TOPOLOGICAL SELF-REGULATION IN NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

A neural network learns how to use its channels during training, but where those channels sit is decided before the first gradient is taken, by convention rather than by the task. One part of a network can be starved of channels while another carries channels it barely uses. We ask whether training itself can revise this allocation. Topological Self-Regulation (TSR) treats channel width as a budgeted, revisable part of the model: silent candidate channels are attached throughout the network, the ordinary backward pass reveals which of them would be worth introducing and which incumbents contribute least, and capacity is transferred between them through edits that keep every dependent tensor consistent. A coupling calculus makes such edits valid across residual trunks, depthwise stacks, normalization state and optimizer moments, with exact accounting of the parameters they move. Across four image-classification families and four forecasting backbones, the discovered architectures match the dense reference at roughly fifteen percent fewer parameters in vision and at reductions reaching well over half the network in forecasting. Where the reduction is large, the task-directed allocation outperforms a random one of the same size, and pruning the same reference to the same count shows how much of that is reallocation rather than removal.

1 INTRODUCTION

A network can learn how to use its channels without learning where those channels should be. Conventional widths are chosen before training, often by doubling channels after downsampling or imposing a fixed expansion ratio. Weights then adapt within that allocation, even when a useful feature cannot be expressed at one site and capacity at another contributes little. We ask whether the task loss can also guide a revision of the network's internal dimensions. This requires a connection between redundancy and opportunity. Removal asks what the network would lose, while growth asks what a proposed feature could contribute, which depends on how it enters the computation. We evaluate that opportunity through a candidate whose outgoing weights are zero. Its presence leaves the current function unchanged, yet the loss gradient at those weights need not vanish, and that gradient is the direction along which the feature would enter. The same sensitivity read at an incumbent recovers the activation-gradient product of Taylor pruning (Molchanov et al., 2017).

Topological Self-Regulation uses this connection to revise widths during a discovery run. Its contribution is a common description of growth and removal, coupled to an index representation that makes supported edits valid across dependent tensors. Two levels are analyzed: the released controller reallocates capacity using gradient-derived surrogate scores, and a calibrated reference rule replaces those scores by loss-change bounds that admit a local descent certificate. That certificate belongs to the calibrated variant rather than to every practical edit.

Width allocation occupies a middle ground between post hoc pruning, which asks which components of a chosen architecture can be discarded (Han et al., 2015; Li et al., 2016; He et al., 2017), and architecture search, which evaluates many designs under a separate search procedure or a proxy training budget (Zoph & Le, 2017; Liu et al., 2018; Elsken et al., 2019). TSR instead holds the computational skeleton fixed and lets training redistribute a prescribed resource among its editable groups. When a model family has already fixed its operators and its connectivity, can the task signal improve the allocation of channels inside that family? Asking it this way is what makes the comparison possible, against a dense reference and against the same discovered widths trained

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.

from scratch. That comparison also separates effects that dynamic-architecture results can otherwise confound. A compact network may be better because its widths are better allocated, or because its parameters took a particular sequence of edits and a different optimization trajectory. We therefore treat the online discovery run and the static retraining of its exported shape as different experimental objects, the first measuring what self-regulation costs during learning and the second asking whether the discovered allocation is valuable on its own. Structural validity and resource accounting carry as much weight here as the saliency rule does, as do the empirical controls.

2 Related Work

Cascade-Correlation learns candidate features from residual errors (Fahlman & Lebiere, 1989), and GradMax uses gradient information to initialize network growth (Evci et al., 2022). TSR extends this perspective to candidate ports at interior sites and relates their sensitivity to removal. Its candidate objective is connected to covariance selection, with the centering and norm distinctions derived in section B. Net2Net provides a related foundation for preserving an existing function in a larger parameterization (Chen et al., 2016). Skeletonization and Taylor pruning measure the sensitivity of existing units (Mozer & Smolensky, 1988; Molchanov et al., 2017). Curvature refines the linear estimate when trained gradients become small (LeCun et al., 1989; Hassibi & Stork, 1992). DepGraph addresses the dependencies that constrain channel removal (Fang et al., 2023). Our coupling calculus extends this discipline to bidirectional edits and optimizer state. Width-allocation methods such as MorphNet and AutoSlim are relevant comparisons because they also redistribute capacity within an architectural family (Gordon et al., 2018; Yu & Huang, 2020). TSR makes these decisions from the evolving task gradient. Its edits produce dense networks with different widths, whereas dynamic sparse methods such as RigL reallocate individual connections (Evci et al., 2020). Further discussion appears in section M.

