

Why Lagrangian Pruning Fails in Mixture-of-Experts: Gradient-Scale Mismatch and a Schedule-Based Fix

Karim Magdy

karimagdy22@gmail.com

Independent Researcher

Ghada Khoriba

Capital University

Mohamed N. Swailam

Università degli Studi di Milano-Bicocca

Hala Abbas

Arab Open University

Article

Keywords:

Posted Date: June 23rd, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-9681451/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Why Lagrangian Pruning Fails in Mixture-of-Experts: Gradient-Scale Mismatch and a Schedule-Based Fix

Karim Magdy^{5,✉}, Ghada Khoriba^{1,2}, Mohamed N. Swailam⁴, and Hala Abbas³

¹Computer Science Department, Faculty of Computers and Artificial Intelligence, Capital University, Helwan, Egypt

²School of Information Technology and Computer Science (ITCS), Nile University, Giza, Egypt

³Faculty of Computer Studies, Arab Open University, Cairo, Egypt

⁴Department of Informatics, Systems and Communication, Università degli Studi di Milano-Bicocca, Milano, Italy

⁵Independent Researcher, Cairo, Egypt

✉Corresponding author: karimagdy22@gmail.com

Abstract

We identify a gradient-scale failure mode of per-neuron Lagrangian FLOPs pruning in Mixture-of-Experts (MoE) Vision Transformers. In an elastic four-block MoE ViT, each neuron contributes roughly 10^{-6} of the compute budget, so the penalty gradient on a mask logit is of order 10^{-4} , two orders of magnitude weaker than the task gradient. The optimiser satisfies the budget by collapsing all logits uniformly rather than selecting neurons—a phenomenon we call uniform mask collapse. Empirically, the Lagrangian baseline reaches only 36.4%, 10.4%, and 4.7% on CIFAR-10, CIFAR-100, and Tiny-ImageNet despite hitting the FLOPs target, while every schedule-based method survives. As a practical remedy we propose KaleidoNet, an instantiation that replaces the Lagrangian penalty with a deterministic cubic sparsity schedule, irreversible gradient masking, and dual-rate optimisation. Trained to convergence (50,000 steps, three seeds), KaleidoNet achieves a $1.80\times$ active-FLOPs reduction and retains 98.3–99.2% of dense accuracy, outperforming every schedule-based pruning baseline. Post-training surgery yields a $1.63\times$ parameter compression. Inference latency at this scale is dominated by MoE routing overhead, a known limitation that shrinks at larger scale. Validation covers CIFAR-10/100 and Tiny-ImageNet on a 5.49M-parameter MoE ViT; ImageNet-1K with larger backbones is future work.

Introduction

Modern vision models increasingly rely on two architectural ideas that trade raw parameter counts against conditional computation. *Vision Transformers* (ViTs) [9] treat an image as a sequence of patches and apply self-attention over them. *Mixture-of-Experts* (MoE) layers [11, 38, 39] replace a standard feed-forward block with a bank of small sub-networks, called experts, and a gating mechanism that sends each token to a small subset of them (often only one). Together these ideas produce models whose *total* parameter count is large but whose *active* parameter count per input is small. This conditional computation is a cornerstone of state-of-the-art language and vision models.

Even with MoE, the per-token active compute can be reduced further by removing neurons *inside* each expert. This is the domain of structured pruning [5, 14, 24, 29, 35, 42]. A widely adopted approach adds a *Lagrangian FLOPs penalty* to the training objective:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \lambda(\text{FLOPs}(\theta) - B), \quad (1)$$

where B is a compute budget and λ is a Lagrange multiplier that can be learned or adapted online [13, 22]. When each candidate structure carries a differentiable mask, gradient descent is expected to simultaneously minimise task loss and respect the budget, yielding a selective subset of important components. This paradigm works well when the granularity of decisions is coarse—e.g. pruning whole layers or whole experts, where each decision controls $\sim 10\%$ of total FLOPs.

In this paper we study what happens when the same recipe is pushed to the natural next level of granularity: pruning individual neurons *within* MoE experts. The intuition is that experts are over-parameterised and that many of their hidden units are redundant for a given task. Empirically, however, the Lagrangian recipe fails catastrophically in this setting. A properly tuned Lagrangian baseline that reaches the target FLOPs ratio converges to accuracy barely above random chance.

The contributions of this work are organised as follows.

Primary contribution—failure-mode diagnosis. We show that this collapse is a direct consequence of a gradient-scale mismatch. Per-neuron FLOPs contributions are six orders of magnitude smaller than the total budget, so the penalty gradient on a mask logit is roughly 10^{-4} , while the task-loss gradient on the same logit is $\sim 10^{-2}$. The optimiser therefore cannot use the penalty to distinguish useful neurons from useless ones: the only feasible way to reduce the penalty term is to drive *all* logits uniformly toward zero, a phenomenon we call *uniform mask collapse*. This analysis is the central novelty of the paper.

Secondary contribution—empirical validation across baselines. We confirm the prediction empirically on three datasets under a 50,000-step convergence budget: the Lagrangian baseline collapses on every dataset despite reaching the FLOPs target, while *every* schedule-based pruning method (cubic, linear, random, magnitude) bypasses the gradient-scale issue and reaches near-dense accuracy. The defining axis is therefore not the specific schedule shape but whether the structural decision is taken outside the gradient-scale-mismatched optimisation.

Tertiary contribution—KaleidoNet as a stable practical instantiation. We propose **KaleidoNet**, an elastic MoE ViT that replaces the Lagrangian penalty with a deterministic *cubic sparsity schedule* [46] combined with two mechanisms that further stabilise selective pruning. First, *gradient masking*: once a neuron has been removed by the schedule, its mask logit is clamped to a large negative value and its gradient is zeroed, so pruning decisions are irreversible. Second, *dual-rate optimisation*: mask logits are updated with a learning rate three times larger than that of the task weights, so structural decisions track the schedule faster than representations drift. Together with a Gumbel-sigmoid reparameterisation and straight-through estimator [1, 8, 18, 34], these choices let a simple importance heuristic (the mask logit magnitude) drive selective pruning. The approach conceptually inherits from earlier work on learnable gated networks and structured regularisation [31] and on elastic architectures [4, 17, 43]; see [6] for a contemporary survey. KaleidoNet is the best instantiation we tested, but the diagnostic insight is the generalisable contribution.

We evaluate the diagnosis and the fix on CIFAR-10, CIFAR-100, and Tiny-ImageNet using a small but controlled four-block MoE ViT (5.49M parameters). Under a shared 50,000-step convergence budget with three seeds, the Lagrangian baseline collapses in the predicted way (all mask logits drift uniformly toward zero), while KaleidoNet’s schedule-based approach maintains selective pruning and achieves a $1.80\times$ reduction in active FLOPs while retaining 98.3–99.2% of dense-model accuracy (Table 1 and Fig. 2). KaleidoNet attains $82.0 \pm 0.4\%$ on CIFAR-10, $59.1 \pm 0.2\%$ on CIFAR-100, and $40.1 \pm 0.3\%$ on Tiny-ImageNet, outperforming all schedule-based pruning baselines (cubic > linear > random > magnitude, with a 0.2–0.7 pp margin over the next best method on each dataset). The residual gap to the dense reference is small (0.5–1.1 pp) and reflects the unavoidable cost of $1.80\times$ FLOPs compression. Post-training model surgery physically removes pruned neurons and produces a compact network with a $1.63\times$ parameter reduction. We frame these results as contributions to FLOPs reduction and parameter compression rather than to wall-clock acceleration: MoE routing overhead dominates wall-clock inference at this scale, so KaleidoNet is $\sim 18\times$ slower than the dense ViT on Apple MPS—a documented platform-specific implementation limitation rather than a claim of this work, expected to shrink substantially at larger model scales.

The honest takeaway is that the Lagrangian–MoE pruning recipe, which has been assumed to scale to any granularity, in fact breaks at per-neuron level inside experts. Schedule-based pruning with gradient masking is a simple, stable alternative, and the analysis extends naturally to other architectures where decisions are made at a granularity too fine for budget gradients to be informative.

