projection. The median step is 1.58279 versus 0.118807 s. Boundary compilation takes 9.402 s in this profile; model-and-optimizer checkpoint probes take 23.387 versus 5.318 s. This diagnostic locates execution costs; the five-repeat benchmark provides the headline speedup.

External structured-sparsity context. Table 8 compares five methods on Pythia-1.4B with 2:4 masks over the same 96 matrices. The external methods use our Hugging Face/GPT-NeoX adaptations: Wanda and SparseGPT use 128 C4 calibration sequences, while MaskLLM, Full-score, and Boundary use 1024 batch-one updates with context length 2048. All rows use seed 17 and the same full WikiText-2 validation evaluation (context 2048, stride 1024, and 252,480 predicted tokens). SparseGPT retains its compensated weights; MaskLLM uses its exported hard argmax masks. Time excludes checkpoint I/O and covers calibration and pruning for one-shot methods or the training loop, including in-loop recompilation, for learned masks. Peak memory is CUDA allocated memory in decimal GB, including controller setup for Full-score and Boundary. Boundary retains Full-score's language-model quality with substantially lower training cost; the external rows show the quality–cost trade-off at their stated budgets. Optimizer-backend measurements appear separately in Table 9.

Table 8: Pythia-1.4B with 2:4 sparsity (seed 17). All methods use the same WikiText-2 validation evaluation. Budgets are calibration samples or training updates; time and allocated peak memory cover mask construction or training.

Method	Budget	PPL ↓	Time (s)	Peak (GB)
Wanda	128 samples	498.674	28.89	5.31
SparseGPT	128 samples	33.466	166.50	6.13
MaskLLM	1024 updates	68.433	409.63	39.09
Full-score	1024 updates	20.650	1706.30	31.84
Boundary	1024 updates	20.719	213.92	14.21

Table 10: Development sweep over radius r, refresh interval J, and prefix cut. Entries are clip-rates for seeds 17 and 29.

r	J	cut	seed 17	seed 29
0.03	160	1	1.1833e−3	1.2424e−3
0.06	80	1	0	0
0.06	160	1	4.0237e−7	4.5587e−7
0.06	160	2	4.2131e−7	4.7407e−7
0.06	160	3	4.7643e−7	5.5735e−7
0.06	320	1	4.3804e−5	4.7932e−5
0.12	160	1	0	0

Table 11: **Matched development controls for the active-state comparison.** Mean training cross-entropy, active fraction, and persistent controller bytes over seeds 17/29/43. These follow-up rows use the same one-epoch host protocol but remain development measurements; no official validation or test split was opened for these controls.

Model	Method	Mean CE↓	Active % ↓	Controller bytes↓
ConvNeXt-T	Projected Full-score	0.711978	100.000	669,613,732
ConvNeXt-T	Full-score + prefix	0.714612	99.136	669,613,732
ConvNeXt-T	Nearest-count	0.724574	10.567	110,330,412
ConvNeXt-T	Nearest-storage	0.717440	10.793	122,176,980
DeiT-B	Projected Full-score	0.481696	100.000	2,063,278,276
DeiT-B	Full-score + prefix	0.482325	74.483	2,063,278,276
DeiT-B	Nearest-count	0.483342	10.783	340,829,236
DeiT-B	Nearest-storage	0.482602	10.910	374,349,292
ResNet-50	Nearest-storage	0.776449	11.224	106,370,996

Table 9: Optimizer-backend comparison on the Pythia controller. All rows use 1,024 updates with seed 17; controller checkpoint bytes exclude the frozen backbone.

Backend	Median step (s)	Wall (s)	Optimizer state (GB)	Controller ckpt. (GB)
Adafactor	1.6823	1749.29	14.50	9.67
8-bit AdamW (compat.)	1.7984	1876.43	16.91	12.08

Sensitivity and implementation checks. The development sweep covers 14 frozen combinations of radius, refresh interval, and prefix cut. Across the sweep, mean, median, and maximum clip-rates are 1.8001×10^{-4}, 4.6497×10^{-7}, and 1.2424×10^{-3}. For the default $(r, J, \mathrm{cut}) = (0.06, 160, 1)$, the two seed rates are 4.0237×10^{-7} and 4.5587×10^{-7} (Table 10). The matched dense-layout check uses six steps with recompilation every two steps on a frozen ResNet-50 2:4 layer. Packed and active-restricted dense layouts agree step by step for masks, logits, loss, active gradients, updates, re-entry, prefix behavior, and cold recovery under AdamW and SGD; SGD is exact in the reported cold-state comparison and AdamW is allclose.