Channel pruning removes dimensions from a trained network (Li et al., 2016; He et al., 2017; Liu et al., 2017), while growth methods add them when new features appear useful. TSR combines the two as budgeted exchanges, so that an addition is always weighed against the cost of what it displaces. Dynamic sparse training reallocates a mask over individual weights and leaves the ambient shapes alone (Evci et al., 2020). Its structured form, SRigL, constrains that mask to whole neurons so the trained network can afterwards be condensed (Lasby et al., 2024), and BilevelPruning learns a channel mask during training and applies it statically at the end (Gao et al., 2024). NeST grows connections and then prunes them in separate phases (Dai et al., 2019), and structured continuous sparsification grows widths under a joint accuracy and sparsity objective (Yuan et al., 2021). These are the closest training-time relatives of TSR. What separates them is what gets decided. A mask chooses which of the existing channels survive. TSR changes how many channels each site has, and can put capacity where the reference had none, editing every dependent tensor and every optimizer state to match. An unstructured sparsity result is context for this work rather than a control, and a structured mask controls the removal half of an exchange without controlling its growth half. MorphNet and AutoSlim redistribute capacity inside a family through regularization or a slimmable supernetwork (Yu et al., 2019), where TSR makes discrete decisions from the evolving task gradient. It also keeps a transparent correspondence between a discovered width vector and a retrainable static control, which is what allows the controls of section 6 to locate the source of an effect. Section M expands these comparisons.

3 Structural Optimization

Figure 1 shows the loop these definitions serve: the coupling groups the calculus recovers, the tests that decide which of them may be edited, the two signals read at a dormant candidate and at an incumbent, and the budgeted exchange that spends one against the other. The rest of this section and section 4 define its quantities in order.

Let $\mathcal{S}$ index channel groups that can be edited independently, with admissible widths $\boldsymbol{n} \in \mathcal{N}$ and weights $\theta \in \Theta(\boldsymbol{n})$. We consider

$$\min_{\boldsymbol{n}\in\mathcal{N}} \min_{\theta\in\Theta(\boldsymbol{n})} \mathcal{L}(\theta, \boldsymbol{n}) \quad \text{subject to} \quad C(\boldsymbol{n}) \le B, \tag{1}$$

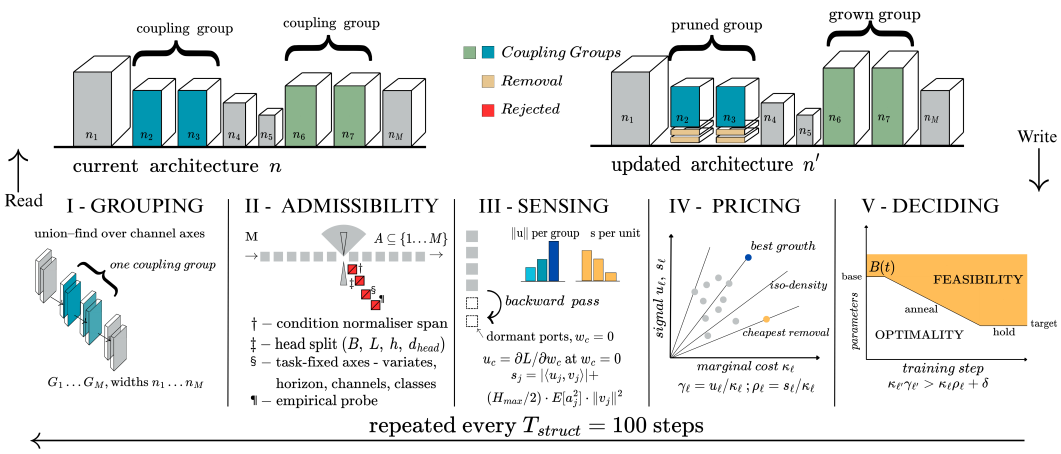


Figure 1: One structural update. Every $T_{\text{struct}} = 100$ optimizer steps the controller reads the current widths, prices every coupling group and applies one edit; θ trains continuously in between. **I** the coupling groups, recovered by union–find over channel axes. **II** the four admissibility tests a group must pass. **III** growth sensed at dormant ports held at $w_c = 0$, removal saliency read at incumbents. **IV** both divided by the marginal parameter cost κ_ℓ. **V** force-prune above the annealed budget $B(t)$, exchange below it. Only the task-fixed axes differ between forecasting and image classification.

where $\mathcal{L} = \mathbb{E}[\ell(f_{\theta,\boldsymbol{n}}(x), y)]$ and C counts deployed parameters. The architecture skeleton and task dimensions remain fixed.