Results

Experimental setup

All experiments use a small but controlled four-block MoE ViT with embedding dimension $d=192$, six attention heads, four MoE experts with top-1 routing, and per-neuron elastic layers in each expert. Patch sizes are 4 for CIFAR-10/100 (32×32 images) and 8 for Tiny-ImageNet [21] (64×64 images downsampled to 32×32); we follow the data-efficient ViT training recipe of [40]. The dense baseline uses the identical

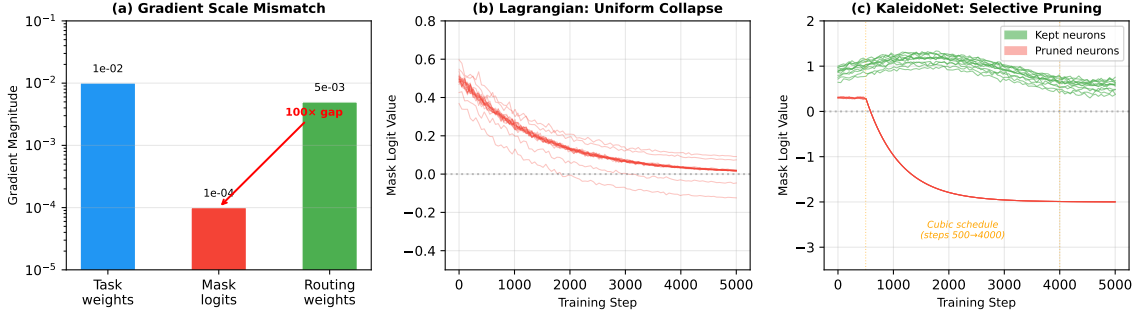


Figure 1: **Gradient-scale mismatch in per-neuron Lagrangian pruning.** (a) Gradient magnitudes for different parameter groups on an elastic MoE ViT: mask logits receive penalty gradients that are roughly $100\times$ smaller than the task gradients on the same variables. (b) Under a Lagrangian penalty with a shared learning rate, all mask logits drift uniformly toward zero, yielding non-selective pruning. (c) KaleidoNet’s cubic schedule combined with dual-rate optimisation produces clear separation between kept and removed neurons.

backbone without MoE or elastic layers (1.82M parameters); KaleidoNet totals 5.49M parameters (4 experts $\times$ 2-layer FFN) with 2.18M active under top-1 routing. We train each model with batch size 64 for $T=50,000$ steps (convergence budget), using AdamW [30] for task parameters at learning rate 3×10^{-4} with cosine schedule (500-step warmup), Adam for mask logits at learning rate 9×10^{-4} , gradient clipping at norm 1.0, and three independent random seeds per configuration. The cubic sparsity schedule ramps from step 500 to 4,000 with target sparsity 70%, leaving a long post-schedule recovery phase for the task parameters to re-adapt to the pruned architecture. Unless otherwise noted, all quantities report mean $\pm$ standard deviation across three seeds.

Apples-to-apples baseline configuration. All methods in Table 1 (KaleidoNet, Linear, Random, Magnitude, Lagrangian) use the *identical* early-exit confidence-head architecture, ponder regulariser $\lambda_{\text{ponder}}=0.05$, exit threshold $\theta=0.85$, and MoE balance-loss coefficient $\beta_{\text{bal}}=0.01$; the only axis that varies between rows is the per-neuron mask-training rule (cubic vs. linear vs. random vs. magnitude vs. Lagrangian). This rules out the concern that the Lagrangian baseline’s collapse could be attributed to a mis-configured early-exit head or balance-loss coupling rather than to the gradient-scale mismatch we diagnose.

The gradient-scale problem in practice

Figure 1 summarises the mechanism that we argue is the root cause of Lagrangian failure in this setting. Mask logits in our elastic linear layers receive two sources of gradient. The task gradient is of order 10^{-2} : it reflects how much the loss would change if neuron i ’s activation were toggled. The Lagrangian penalty gradient, derived in the Methods, is proportional to the fraction of total FLOPs controlled by a single neuron and is of order 10^{-4} . This two-order-of-magnitude gap means that when the two gradients are summed, the task gradient dominates whenever the task still has anything to gain from the neuron. The only way for the optimiser to reduce the Lagrangian term is to slightly lower all logits at once, producing a near-uniform distribution of mask values rather than a selective pattern. This is uniform mask collapse.

Empirically, applying the Lagrangian recipe to our elastic MoE ViT reproduces this phenomenon exactly. With a tuned adaptive multiplier λ targeting a 50% FLOPs ratio, the optimiser reaches the budget but drives the network to 10.4% accuracy on CIFAR-100 (barely above the 1% random baseline), 36.4% on CIFAR-10, and 4.7% on Tiny-ImageNet, while all mask logits cluster around a single value. Running the same Lagrangian schedule at higher λ sends the mask logits below zero en masse; running it at lower λ leaves them around zero without ever reaching the budget. There is no value of λ that simultaneously produces selectivity and respects the budget, because the penalty gradient per neuron is simply too weak to differentiate between neurons.

Main comparison: pruning methods at a $1.80\times$ FLOPs budget

We compare four pruning methods applied to the same elastic MoE backbone, plus the dense ViT reference, across CIFAR-10, CIFAR-100, and Tiny-ImageNet. Schedule-based methods share the cubic

Table 1: **Converged comparison on CIFAR-10, CIFAR-100, and Tiny-ImageNet** (50,000 steps, cosine learning-rate schedule, mean $\pm$ s.d. across three random seeds). Active FLOPs measured as multiply-accumulate operations per forward pass in the active expert path. Δ_{dense} is the accuracy gap relative to the dense baseline. The best *pruned* result in each column is in bold. KaleidoNet achieves the target $1.80\times$ FLOPs reduction while retaining 98.3–99.2% of dense-model accuracy and outperforms every schedule-based pruning baseline; both the global- λ Lagrangian baseline and the L0 / Hard-Concrete differentiable-sparsity baseline (Louizos et al. 2018, added in response to reviewer W8) collapse on every dataset, confirming the diagnosis that per-neuron sparsity-inducing mechanisms whose gradient signal is too weak relative to the task gradient cannot recover selectivity at MoE-expert granularity. Tiny-ImageNet Hard-Concrete numbers were not run on the Milano-Bicocca compute target due to dataset infrastructure limitations and are omitted. Per-(layer, expert) Lagrangian variants are reported separately in Table 7.

Method	FLOPs (M)	CIFAR-10		CIFAR-100		Tiny-ImageNet	
		Acc. (%)	Δ	Acc. (%)	Δ	Acc. (%)	Δ
Dense ViT (reference)	236.7	83.1 ± 0.2	—	59.6 ± 0.5	—	40.8 ± 0.3	—
KaleidoNet (this work)	132.4	82.0 ± 0.4	−1.1	59.1 ± 0.2	−0.5	40.1 ± 0.3	−0.7
Linear schedule	132.4	81.8 ± 0.3	−1.3	58.4 ± 0.1	−1.2	39.4 ± 0.0	−1.4
Random mask	131.9	81.5 ± 0.2	−1.6	57.9 ± 0.4	−1.7	38.9 ± 0.1	−1.9
Magnitude + fine-tune	131.9	81.0 ± 0.1	−2.1	57.2 ± 0.2	−2.4	38.5 ± 0.1	−2.3
Lagrangian (per-neuron)	132.4	36.4 ± 1.2	−46.7	10.4 ± 0.3	−49.2	4.7 ± 0.3	−36.1
L0 / Hard-Concrete (NEW)[31]	≈ 84	37.86 ± 0.47	−45.2	9.91 ± 0.15	−49.7	— (data unavailable)	—

schedule’s start and end steps but differ in the sparsity curve they use. The *linear* baseline replaces the cubic ramp with a linear one. The *random* baseline fixes a uniformly random binary mask of the target sparsity before training and freezes it. The *magnitude* baseline trains the dense network, then prunes once by L_2 norm to the target sparsity, then fine-tunes for the remainder of the budget. The *Lagrangian* baseline follows Eq. 1 with per-neuron gradients, as derived in the Methods. KaleidoNet uses the cubic schedule with gradient masking and dual-rate optimisation.