Longer-training ImageNet. The extended comparison uses 30 epochs, or 144,300 updates, with the same frozen weights, data split, batch size, optimizer, and seeds 17/29/43. Boundary recompiles at epochs 5/10/15/20/25 and applies the frontier at epoch 15, retaining the architecture-specific cut. Independent evaluation uses the fixed final epoch-30 checkpoint and all 50,000 official validation images. Table 12 reports all three paired seeds for each architecture; every pair improves. Final logical controller storage, including retained history, decreases by 80.3%, 81.3%, and 85.6% on ResNet-50, ConvNeXt-T, and DeiT-B.

CIFAR-10 transfer. The Edge-Popup-style Conv4 host uses 45,000 training images and 5,000 development images. The official 10,000-image test comparison uses final epoch-100 checkpoints without reselection. Seeds 17/29/43/59 give test accuracies 83.68/83.45/83.66/83.51 for Full-score, 83.90/83.15/83.66/83.47 for Boundary-only, and 82.84/83.00/82.73/83.07 for Hard-freeze. The ap-

Table 12: **ImageNet-1K quality after 30 epochs (144,300 updates).** Top-1 accuracy on all 50,000 validation images, mean ± sample SD over seeds 17/29/43 using fixed final checkpoints. Δ is the paired gain in percentage points.

Backbone	Full-score	Boundary + frontier	Δ (pp)
ResNet-50	79.505 ± 0.029	79.856 ± 0.041	$+0.351$
ConvNeXt-T	80.671 ± 0.063	80.835 ± 0.040	$+0.164$
DeiT-B	79.568 ± 0.091	80.057 ± 0.028	$+0.489$

Table 13: **CIFAR-10 official-test accuracy and continuation cost.** Accuracy is four-seed mean ± sample SD; costs cover epochs 81–100 after epoch-80 compilation and exclude gradients from controller state.

Method	Top-1 (%)↑	Ctrl saving (%)↑	Continuation speedup (×)↑
Full-score	83.575 ± 0.113	0.0	1.00
Hard-freeze	82.910 ± 0.154	–	–
Boundary-only	83.545 ± 0.317	73.4	2.00

proximately $2.00\times$ speedup in Table 13 is the reciprocal of the mean paired continuation-time ratio, 0.500943, over epochs 81–100. The 73.4% resident-controller saving counts scores, momentum, active indices, anchors, radii, and fixed bits, excluding transient gradients, using the transfer host's accounting rather than the vision Adam controller definition. In a separate development frontier ablation (seeds 43/59), Full-score, Boundary-only, and Boundary + frontier reach 83.97%, 84.12%, and 83.99%; the combined method uses 0.238 of the controller state and 0.171 of the epoch-80–100 continuation time.

C Boundary dynamics and correctness diagnostics

This section reports three complementary diagnostics: (a) the retrospective path envelope, (b) endpoint-change recall near the threshold, and (c) the fixed-prefix execution cost. The retrospective group radius $r_G = \max_{i \in G} \sum_{t \in \text{epoch}} |s_i^{t+1} - s_i^t|$ describes coordinate motion over an observed epoch, including intermediate reversals. Figure 2(a) reports the resulting uncertain fraction with all coordinates as denominator. Panel (b) measures endpoint- change recall for an empirical near-threshold band, while panel (c) measures step time, peak allocated memory, and development accuracy for a fixed prefix. The envelope uses the full observed path, whereas endpoint recall uses window endpoints.

For a single interval extending to epoch 100, the retrospective coordinatewise path envelope is essentially full at epochs 60/70, approximately 82% at epoch 80, and approximately 14% at epoch 90. Its horizon and coordinatewise radii characterize boundary dynamics at this longer interval and differ from the common-radius, one-epoch construction.

Correctness coverage. Existing checks cover stable ties, signed and zero scores, empty and full active sets, per-tensor and 2:4 quotas, state packing, optimizer alignment, one-step comparisons, and fixed-prefix suffix-gradient checks.

D Resource Comparison

Table 14 provides supplementary A40 costs; the main ImageNet profiling table is Table 3. Figure 4 separates state allocation from execution; ratios below one denote lower costs. Pythia uses its C4 controller and allocation records, not the vision timing protocol.

Same-GPU short systems profile. To isolate execution effects from host differences, we ran Full-score and Boundary + frontier sequentially on the same A800 GPU with seed 43 for each backbone, using 500 updates, batch size 256, and the fixed ImageNet streams. The primary stable-step statistic

Table 14: **Supplementary A40 training costs.** Six runs per backbone (two methods, three seeds). Time covers training and event compilation; peak is training-region PyTorch allocation. †Hard-freeze reports analytical dense learning state, while Boundary-only reports final logical state including history.