A coupling group G carries an abstract index set I_G. Each occurrence of this index in a parameter or state tensor has a map $\rho_b : I_G \times \{1, \dots, m_b\} \to \mathbb{N}$, where m_b handles layouts such as a flattened spatial block. Residual addition identifies index sets, whereas concatenation preserves separate blocks and their offsets. An admissible edit changes I_G through every represented map and respects divisibility, reshape, normalization, and task-interface constraints. Counting each shared parameter tensor once gives

$$C(\boldsymbol{n}) = \sum_T \prod_a d_{T,a}(\boldsymbol{n}), \tag{2}$$

so the exact resource change is a finite difference of C, not necessarily a sum of isolated marginal costs. Cross terms occur when two edited groups index different axes of one tensor. The complete index calculus and its operator-specific limitations are developed in section H.

A candidate has incoming parameters w_c and a port v_c containing every independent outgoing coordinate through which it can affect retained computation. An admissible silent embedding satisfies $f_{\theta,w_c,0} = f_\theta$. Consequently,

$$\nabla_\theta \mathcal{L}(\theta, w_c, 0) = \nabla_\theta \mathcal{L}(\theta), \qquad \nabla_{w_c} \mathcal{L}(\theta, w_c, 0) = 0, \tag{3}$$

while $\nabla_{v_c} \mathcal{L}$ need not vanish. Silence requires all paths from the candidate into retained outputs to be blocked, including statistical paths. Condition (N) excludes normalizers whose statistics span the edited axis: appending one value to n shifts their shared mean by $q/(n+1)$ and their variance by an $O(1/n)$ amount (equation (H.2)), so a zero outgoing port cannot prevent LayerNorm over the edited feature axis or ordinary GroupNorm from perturbing retained coordinates. Condition (R) requires compatibility with hardcoded reshapes such as multi-head splits. Here "topological" is used in the sense of the topological derivative of shape optimization (Sokolowski & Zochowski, 1999; Novotny & Sokołowski, 2012): it refers to adjoining a graph coordinate and differentiating at its silent embedding, and it is not a coordinate-free graph invariant. When the original weights are stationary, a nonzero u_c makes the silent embedding non-stationary and a vanishing u_c with a nonzero mixed block makes it a strict saddle (Ge et al., 2015) (proposition A.3). Sections A and K supply the proofs and the full analytic and architectural conditions.

4 From Sensitivity to Reallocation

The topological derivative is

$$u_c = \nabla_{v_c} \mathcal{L}(\theta, w_c, v_c)|_{v_c=0}\,. \tag{4}$$

If an independent port coordinate s is tied across uses $q \in \mathcal{U}_s$, reverse-mode differentiation gives

$$(u_c)_s = \mathbb{E}\left[\sum_{q\in\mathcal{U}_s} a_{s,q} g_{s,q}\right]. \tag{5}$$

For a dense consumer this is $\mathbb{E}[a_c g]$. For a convolution, spatial products are summed at their aligned offsets before expectation. Pooling activations and adjoints separately would introduce incorrect cross-position products. Multiple independent consumers form blocks of one joint port, while contributions to a tied coordinate must be summed before its norm is taken. These distinctions are proved in section A.

For disjoint consumer blocks, a block-normalized birth sets each nonzero $v_c^{(r)}$ opposite to $u_c^{(r)}$, with norm ε. It is the steepest first-order direction under $\max_r \left\|v_c^{(r)}\right\|_2 \le \varepsilon$ and has directional change $-\varepsilon \sum_r \left\|u_c^{(r)}\right\|_2$. For one consumer it reduces to the Euclidean direction. Section C gives the coordinate formula and proof, and separates this implemented geometry from the joint Euclidean reference rule below.