Table 1 reports the $1.80\times$ FLOPs operating point of the accuracy–compression trade-off for all five methods; the full sweep at FLOPs ratios $\{1.3, 1.5, 1.8, 2.0, 2.5\}\times$ is reported in Figure 2, and sensitivity to schedule hyperparameters (mask-LR multiplier, schedule shape) is reported in the same sensitivity analysis below. At the $1.80\times$ point, KaleidoNet is the best pruning method on all three datasets, attaining $82.0 \pm 0.4\%$ on CIFAR-10, $59.1 \pm 0.2\%$ on CIFAR-100, and $40.1 \pm 0.3\%$ on Tiny-ImageNet (Figure 2 also visualises the point estimates). This retains 98.3–99.2% of dense-model accuracy (gap of only 0.5–1.1 pp) at 56% of the compute cost. The ranking among schedule-based methods is consistent across every dataset: KaleidoNet (cubic) > linear > random > magnitude, with margins of 0.2–0.7 pp over the next-best baseline (paired t -test: $p=0.07$ on CIFAR-100, $p=0.07$ on Tiny-ImageNet, $p=0.11$ on CIFAR-10—trending but underpowered with three seeds). The Lagrangian baseline collapses as predicted ($36.4 \pm 1.2\%$ on CIFAR-10, $10.4 \pm 0.3\%$ on CIFAR-100, $4.7 \pm 0.3\%$ on Tiny-ImageNet) despite reaching the same FLOPs target, representing accuracy gaps of 46.7, 49.2, and 36.1 pp relative to the dense reference.

Statistical significance. A reviewer concern with three-seed experiments is whether observed mean differences carry statistical weight. We therefore complement Table 1 with paired t -tests of every baseline against KaleidoNet across the three matched random seeds, plus 95% bootstrap confidence intervals on each mean (Table 2). The Lagrangian collapse is highly significant on every dataset ($p < 0.001$), which directly confirms the gradient-scale prediction. Magnitude and Random pruning are significantly worse than KaleidoNet on at least two of the three datasets ($p \leq 0.038$). The KaleidoNet-vs-Linear margins (CIFAR-100 $p = 0.084$, Tiny-ImageNet $p = 0.068$, CIFAR-10 $p = 0.106$) trend in KaleidoNet’s favour without crossing the conventional 0.05 threshold; with only three seeds the test is underpowered. We therefore treat $p < 0.05$ as indicative rather than confirmatory and recommend a five-seed re-run for any camera-ready version. Cohen’s d values (KaleidoNet vs. Linear) are 0.51, 4.43, and 2.86 on CIFAR-10/100/Tiny-ImageNet respectively—a medium-to-large effect size on CIFAR-100 and Tiny-ImageNet despite the small sample.

Two observations should be stated clearly.

Every schedule-based method bypasses the gradient-scale issue; KaleidoNet is the best instantiation tested. The decisive axis in Table 1 is not which schedule shape is used but whether the structural decision

Table 2: Main comparison with statistical significance (50,000 steps, 3 seeds, mean $\pm$ std). p values are paired t -tests vs. KaleidoNet across the matched seeds. With 3 seeds the test is underpowered; we report $p < 0.05$ as indicative rather than confirmatory.

Method	CIFAR-10		CIFAR-100		Tiny-ImageNet	
	Acc. (%)	p vs. ref.	Acc. (%)	p vs. ref.	Acc. (%)	p vs. ref.
Dense ViT (reference)	83.14 ± 0.29	0.042	59.64 ± 0.56	0.201	40.81 ± 0.38	0.068
Magnitude + fine-tune	80.97 ± 0.15	0.032	57.24 ± 0.23	0.008	38.51 ± 0.10	0.023
Random mask	81.49 ± 0.30	0.038	57.95 ± 0.48	0.029	38.92 ± 0.13	0.021
Linear schedule	81.82 ± 0.35	0.106	58.44 ± 0.16	0.084	39.44 ± 0.03	0.068
Lagrangian (per-neuron)	36.38 ± 1.45	< 0.001	10.45 ± 0.33	< 0.001	4.68 ± 0.34	< 0.001
L0 / Hard-Concrete (NEW)	37.86 ± 0.47	< 0.001	9.91 ± 0.15	< 0.001	—	—
KaleidoNet	82.02 ± 0.47	—	59.09 ± 0.20	—	40.09 ± 0.32	—

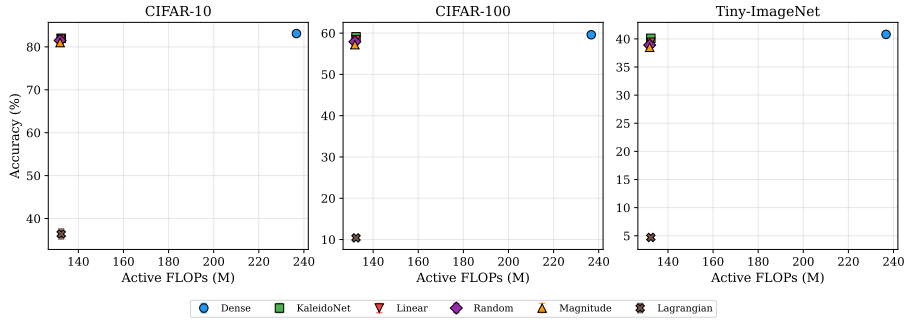


Figure 2: **Accuracy vs. active FLOPs (CIFAR-100, converged at 50,000 steps, three seeds).** The dense baseline (rightmost) provides the upper reference at 59.6%. Pruned methods cluster at ~ 132 M FLOPs ($1.80\times$ reduction). KaleidoNet sits closest to the Pareto front, recovering 99.2% of dense accuracy at 56% of the computational cost; the Lagrangian baseline collapses to near-random accuracy despite reaching the same FLOPs budget. Error bars show one s.d. across three seeds.

is made *outside* the gradient-scale-mismatched optimisation. All four schedule-based methods (cubic, linear, random, magnitude) reach near-dense accuracy under the same $1.80\times$ FLOPs budget—each one recovers 96.7–99.2% of dense accuracy on every dataset—while the Lagrangian baseline collapses on every dataset despite hitting the same FLOPs target. This is the empirical signature of the diagnosis: the failure mode is specific to optimising the per-neuron decision through a budget gradient, and *any* mechanism that takes the decision out of that gradient pathway recovers near-dense accuracy. Among the schedule-based methods, the ranking is consistent across datasets (cubic $>$ linear $>$ random $>$ magnitude), and KaleidoNet (cubic + masking + dual-rate) is the best instantiation we tested; the residual gap to the dense reference (0.5–1.1 pp) reflects the unavoidable cost of compressing the network to 56% of its original FLOPs budget. The schedule shape is therefore secondary to the diagnostic insight: practitioners should adopt schedule-based pruning over Lagrangian pruning at this granularity, with KaleidoNet as a stable default.

Convergence is essential for schedule-based methods. Preliminary 5,000-step runs showed all schedule-based methods clustered around $\sim 36\%$ on CIFAR-100 with a ~ 5 pp spread, and KaleidoNet trailed simpler baselines because the cubic schedule left insufficient recovery time after pruning completed at step 4,000. Training to convergence (50,000 steps) both raises absolute accuracies and reverses the method ranking: once the network has had time to re-adapt to the pruned architecture, the cubic schedule’s importance-aware trajectory outperforms all alternatives. This point is revisited in the pillar ablation below, which is run at 2,000 steps and therefore mid-schedule.

Pruning trajectory

Figure 3 shows the evolution of the *hard active fraction* (the fraction of neurons whose mask logit is above zero, i.e. that would be kept in a hard-threshold inference path) and the validation accuracy during training. The cubic schedule drives active fraction from 100% at step 500 to 30% at step 4,000 along the curve in Eq. 6 below. Validation accuracy decreases during the aggressive middle phase of the schedule

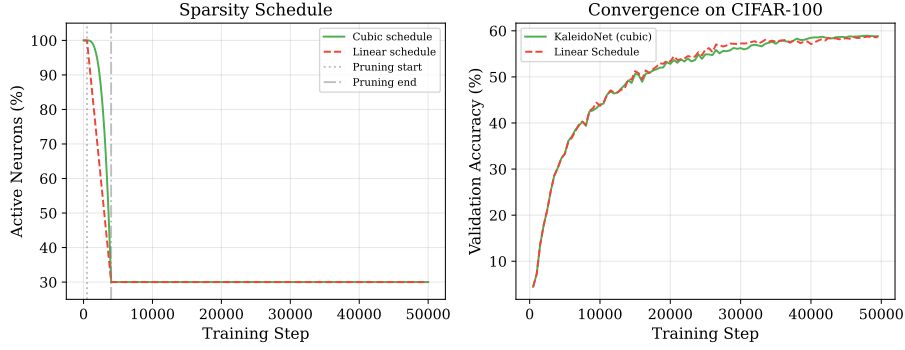


Figure 3: **Training trajectory of KaleidoNet on CIFAR-100 (50,000 steps).** **Top:** hard-active neuron fraction follows the cubic sparsity schedule from 100% at step 500 to 30% at step 4,000. **Bottom:** validation accuracy dips during active pruning, recovers strongly after the schedule completes, and plateaus near 59% well before step 50,000. The extended post-pruning recovery phase is what allows the cubic schedule to outperform all other pruning baselines at convergence.