Method	Seed	Actual updates	Training time (s)	Training peak (GiB)	Controller storage (MiB)
ResNet-50					
Hard-freeze†	17	2,400	700.6	11.363	357.89
	29	2,400	701.6	11.363	357.89
	43	2,400	700.3	11.363	357.89
Boundary-only	17	4,810	1,434.1	11.349	84.89
	29	4,810	1,437.5	11.349	84.87
	43	4,810	1,431.4	11.349	84.87
ConvNeXt-T					
Hard-freeze†	17	2,400	771.4	14.225	423.52
	29	2,400	773.7	14.225	423.52
	43	2,400	771.8	14.225	423.52
Boundary-only	17	4,810	1,674.8	14.014	96.91
	29	4,810	1,575.0	14.014	96.94
	43	4,810	1,568.7	14.014	96.94
DeiT-B					
Hard-freeze†	17	2,400	1,556.5	25.765	1,305.00
	29	2,400	1,548.6	25.765	1,305.00
	43	2,400	1,554.3	25.765	1,305.00
Boundary-only	17	4,810	3,057.4	25.163	301.77
	29	4,810	3,043.7	25.163	301.79
	43	4,810	3,057.4	25.163	301.80

Table 15: **Same-GPU 500-update ImageNet profile (seed 43).** Full-score and Boundary + frontier use identical batches on one A800. Stable-step speedup is the Full-score/Boundary median ratio over updates 301–500; loop speedup includes midpoint recompilation.

Backbone	Stable step speedup ↑	Loop speedup ↑	Allocated peak saving (%) ↑	Controller saving (%) ↑	Initial compile (s)
ResNet-50	1.236×	1.018×	0.158	74.750	0.218
ConvNeXt-T	1.243×	1.036×	0.170	75.483	0.233
DeiT-B	1.215×	1.046×	2.339	81.332	0.211

is the median over updates 301–500; the first 50 updates are warm-up and updates 251–300 stabilize after the midpoint frontier event. Boundary improves the stable step rate by 1.215–1.243× and reduces logical controller state by 74.8–81.3% (Table 15). The loop statistic includes a per-batch SHA-256 diagnostic and the midpoint recompilation, so it is reported as an instrumented profile. The allocated peak covers the full 500-update loop and is separate from controller storage.

E Additional Language-Model Results

Zero-shot metrics. The native lm-evaluation-harness (v0.4.9.2) evaluates the frozen models on LAMBADA, HellaSwag, PIQA, ARC-Easy, ARC-Challenge, and WinoGrande with zero demonstrations and no sample limit. Each model covers 21,848 examples. Table 16 uses raw `acc` for every task, including tasks that also provide length-normalized `acc_norm`. The mean gives each of the six tasks equal weight; trained models then average seeds 17/29/43. Dense and magnitude masks are deterministic references. The alternative normalized convention is shown separately.

Separate short-context timing workload. The completed standalone workload uses WikiText-2 training tokens, context 128, 100 warm-up steps, and 500 timed steps with five recompilations. Five paired processes ran serially on one A800 host, with method order alternated. It measures