For deletion, let $F_j(v)$ be the loss as a function of an incumbent port. Its exact expansion is

$$F_j(0) - F_j(v_j) = -u_j^\top v_j + \int_0^1 (1-s) v_j^\top \nabla^2 F_j(v_j - s v_j) v_j \, ds. \tag{6}$$

The linear term equals the derivative of a scalar channel gate and recovers activation–gradient saliency (Molchanov et al., 2017). At stationarity it can vanish even when deletion is harmful. If the port Hessian operator norm is at most β_j throughout the segment, the loss increase is bounded by $|u_j^\top v_j| + \beta_j \|v_j\|_2^2/2$. A per-position activation mean square need not supply such a β_j for convolution because one weight is tied across positions. Section D gives the proof, a concrete underestimation counterexample, and sufficient shrink-then-remove conditions.

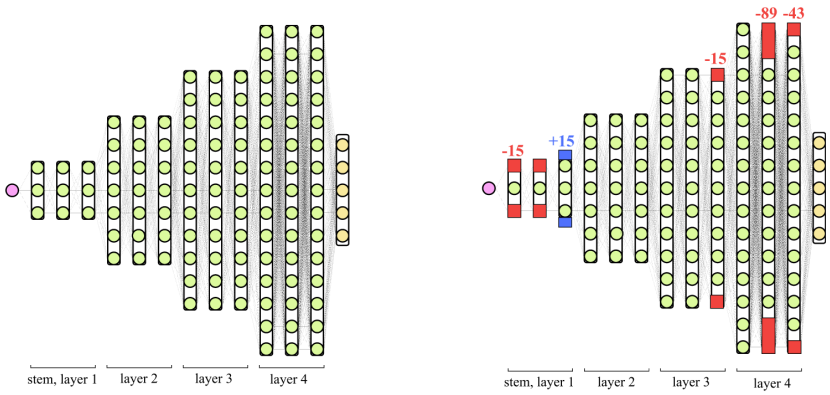
Because dormant incoming weights receive no task gradient (equation (3)), the released controller evaluates fixed random incoming proposals rather than learning them. A principled candidate objective, its connection to Cascade-Correlation, and its scale ambiguity are derived in section B. For comparison with the released surrogate scores, a calibrated variant replaces them with a lower bound G_c on the gain of a Euclidean birth and an upper bound R_j on the damage of a deletion, each carrying an explicit remainder (equation (C.1); sections C and D). An exchange then admits a loss-decrease certificate when

$$\widehat{G}_c - \widehat{R}_j > \delta + \eta, \qquad C(\boldsymbol{n}') \le B, \tag{7}$$

where η covers estimation error, model remainder, and interaction between edits, and the true decrease then exceeds δ (section E). The released controller does not evaluate this test. Under the same calibrated quantities, three further statements hold. A fractional relaxation is stable at a tight budget exactly when a single shadow price separates every growth density from every removal density, which does not characterize indivisible exchanges (section F). A gradient-directed birth has a $\Theta(\sqrt{d})$ first-order advantage over a uniformly random port direction, and a random bank needs exponentially many directions in d to recover a fixed fraction of it (section G). And the number of certified events is bounded by the initial loss and the positive variation of every other transition (section J). None of these three transfers to the released heuristic, whose forced feasibility removals and tolerance-based edits are not certified events. What it does inherit is everything above the certificate: the silent embedding, the port derivative and the consistency of a group edit hold for the released controller as they stand, and the calibrated results delimit what its surrogate scores can be expected to recover.

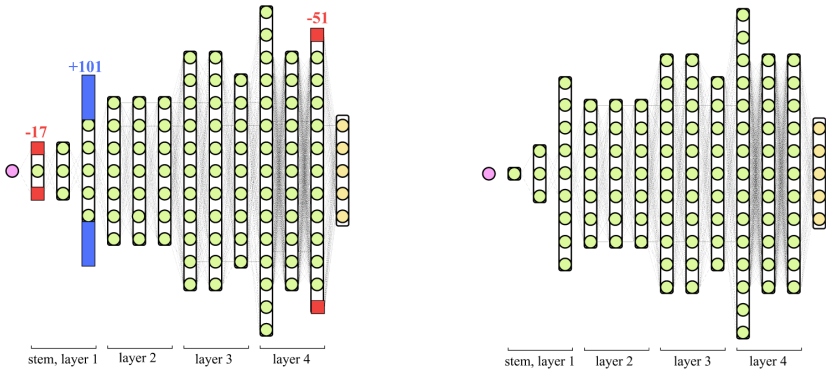
5 Methodology

The implementation traces a model with PyTorch's symbolic tracer, propagates shapes, and couples axes by union–find. A group records tied producer, consumer, normalization, and layout coordinates.