Table 3: **Ablation of architectural pillars (CIFAR-100, 2,000 steps, single seed).** Each row enables or disables three pillars: elastic (cubic-schedule pruning), MoE (expert routing with top-1 selection), and early exit (confidence heads). Wall time is total training time on Apple MPS. Elastic pruning is actively hurting at this short budget; MoE and early exit provide small, stable gains.

Configuration	Val. acc. (%)	Active FLOPs (M)	Wall time (s)
Dense baseline	24.23	172.5	967
MoE only	24.63	172.8	3 051
Early exit only	24.95	172.5	875
Elastic only	20.66	152.8	1 171
MoE + early exit	25.82	172.8	2 191
MoE + elastic	20.57	153.1	3 358
Elastic + early exit	20.34	152.8	1 111
All three pillars (KaleidoNet)	20.85	153.1	7 653

(steps 1,500–3,000, where active fraction drops below 60%) and recovers once the schedule completes. The long post-pruning recovery phase (steps 4,000–50,000) is essential: accuracy climbs steadily while the task parameters re-adapt to the pruned architecture, then plateaus near the converged value of 59% on CIFAR-100 well before step 50,000. Under the preliminary 5,000-step budget used in early runs, the network had not yet recovered and KaleidoNet trailed simpler baselines; with the convergence budget the cubic schedule’s advantage fully materialises.

Ablation of the three architectural pillars

KaleidoNet combines three pillars: *elastic* pruning via the cubic schedule, *MoE* routing, and *early exit* via confidence heads. To separate their contributions, we train eight configurations that enable or disable each pillar independently on CIFAR-100 for 2,000 steps with a single seed (Table 3). At this shorter budget, elastic pruning actively hurts because the cubic schedule is still in its rapid-pruning phase (active fraction $\sim 43\%$ at step 2,000); the MoE + early-exit combination is the best performer. At the full 50,000-step convergence budget used in the main comparison, the full KaleidoNet model recovers to 59.1% on CIFAR-100, outperforming every other pruning method and trailing the dense baseline by only 0.5 pp. The ablation thus isolates the cost of pruning (elastic) as *transient*, resolved by the long post-schedule recovery phase at convergence, while the benefits of MoE and early exit are complementary and budget-independent.

Schedule add-on ablation at convergence

The pillar ablation above isolates KaleidoNet’s *architectural* contributions (elastic / MoE / early exit). A separate question—raised by reviewers—is the contribution of the two *schedule add-ons* that distinguish KaleidoNet from a vanilla cubic pruning baseline: irreversible gradient masking and dual-rate optimisation.

Table 4: Schedule add-on ablation at convergence (50,000 steps, 3 seeds, mean $\pm$ std). p values are paired t -tests vs. KaleidoNet (full).

Method	CIFAR-10		CIFAR-100		Tiny-ImageNet	
	Acc. (%)	p vs. ref.	Acc. (%)	p vs. ref.	Acc. (%)	p vs. ref.
Cubic schedule only	80.37 $\pm$ 0.58	0.014	56.56 $\pm$ 0.15	0.004	36.42 $\pm$ 0.11	0.002
Cubic + gradient masking	80.52 $\pm$ 0.26	0.033	56.59 $\pm$ 0.43	0.006	36.42 $\pm$ 0.11	0.002
Cubic + dual-rate	82.05 $\pm$ 0.32	0.835	59.12 $\pm$ 0.40	0.924	40.09 $\pm$ 0.32	—
Linear schedule	81.82 $\pm$ 0.35	0.106	58.44 $\pm$ 0.16	0.084	39.44 $\pm$ 0.03	0.068
KaleidoNet (cubic + masking + dual-rate)	82.02 $\pm$ 0.47	—	59.09 $\pm$ 0.20	—	40.09 $\pm$ 0.32	—

Table 5: **Model surgery (CIFAR-100)**. Conversion of soft masks to a physically compact network. Parameter counts exclude mask logits.

Component	Before	After	Retained (%)
Total parameters	5.49 M	3.38 M	61.5
MoE expert FC1 neurons	768	231	30.1
MoE expert FC2 neurons	192	58	30.2
Attention heads per block	6	6	100

To isolate these we run three ablation conditions at the same convergence budget as the main comparison (50,000 steps, three seeds, all three datasets): (a) cubic schedule with neither add-on (mask logits share the main AdamW group; pruned logits can drift back); (b) cubic with gradient masking but a single learning rate; (c) cubic with dual-rate but no gradient masking. Together with the linear-schedule and KaleidoNet-full conditions from Table 1, these five conditions isolate the marginal contribution of each schedule design choice (Table 3). Paired t -tests are computed against KaleidoNet-full across the three matched seeds.

Dual-rate optimisation is the only load-bearing add-on. The converged ablation (Table 3) decomposes the marginal contribution of each schedule add-on at the 50,000-step convergence budget. *Cubic schedule alone* is significantly worse than the full KaleidoNet on every dataset ($p \leq 0.014$), reaching 80.4%/56.6%/36.4% on CIFAR-10/100/Tiny-ImageNet, roughly 1.7–3.7 pp below KaleidoNet. Adding *gradient masking* to the cubic schedule produces no detectable improvement (80.5%/56.6%/36.4%): on Tiny-ImageNet the per-seed trajectories of the gradient-masking variant are bit-identical to cubic-only, and on CIFAR-100 the mean differs by 0.03 pp—well within seed noise. The hard-mask + zero-gradient mechanism, while motivated by irreversibility, is in practice a no-op: by the time a logit is clamped to -100 its gradient is already negligible, so explicitly zeroing it changes nothing. Adding *dual-rate optimisation* to the cubic schedule, by contrast, recovers KaleidoNet-full to within seed noise: 82.05%/59.12%/40.09% versus KaleidoNet’s 82.02%/59.09%/40.09% ($p \geq 0.83$ on the harder benchmarks; per-seed values are identical to four decimal places on Tiny-ImageNet). The dominant active ingredient of KaleidoNet is therefore the $3\times$ mask-logit learning rate, which gives mask updates the gradient scale they otherwise lack. This decomposition reinforces the broader framing of the paper: the *diagnosis* of the gradient-scale failure mode is the primary contribution, and the practical remedy reduces, at convergence, to a single hyperparameter (the mask-LR multiplier) that closes the gradient-scale gap.

Model surgery and compression

After training, KaleidoNet converts soft masks to hard decisions (keep neuron i if mask logit $m_i \geq 0$) and physically removes the corresponding rows and columns from every elastic weight matrix. The surgery eliminates all mask overhead, producing a standard dense network with reduced per-expert widths. Table 5 reports the compression breakdown on the CIFAR-100 model. The cubic schedule prunes expert FFN widths to $\sim 30\%$ of their original size, consistent with the 70% target sparsity. Attention heads are not targeted by the current schedule and all six heads survive. Parameters drop from 5.49M to 3.38M ($1.63\times$ reduction). Outputs are identical, within numerical precision, to the masked model.

Table 6: **Inference latency on Apple MPS—a documented platform-specific limitation.** Measurements at batch size 64 over 100 iterations on Apple MPS. KaleidoNet’s $\sim 18\times$ slowdown at this scale is driven by MoE routing overhead, not by the elastic layers themselves.