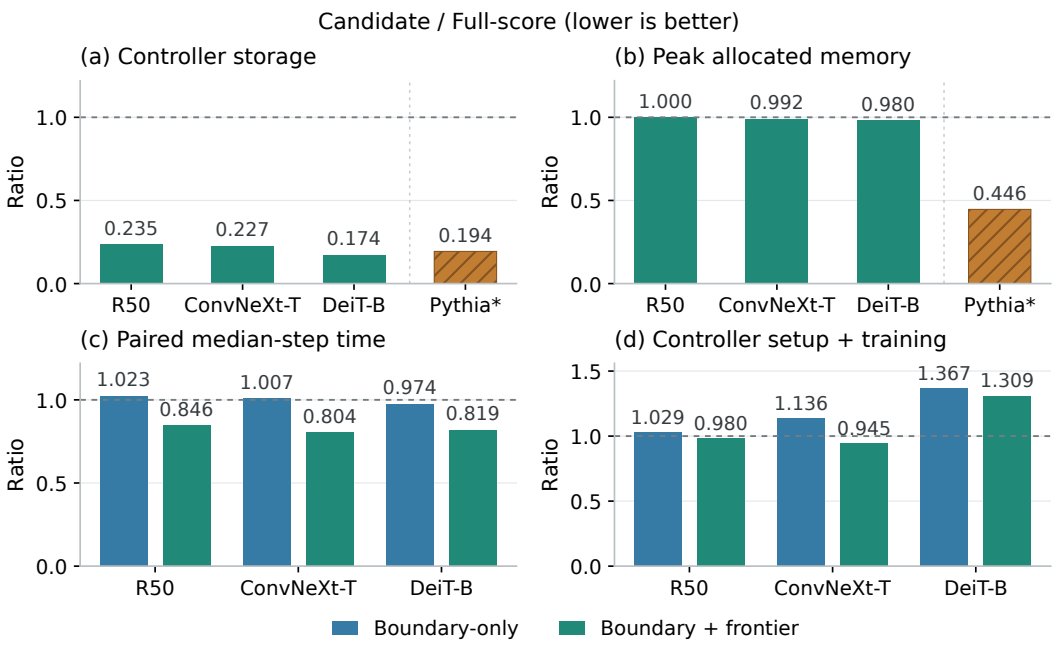

Figure 4: **Separate costs of mask optimization.** Candidate/Full-score ratios (↓) for vision Boundary + frontier and Pythia Eventwise Boundary. Panels (a,b) show controller storage and peak allocation; (c,d) show vision median-step and setup-plus-training time. Vision storage is logical tensor state; Pythia* denotes the C4/context-2048 profile, while the short-context five-repeat timing is reported separately in Table 4. Dashed line: Full-score.

Table 16: **Pythia-1.4B zero-shot results from lm-evaluation-harness 0.4.9.2.** Percentages use 0-shot prompts and 21,848 examples per model. Full-score and Boundary average seeds 17/29/43; deterministic references are single runs. The main panel uses equal-weight raw `acc`; the lower panel uses `acc_norm` where available and raw `acc` otherwise.

Method	LAMBADA OpenAI	HellaSwag	PIQA	ARC-Easy	ARC-Ch.	WinoGrande	Mean
Examples	5,153	10,042	1,838	2,376	1,172	1,267	–
Main metric: raw `acc` on all six tasks							
Dense pretrained	61.19	40.35	71.11	60.61	25.68	56.99	52.65
Magnitude 2:4	1.75	27.58	58.27	35.06	19.28	51.07	32.17
Full-score	39.15	33.06	65.25	48.36	20.71	53.22	43.29
Boundary	38.88	33.10	65.27	48.08	20.62	53.75	43.28
Supplementary metric: `acc_norm` where defined, `acc` otherwise							
Dense pretrained	61.19	51.95	71.06	54.42	29.18	56.99	54.13
Magnitude 2:4	1.75	30.59	56.75	36.66	22.53	51.07	33.22
Full-score	39.15	40.23	65.20	43.36	25.28	53.22	44.41
Boundary	38.88	40.19	65.40	43.36	25.23	53.75	44.47

in-memory batch construction, transfer, updates, and recompilation, excluding model loading and tokenization. Arithmetic mean candidate/Full-score ratios are 0.1943 for controller state, 0.3289 for peak allocated memory, 0.3929 for peak reserved memory, and 0.1087 for timed-loop cost. Compilation takes 24.79% of candidate loop time. This workload supplies the resource columns of Table 4; its quality columns come from the separate three-seed C4 runs. The C4/context-2,048 run resources appear in Table 17.

Table 17: **Per-seed Pythia-1.4B C4 quality and resources.** Learned-mask runs use 1,024 updates at context 2,048. PPL uses WikiText-2 validation; raw zero-shot accuracy is the equal-weight six-task mean. Controller and checkpoint columns report recoverable training state. The per-seed GPU peak and time profile includes compilation and checkpoint I/O; aggregate short-context timing is reported separately in Table 4. Compile is the time share; storage is in GiB.

Method	Seed	PPL	Raw acc (%)	Controller (GiB)	Checkpoint (GiB)	GPU peak (GiB)	Time (s)	Compile (%)
Full-score	17	20.650	43.614	22.500	18.000	29.651	1941.51	0.001
Boundary	17	20.719	43.602	4.351	2.664	13.232	244.94	5.608
Full-score	29	20.774	43.044	22.500	18.000	29.651	1936.99	0.001
Boundary	29	20.867	42.884	4.356	2.668	13.232	240.15	6.233
Full-score	43	20.499	43.216	22.500	18.000	29.651	1935.87	0.001
Boundary	43	20.516	43.364	4.358	2.669	13.232	246.12	6.647