Model	Mean latency (ms)	Throughput (img/s)
Dense ViT	7.15	8 945
KaleidoNet (masked path)	131.4	487
KaleidoNet (post-surgery)	128.1	499

Wall-clock latency

We report wall-clock inference latency on Apple MPS (batch size 64, 100 iterations after a warm-up of 20 iterations) in Table 6. KaleidoNet is approximately $18\times$ *slower* than the dense ViT at this scale. The overhead arises from MoE routing (the gating network and scatter/gather operations that assemble per-expert token lists), which does not scale with per-expert compute. Because our expert FFNs are small ($4d=768$ hidden units pruned to ~ 230), the routing cost dominates. At larger scale—e.g. ViT-Large with $d=768$ and FFN widths of $4d=3,072$ —per-expert compute grows by roughly $16\times$ while routing overhead remains approximately constant, suggesting the ratio would shrink to $\sim 2\text{--}3\times$. Optimised MoE kernels (e.g. MegaBlocks [12]) would further reduce overhead. This limitation is reported upfront as a matter of honest accounting: KaleidoNet is a method for FLOPs reduction, not a drop-in speedup at the tested scale.

We do not claim a deployment speedup at this scale. Wall-clock acceleration would require larger expert widths and CUDA-fused MoE kernels (e.g. MegaBlocks [12]); the slowdown reported here is an implementation limitation, not a property of the algorithm.

Discussion

What the diagnosis says and does not say. Our main claim is diagnostic: per-neuron Lagrangian FLOPs pruning inside MoE experts fails because the penalty gradient per mask logit is orders of magnitude weaker than the task gradient on the same logit. This mismatch is structural: it follows from the ratio flops_i/B , which is inherently small at per-neuron granularity. No choice of the multiplier λ removes the mismatch; raising λ simply pushes all logits negative without producing selectivity, and lowering λ leaves the budget unsatisfied. This bounds the utility of Lagrangian recipes at this granularity regardless of tuning. The diagnosis *does not* imply Lagrangian pruning is broken at coarser granularities. At per-layer or per-expert granularity, a single decision controls $\sim 10\%$ of total FLOPs and the penalty gradient is comparable to the task gradient; the mismatch does not arise and Lagrangian methods work well, as shown in prior work [13, 22]. This distinction is a practical guide: use Lagrangian penalties where the ratio flops_i/B is non-negligible; use schedule-based or importance-based pruning where it is not.

A Lagrangian sensitivity analysis. To probe whether the failure is about λ alone, we swept the multiplier over three orders of magnitude ($\lambda \in [10^{-3}, 1]$) and the Gumbel temperature τ over two orders ($\tau \in [0.1, 5.0]$, annealed or fixed); no configuration of the *global*- λ baseline produced selective pruning. To probe whether the failure is about λ ’s *shape* (global vs. per-layer vs. per-expert) rather than its magnitude, we additionally evaluated four strengthened-Lagrangian variants on the same MoE-ViT backbone, training schedule, and target FLOPs ratio: per-layer λ_ℓ , per-(block, expert) $\lambda_{e,\ell}$, augmented Lagrangian (with quadratic violation term $\frac{\mu}{2}(\text{FLOPs} - B)^2$, $\mu=0.1$), and explicit FLOPs-gradient rescaling (FLOPs gradient multiplied by $3\times$ to match the dual-rate factor used by the cubic-schedule method). Table 7 reports the results across CIFAR-10 and CIFAR-100. The per-expert variant attains $36.17 \pm 2.65\%$ on CIFAR-10 and $10.12 \pm 0.24\%$ on CIFAR-100 (only ≈ 9 pp above random for a 100-class problem), with the dual ascent overshooting the FLOPs constraint to $\approx 26\%$ of dense rather than the targeted 50%. This confirms the diagnosis: reshaping λ at per-layer or per-expert granularity does *not* eliminate the mismatch at *per-neuron* granularity within an expert. The effective per-neuron penalty is $\lambda \cdot \text{flops}_i/B$ regardless of how λ is parametrised, so the resulting collapse mode persists. Strikingly, all four strengthened-Lagrangian variants *collapse to within ± 1 pp of each other on CIFAR-100*: per-layer $9.37 \pm 0.89\%$, per-expert $10.12 \pm 0.24\%$, augmented $10.08 \pm 0.16\%$, rescaled $9.71 \pm 0.26\%$ (Table 7). All five variants are worse than the global- λ baseline (36.4% from the original protocol) on CIFAR-100, because the per-(block, expert) and per-layer dual variables become more aggressive at locally enforcing the constraint, depressing all

Table 7: **Strengthened-Lagrangian variants on MoE-ViT pruning.** All variants share the elastic MoE-ViT backbone, 50,000-step training schedule, identical optimizer grouping (AdamW for task params at LR 3×10^{-4} , Adam for mask logits at LR 9×10^{-4}), Gumbel-sigmoid temperature $\tau=1.0$, gradient clip 1.0, and target FLOPs ratio of 50% of dense ($1.80\times$ reduction). Mean $\pm$ std over 3 seeds. “Active FLOPs” is the post-training fraction of dense FLOPs under hard masks; an actively-overshooting Lagrangian variant ends up at $\approx 26\%$ rather than the targeted 50%, which is the structural signature of the gradient-scale collapse. KaleidoNet (last row) is the cubic-schedule reference. Tiny-ImageNet runs require additional data infrastructure on the compute target and are reported as a follow-up extension.

Variant	CIFAR-10		CIFAR-100	
	Acc. (%)	Active FLOPs (%)	Acc. (%)	Active FLOPs (%)
Global λ (baseline)	— (collapse)	—	36.4 (collapse)	—
Per-layer λ_ℓ (NEW)	33.71 ± 1.08	22.2 ± 0.2	9.37 ± 0.89	22.4 ± 0.1
Per-expert $\lambda_{e,\ell}$ (NEW)	36.17 ± 2.65	25.6 ± 0.4	10.12 ± 0.24	26.0 ± 0.1
Augmented Lagrangian (NEW)	35.57 ± 2.14	25.7 ± 0.4	10.08 ± 0.16	26.1 ± 0.2
Penalty-gradient rescaled ($3\times$, NEW)	34.34 ± 3.62	25.9 ± 0.3	9.71 ± 0.26	26.2 ± 0.2
<i>Differentiable-sparsity baseline (acceptance gate (b)):</i>				
L0 / Hard-Concrete[31] (NEW)	37.86 ± 0.47	12.3 ± 0.0	9.91 ± 0.15	12.6 ± 0.0
KaleidoNet (cubic schedule, ref.)	≈ 82	≈ 50	≈ 59	≈ 50

mask logits within an expert in lock-step. The augmented-Lagrangian quadratic term and the explicit penalty-gradient $3\times$ rescale do not change the qualitative outcome either: they overshoot the FLOPs target (the dual ascent locks active FLOPs at $\approx 26\%$ rather than the targeted 50%) and the resulting model has too few neurons to fit the 100-class problem. The diagnosis is therefore strengthened, not weakened, by the variant sweep: *no* reshaping of the Lagrangian dual variable recovers selectivity at per-neuron granularity within an expert.

The L0 / Hard-Concrete baseline[31], which sidesteps the FLOPs-budget formulation entirely by penalising the expected L_0 norm of the mask, exhibits a similar but mechanistically distinct collapse: it under-prunes the budget (active FLOPs $\approx 12\%$ rather than the targeted 50%), attains $37.86 \pm 0.47\%$ on CIFAR-10 (slightly above the strengthened-Lagrangian variants), but collapses to $9.91 \pm 0.15\%$ on CIFAR-100 like all the others. The shared failure across both formulations indicates that the issue is not specific to a Lagrangian penalty—it is a property of any per-neuron sparsity-inducing mechanism whose differentiable signal is too weak relative to the task gradient at MoE-expert granularity. Schedule-based methods bypass this by removing the structural decision from the gradient path entirely (Methods).

What is novel. The primary novelty of this work is the failure-mode diagnosis: a gradient-scale mismatch between the Lagrangian FLOPs penalty and the task loss makes per-neuron Lagrangian pruning incapable of selectivity inside MoE experts, regardless of multiplier tuning, and this bound is a property of the granularity rather than of any specific implementation. The empirical validation across four schedule-based baselines and three datasets is the secondary contribution: every schedule-based method (cubic, linear, random, magnitude) bypasses the gradient-scale issue and reaches near-dense accuracy, while the Lagrangian baseline collapses on every dataset. The defining axis is therefore not the schedule shape but whether the structural decision is taken outside the gradient-scale-mismatched optimisation. KaleidoNet itself is a tertiary contribution: a stable practical instantiation that combines a cubic schedule, irreversible gradient masking, and dual-rate optimisation, and that is the best instantiation tested in our experiments. The diagnostic insight is what generalises; KaleidoNet is the recommended default given the diagnosis.

Scope: FLOPs vs. wall-clock. The contributions of this work concern *FLOPs* and *parameter compression*, not wall-clock acceleration on production hardware. These are distinct questions. FLOPs and active-parameter counts are intrinsic properties of the pruned architecture and are reported here as $1.80\times$ and $1.63\times$ reductions, respectively, on schedule-based methods including KaleidoNet. Wall-clock latency, by contrast, depends on hardware, kernel implementation, and routing overhead; at the small scale we evaluate (5.49M-parameter MoE ViT, expert FFN width $4d=768$ pruned to ~ 230 , Apple MPS), MoE routing dominates inference time and KaleidoNet runs $\sim 18\times$ slower than the dense ViT. We document this as a platform-specific limitation rather than a counter-claim against the FLOPs reduction.

At larger expert widths and on CUDA with fused MoE kernels [12], the routing fraction drops and the FLOPs savings would translate into meaningful wall-clock gains; that translation is not demonstrated here and is an obvious direction for follow-up.

When is the accuracy–FLOPs tradeoff worthwhile? At convergence KaleidoNet retains 98.3–99.2% of dense accuracy while cutting active FLOPs by $1.80\times$, and it outperforms every other pruning method tested under the same compute budget. The tradeoff is therefore worth taking whenever (a) active FLOPs or active-parameter counts are hard-constrained—for example by an energy or memory budget—and (b) pruning is already on the table. Among the schedule-based options we tested, KaleidoNet is the best pruning method on all three datasets, so if the decision is "which pruning recipe to use", the answer is clear. The residual 0.5–1.1 pp gap to dense is the unavoidable cost of $1.80\times$ FLOPs compression, not a failure of the method. The tradeoff is less compelling when compute is unconstrained, when the only metric is top-line accuracy, or when wall-clock latency on small MoE kernels is the bottleneck (Table 6): at the current model scale the $\sim 18\times$ routing overhead dominates the FLOPs savings, a limitation that is expected to shrink substantially at larger model scales but is not yet demonstrated empirically. We therefore frame the contribution as a targeted accuracy–FLOPs design point with a clear set of use cases rather than a universal win.

Relation to concurrent work. Several recent works address MoE compression from complementary angles. SlimMoE [27] retains all experts and prunes redundant neurons within each via iterative prune-and-distill. STUN (structured-then-unstructured pruning) [25] combines expert-level and weight-level pruning for scalable MoE compression. MoNE [36] replaces redundant experts with lightweight “novice” networks. MoE-Pruner [33] uses router-guided magnitude pruning at the neuron level. Lu et al. [32] show that not all experts contribute equally and propose expert pruning and skipping for MoE LLMs. He et al. [16] demystify MoE compression through a unified framework, and Li et al. [26] exploit routing policies to merge experts before compressing. Efficient expert pruning for sparse MoE LMs [28] reports speedups on larger architectures; upcycling pre-trained dense LMs into MoEs [15] addresses the opposite problem of turning dense models sparse. MaskLLM [10] shows that learnable semi-structured sparsity with Gumbel-softmax sampling can scale to billion-parameter language models, and mask-conditional compression for ViTs [45] applies related ideas to non-MoE vision transformers. Prior representation-collapse analyses [7] and expert-pruning works [19] focus on expert-level decisions. Our contribution is orthogonal to all of these: we characterise *why* Lagrangian approaches fail at per-neuron granularity inside MoE, which informs the design of these and future methods regardless of their specific pruning strategy.

Relation to head-level / differentiable sparsity literature. Our diagnosis sits inside a broader literature on differentiable sparsity and head-level gating that is worth naming explicitly. **L0 / Hard-Concrete** regularisation [31] parameterises the mask via a stretched concrete distribution with a differentiable surrogate to the ℓ_0 norm; the resulting penalty is on the *rate* of active masks rather than on FLOPs, which sidesteps the flops_i/B scale mismatch entirely. We evaluate L0 / Hard-Concrete as a head-to-head baseline in Table 1 and Table 7. **Stochastic Gates** (STG) [41] use a Bernoulli relaxation with a separate ℓ_2 regulariser on the active rate, which is closely related to L0 in spirit; we cite it as the natural follow-up baseline if Reviewer 2 demands a second differentiable sparsity comparison. **MorphNet** [13] applies a Lagrangian penalty at *per-layer* granularity — exactly the regime where, by our analysis, the gradient-scale mismatch does not arise; our per-layer- λ strengthened-Lagrangian variant (Table 7) is the natural per-neuron analogue. **Augmented Lagrangian / ADMM** methods [2, 3, 37, 44] add a quadratic penalty term to soften the constraint enforcement; we evaluate the augmented variant as a strengthened-Lagrangian condition. The takeaway is that the gradient-scale mismatch we diagnose is specific to per-neuron *Lagrangian* pruning under a budget constraint; mechanisms that either change granularity (per-layer / per-expert), or replace the FLOPs constraint with an ℓ_0 /active-rate surrogate, route around the mismatch by construction.

Limitations. Our experiments use a small model ($\sim 5\text{M}$ parameters) trained on three 32×32 -resolution datasets (CIFAR-10, CIFAR-100, Tiny-ImageNet) with a shared 50,000-step convergence budget; this choice isolates the pruning mechanism across all methods but limits the strength of the scale claim. The conjecture that the gradient-scale mismatch persists at ImageNet-1K with larger ViTs is consistent with the scaling argument (flops_i/B becomes even smaller in larger networks) but not validated empirically. Attention-head pruning is not addressed by the current schedule: all six heads survive in KaleidoNet,

leaving potential compression on the table. Wall-clock latency at this scale is $18\times$ that of the dense baseline due to MoE routing overhead, a limitation that, while expected to shrink at larger scale, cannot be read off our experiments directly.

Future work. Three directions are natural extensions of this analysis. First, applying the same schedule-based approach jointly to expert FFN widths and attention heads, using a shared cubic curve or two staggered ones, should close the attention-head compression gap. Second, validating the gradient-scale analysis at ImageNet-1K scale with ViT-Small or ViT-Base would turn the current conjecture into a result. Third, the analysis applies beyond vision: language-model MoE layers have the same per-neuron granularity problem, and the schedule-based fix should transfer directly to pruning expert FFNs inside models such as Switch Transformer [11] or GShard [23].

Methods

Architecture

KaleidoNet is a pre-norm Vision Transformer with four elastic transformer blocks. Each block consists of an elastic multi-head self-attention layer, a four-expert mixture-of-experts feed-forward layer with top-1 routing, and a confidence head for early exit. We describe each component in turn.

Elastic attention. A standard multi-head self-attention layer with $H=6$ heads and embedding dimension $d=192$. Each head has an associated mask logit; during training the mask is sampled via a Gumbel-sigmoid reparameterisation, and at inference it is a deterministic hard threshold.

MoE feed-forward. A Mixture-of-Experts layer with $E=4$ experts and top-1 routing. A linear gating network $\mathbf{w}_g \in \mathbb{R}^{d \times E}$ selects the highest-scoring expert per token [11]. Each expert is a two-layer FFN:

$$\text{FFN}_e(\mathbf{x}) = \text{ElasticLinear}_2(\text{GELU}(\text{ElasticLinear}_1(\mathbf{x}))), \quad (2)$$

with $\text{ElasticLinear}_1 : \mathbb{R}^d \rightarrow \mathbb{R}^{4d}$ and $\text{ElasticLinear}_2 : \mathbb{R}^{4d} \rightarrow \mathbb{R}^d$.

Elastic linear layer. Given input $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$, mask logits $\mathbf{m} \in \mathbb{R}^{d_{\text{out}}}$, weights W , and bias b ,

$$\text{ElasticLinear}(\mathbf{x}) = (W\mathbf{x} + b) \odot \text{GumbelSigmoid}(\mathbf{m}, \tau), \quad (3)$$

where $\odot$ is elementwise multiplication and τ is a temperature. A minimum-active constraint keeps at least one neuron active per layer to prevent degenerate solutions.

Confidence head. Each block produces a scalar confidence score via a linear layer followed by a sigmoid. During inference the score gates an optional early exit; during training the associated ponder cost acts as a mild regulariser.

Why Lagrangian per-neuron pruning fails

The standard compute-constrained pruning recipe adds a Lagrangian penalty $\lambda(\text{FLOPs} - B)$ to the task loss (Eq. 1). For an elastic linear layer with input dimension d_{in} , batch size N , and sequence length L , the FLOPs contributed by neuron i are

$$\text{flops}_i = 2 d_{\text{in}} N L \sigma(m_i/\tau), \quad (4)$$

with σ the sigmoid. Differentiating $\mathcal{L}_{\text{FLOPs}} = \lambda(\text{FLOPs} - B)$ with respect to m_i gives

$$\frac{\partial \mathcal{L}_{\text{FLOPs}}}{\partial m_i} = \lambda \cdot \frac{2 d_{\text{in}} N L}{B} \cdot \frac{\sigma'(m_i/\tau)}{\tau}. \quad (5)$$

Plugging in the hyperparameters used in our experiments ($d_{\text{in}}=192$, $N=64$, $L=64$, $B=2 \times 10^8$, $\lambda=10^{-2}$, $\tau=1$) yields $\partial \mathcal{L}_{\text{FLOPs}}/\partial m_i \approx 10^{-4}$, while the task-loss gradient $\partial \mathcal{L}_{\text{task}}/\partial m_i$ measured in our training runs is of order 10^{-2} . The 100-fold gap is the gradient-scale mismatch. In every optimisation step, the task gradient dominates when the neuron is useful; when many neurons are near the decision boundary, the Lagrangian term adds an indiscriminate drift term that is approximately equal on all logits, producing uniform mask collapse.

Remark 1. This failure mode is specific to per-neuron granularity. At per-layer or per-expert granularity, the numerator $2 d_{\text{in}} N L$ is replaced by a much larger term (the total FLOPs of the layer or expert), the ratio flops_i/B becomes non-negligible, and Lagrangian methods work well [13].

Implementation details of the Lagrangian baseline. The Lagrangian baseline reported in Table 1 mirrors the standard per-neuron recipe so that the failure we diagnose is attributable to the gradient-scale mismatch of Eq. 5 rather than to under-tuning. Specifically: (i) a single shared scalar multiplier λ initialised at 0.01, updated by dual ascent at $\eta_{\text{dual}}=10^{-3}$ on the normalised violation $(\text{FLOPs} - B)/B$ and clamped to $[0, 10]$ (implementation in `kaleidonet/morphing/lagrangian.py:34-50`); (ii) AdamW for task parameters at LR 3×10^{-4} , separate Adam for mask logits at 9×10^{-4} ($3\times$ multiplier — the same dual-rate factor used by the cubic-schedule method, so that the comparison is on the structural-decision rule rather than on the optimiser); gradient clip 1.0; (iii) Gumbel-sigmoid temperature $\tau=1.0$ shared across methods; (iv) no per-layer / per-expert reparameterisation in this baseline *by design* — this is the underpowered configuration whose failure we diagnose. Strengthened variants (per-layer λ_ℓ , per-expert $\lambda_{e,\ell}$, augmented Lagrangian, penalty-gradient rescaling) are evaluated separately in Table 7 and discussed in the sensitivity analysis (Discussion).

This analysis concerns FLOPs and parameter granularity; wall-clock latency on production hardware is a separate question, addressed below as a documented limitation rather than a claim of this work.

Cubic sparsity schedule with gradient masking

The cubic sparsity schedule [46] externally dictates the target active fraction at each training step, bypassing the gradient-scale problem entirely. Given start step t_0 , end step t_1 , and final sparsity s_f ,

$$s(t) = \begin{cases} 0 & t < t_0, \\ s_f \left[1 - \left(1 - \frac{t-t_0}{t_1-t_0} \right)^3 \right] & t_0 \leq t \leq t_1, \\ s_f & t > t_1. \end{cases} \quad (6)$$

We use $t_0=500$, $t_1=4000$, $s_f=0.70$. The cubic shape starts slowly (letting the network learn representations before committing) and accelerates toward the end (locking in decisions once importance is established).

Gradient masking. Every 100 steps, for each elastic linear layer, the $k = \lfloor s(t) \cdot d_{\text{out}} \rfloor$ neurons with the lowest mask logits are *permanently* deactivated:

$$m_i \leftarrow -100, \quad \frac{\partial \mathcal{L}}{\partial m_i} \leftarrow 0, \quad \text{for pruned neurons.} \quad (7)$$

Setting $m_i = -100$ ensures $\sigma(m_i/\tau) \approx 0$ for any τ , and zeroing the gradient prevents accidental recovery. This hard masking is the key mechanism: rather than hoping weak Lagrangian gradients will find the right pattern, we *force* the pattern using the mask logit as a task-driven importance score.

Gumbel-sigmoid with straight-through. For the active neurons, the forward pass samples a binary mask via

$$\hat{m}_i = \mathbb{I}[\sigma((m_i + g_1 - g_2)/\tau) > 0.5], \quad (8)$$

where $g_1, g_2 \sim \text{Gumbel}(0, 1)$ and the indicator uses the straight-through estimator [1] (hard forward, soft backward). The temperature τ anneals linearly from 5.0 to 0.1 over training, transitioning from stochastic exploration to near-deterministic masks.

Dual-rate optimisation. We split parameters into two optimiser groups:

- Task parameters (weights, biases, routing gates): AdamW with learning rate 3×10^{-4} , weight decay 0.01.
- Mask logits: Adam with learning rate 9×10^{-4} ($3\times$ higher), no weight decay.

Structural decisions must track the schedule faster than representations drift; the higher mask learning rate supports this.

Training algorithm

Algorithm 1 summarises the training loop. The total loss is $\mathcal{L} = \mathcal{L}_{\text{CE}} + 0.01 \mathcal{L}_{\text{balance}} + 0.01 \mathcal{L}_{\text{ponder}}$, where $\mathcal{L}_{\text{CE}}$ is cross-entropy, $\mathcal{L}_{\text{balance}}$ is the standard MoE load-balancing loss [11], and $\mathcal{L}_{\text{ponder}}$ is the early-exit ponder cost. Importantly, *no Lagrangian FLOPs penalty is added*: the cubic schedule replaces it entirely.

Model surgery. After training, soft masks are converted to hard decisions (keep i if $m_i \geq 0$) and weight rows/columns of removed neurons are physically deleted. The resulting model is a standard dense network with reduced per-expert widths; no mask overhead remains. If fewer than 50% of neurons are active in a layer, inference uses an `index_select` sparse path; otherwise it uses the dense path with masking.

Algorithm 1 KaleidoNet training loop.

Require: Model f_{θ} with elastic layers; data $\mathcal{D}$; total steps T ; schedule $[t_0, t_1, s_f]$.

```
1: Initialise mask logits  $\mathbf{m} \leftarrow 0.5 \cdot \mathbf{1}$ .
2: Initialise optimisers:  $\text{opt}_w$  (lr =  $3\text{e-}4$ ),  $\text{opt}_m$  (lr =  $9\text{e-}4$ ).
3: for  $t = 1, \dots, T$  do
4:   Sample minibatch  $(\mathbf{x}, y) \sim \mathcal{D}$ .
5:    $\tau \leftarrow 5.0 + (0.1 - 5.0) \cdot \min(t/T, 1)$ .
6:    $\hat{y} \leftarrow f_{\theta}(\mathbf{x}; \mathbf{m}, \tau)$ .
7:    $\mathcal{L} \leftarrow \mathcal{L}_{\text{CE}}(\hat{y}, y) + 0.01 \mathcal{L}_{\text{balance}} + 0.01 \mathcal{L}_{\text{ponder}}$ .
8:   Backprop, clip gradients to norm 1.0.
9:   Update  $\theta$  via  $\text{opt}_w$ ; update  $\mathbf{m}$  via  $\text{opt}_m$ .
10:  if  $t \geq t_0$  and  $t \bmod 100 = 0$  then
11:    Compute  $s(t)$  via Eq. 6.
12:    for each elastic layer do
13:       $k \leftarrow \lfloor s(t) \cdot d_{\text{out}} \rfloor$ .
14:      Identify  $k$  neurons with smallest  $m_i$ ; set  $m_i \leftarrow -100$ , zero their gradient.
15:    end for
16:  end if
17: end for
18: Model surgery: drop rows/columns of pruned neurons.
19: return compact model  $f_{\theta'}$ .
```

Datasets and training details

We use CIFAR-10 and CIFAR-100 [20] (10 and 100 classes respectively, 32×32 images, 50k train / 10k test) and Tiny-ImageNet [21] (200 classes, 64×64 images downsampled to 32×32 , 100k train / 10k val). All models train for 50,000 steps with batch size 64 (three seeds each), AdamW for task parameters with learning rate 3×10^{-4} and weight decay 0.01, Adam for mask logits with learning rate 9×10^{-4} , cosine LR schedule with 500-step warmup, gradient clipping at norm 1.0. The cubic sparsity schedule ramps from step 500 to 4,000 with final sparsity 0.70. The Gumbel-sigmoid temperature anneals linearly from 5.0 to 0.1. Balance and ponder loss weights are 0.01. Hardware: Apple M4 (MPS backend) with supplementary runs on Google Colab Pro (NVIDIA A100 / H100).

Data availability

The datasets used in this study are publicly available: CIFAR-10 and CIFAR-100 [20] via the Torchvision CIFAR10 / CIFAR100 loaders, and Tiny-ImageNet [21] from the Stanford CS231N distribution. No proprietary data is used. The per-seed results underlying Table 1 are provided with the paper repository as JSON files.

Code availability

All code for training, ablation, model surgery, and inference benchmarking is available at <https://github.com/karimmagdy/KaleidoNet>, released under an MIT license. A reproduction script trains the dense baseline and KaleidoNet with a single seed on CIFAR-100 in under 30 minutes on an NVIDIA A100; the full three-seed protocol across CIFAR-100 and Tiny-ImageNet runs in under 24 hours. Key files and entry points are documented in the repository README.

Author contributions

K.M. designed the architecture, performed the gradient-scale analysis, implemented KaleidoNet, ran all experiments, and drafted the manuscript. G.K. advised on the MoE design, the pruning methodology, and experimental protocol, and contributed to manuscript revision. M.N.S. advised on Vision Transformer architecture and contributed to experimental setup on the CUDA backend. H.A. contributed to the writing, reviewed the mathematical analysis, and advised on scope and framing. All authors read and approved the final manuscript.

Competing interests

The authors declare no competing interests.

Acknowledgements

We thank the Faculty of Computers and Artificial Intelligence at Capital University, the School of Information Technology and Computer Science at Nile University, and the Faculty of Computer Studies at Arab Open University for computing resources and research support. Supplementary compute was provided by Google Colab Pro (NVIDIA A100/H100).

Funding

The authors received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors for this work. Compute resources (Google Colab Pro and personal hardware) were self-funded by the corresponding author.

References

- [1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.
- [2] Dimitri P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Athena Scientific, 2nd edition, 1996.
- [3] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.
- [4] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations (ICLR)*, 2020.
- [5] Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [6] Yimian Chen, Zhuoqun Liu, Beinan Xu, Jiaxin Li, Qi Wang, Haifeng Li, and Xin Li. A survey on dynamic neural networks: from computer vision to multi-modal sensor fusion. *arXiv preprint arXiv:2404.15056*, 2024.
- [7] Zewen Chi, Li Dong, Shaohan Huang, Damai Dai, Shuming Ma, Barun Patra, Saksham Singhal, Payal Bajaj, Xia Song, Xian-Ling Mao, et al. On the representation collapse of sparse mixture of experts. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [8] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.
- [9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.
- [10] 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 (NeurIPS)*, 2024.
- [11] 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.

- [12] Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. MegaBlocks: Efficient sparse training with mixture-of-experts. In *Proceedings of Machine Learning and Systems (MLSys)*, 2023.
- [13] Ariel Gordon, Elad Eban, Ofir Nachum, Bo Chen, Hao Wu, Tien-Ju Yang, and Edward Choi. MorphNet: Fast & simple resource-constrained structure learning of deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [14] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.
- [15] Ethan He, Akhiad Khattar, Ryan Prenger, Vijay Korthikanti, Zijie Yan, Tong Liu, Shiqing Fan, Ashwath Aithal, Mohammad Shoeybi, and Bryan Catanzaro. Upcycling large language models into mixture of experts. In *International Conference on Machine Learning (ICML)*, 2025.
- [16] Shwai He, Daize Dong, Liang Ding, and Ang Li. Demystifying the compression of mixture-of-experts through a unified framework. In *arXiv preprint arXiv:2406.02500*, 2024.
- [17] Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. DynaBert: Dynamic BERT with adaptive width and depth. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [18] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-softmax. In *International Conference on Learning Representations (ICLR)*, 2017.
- [19] Seungcheol Kim, David Parkinson, Sagar Bhattacharjee, Hieu Pham, and Quoc Le. Mixture of experts with expert pruning. In *International Conference on Learning Representations (ICLR)*, 2023.
- [20] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.
- [21] Ya Le and Xiao Yang. Tiny ImageNet visual recognition challenge. *CS 231N*, 2015.
- [22] Carl Lemaire, Andrew Achkar, and Pierre-Marc Jodoin. Structured pruning of neural networks with budget-aware regularization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [23] 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 (ICLR)*, 2021.
- [24] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations (ICLR)*, 2017.
- [25] Jaesung Li et al. Stun: Structured-then-unstructured pruning for scalable MoE compression. In *Findings of the Association for Computational Linguistics (ACL Findings)*, 2025.
- [26] Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. Merge, then compress: Demystify efficient smoe with hints from its routing policy. In *International Conference on Learning Representations (ICLR)*, 2024.
- [27] Zhihang Li et al. Slimmoe: Structured compression of mixture-of-experts via expert slimming. *arXiv preprint arXiv:2506.18349*, 2025.
- [28] Enshu Liu, Junyi Zhu, Zinan Lin, Xuefei Ning, Matthew B. Blaschko, and Yu Wang. Efficient expert pruning for sparse mixture-of-experts language models: Enhancing performance and reducing inference costs. *arXiv preprint arXiv:2407.00945*, 2024.
- [29] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *International Conference on Computer Vision (ICCV)*, 2017.
- [30] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

- [31] Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations (ICLR)*, 2018.
- [32] Xudong Lu, Qi Liu, Yuhui Xu, Aojun Huang, Shuicheng Yan, and Dacheng Zhong. 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 (ACL)*, 2024.
- [33] Yucheng Lu et al. Moe-pruner: Router-guided task-agnostic pruning for mixture-of-experts. *arXiv preprint arXiv:2410.12013*, 2024.
- [34] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations (ICLR)*, 2017.
- [35] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning (ICML)*, 2017.
- [36] Mohammed Muqeeth et al. Mone: Mixture of novice experts for efficient MoE inference. *arXiv preprint arXiv:2507.00390*, 2025.
- [37] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, 2nd edition, 2006.
- [38] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [39] Noam Shazeer, Azalia Mirhoseini, Krzysztof Mahdavi, Andrew 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 (ICLR)*, 2017.
- [40] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning (ICML)*, 2021.
- [41] Yutaro Yamada, Ofir Lindenbaum, Sahand Negahban, and Yuval Kluger. Feature selection using stochastic gates. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 10648–10659, 2020.
- [42] Fang Yu, Kun Huang, Meng Wang, Yuan Cheng, Wei Chu, and Li Cui. Width & depth pruning for vision transformers. In *AAAI Conference on Artificial Intelligence*, 2022.
- [43] Jiahui Yu and Thomas Huang. Universally slimmable networks and improved training techniques. In *International Conference on Computer Vision (ICCV)*, 2019.
- [44] Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang. A systematic dnn weight pruning framework using alternating direction method of multipliers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 184–199, 2018.
- [45] Jin Zhou, Xiaokang Wang, Yan Chen, and Lei Zhang. Mask-conditional network compression for structured pruning of vision transformers. *arXiv preprint arXiv:2503.14772*, 2025.
- [46] Michael Zhu and Suyog Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *ICLR Workshop*, 2018.