(a) Dense Reference
11.22M parameters

(b) Step 19,500, ceiling at target
9.54M parameters; -15.0%



(c) Step 38,900, last update
9.54M parameters; -15.0%

(d) Discovered Architecture
9.54M parameters

Figure 2: One ResNet-18 reallocating itself on CIFAR-100, at four moments of a single discovery run. Each column is a coupling group in data-flow order, drawn at a height that follows its width; red marks channels removed since the previous panel, blue channels added beyond the dense reference, and the number above a group its net change from the reference. (a) the dense reference; (b) step 19,500, where the annealed ceiling reaches its 85% target; (c) the last structural update, step 38,900; (d) the discovered architecture, 15.0% smaller. Replayed from the archived decision log and checked against the exported checkpoint. One run, followed in full in section T.

Requiring both a producer and consumer excludes inputs and task heads. Current bundles cover the convolutional, linear, residual, normalization, depthwise, and flatten cases exercised by the benchmarks. Concatenation offsets, recurrence, attention internals, arbitrary grouped convolutions, and side paths require further rules. A successful trace alone is not an admissibility proof. Figure 2 shows what the resulting edits do to one network over one run.

Condition (N) admits per-channel BatchNorm and InstanceNorm (Ioffe & Szegedy, 2015; Ulyanov et al., 2016) on their channel axes and excludes feature-spanning LayerNorm and GroupNorm (Ba et al., 2016; Wu & He, 2018). Condition (R) excludes incompatible reshapes and head arithmetic (Vaswani et al., 2017). Input variates, lookback, class count, and forecast horizon remain task-fixed even when an isolated tensor could be sliced. The plastic set this leaves in each backbone, and the one-index probe the forecasting harness runs before attaching candidates, are given in section K.

For each eligible width, the bank appends k random producer rows and zero consumer columns. These rows cost computation despite functional silence, about 2% of the dense parameters and compute at $k = 4$ (section L.1). For disjoint consumer blocks, the released growth score is

$$\bar{q}_c = K^{-1} \sum_{t=1}^{K} \|u_{t,c}\|_2 \,, \qquad \|u_{t,c}\|_2^2 = \sum_r \left\| u_{t,c}^{(r)} \right\|_2^2 . \tag{8}$$

At a fixed state this window is an unbiased estimate of the expected minibatch gradient norm, which upper-bounds the norm of the population gradient. It therefore measures gradient activity rather than

a population direction or a finite loss decrease, and drift and candidate selection add staleness and selection effects (section I).

The removal score averages absolute minibatch gate derivatives and adds an activation-energy penalty,

$$\bar{s}_j = K^{-1} \sum_{t=1}^{K} |u_{t,j}^{\top} v_{t,j}| + \tfrac{1}{2} \widetilde{H} \, \widehat{\mathbb{E}[a_j^2]} \, \|v_j\|_2^2 . \tag{9}$$

Its first term is Taylor saliency and its second is a stability proxy. For tied convolutional weights, the proxy is not generally a Hessian bound because the recorded mean square omits spatial uses and consumer interactions. The controller therefore treats $\bar{q}_c$ and $\bar{s}_j$ only as ranking scores.

Each optimizer step is followed by projecting candidate ports back to zero, and the block directions used at birth come from the most recent minibatch gradient rather than the window average (section L).

Let C_0 be the initial deployed count, B^* the target, and T_{total} the number of training steps. The scheduled ceiling is

$$B(t) = C_0 + \min\{t/t_{\text{end}}, 1\}(B^* - C_0), \qquad t_{\text{end}} = \max\{1, \lfloor \alpha T_{\text{total}} \rfloor\}, \tag{10}$$

with configurable α and released default 0.5. Decisions occur every K steps, so the model may temporarily exceed a newly lowered ceiling. The target must exceed the minimum count reachable under width floors and admissibility restrictions.

The released controller divides each group's best growth and removal scores by its bundle-based marginal parameter estimate,

$$\widetilde{\gamma}_\ell = \frac{\max_c \bar{q}_{\ell c}}{\widetilde{\kappa}_\ell}, \qquad \widetilde{\rho}_\ell = \frac{\min_j \bar{s}_{\ell j}}{\widetilde{\kappa}_\ell}. \tag{11}$$

Above $B(t)$, the feasibility regime orders one removal per prunable group by $\widetilde{\rho}_\ell$ and commits proposals until its projected count reaches the ceiling or the edit cap. These forced removals bypass the quality threshold and may increase loss. Within budget, the controller considers growth using slack, removal below 10^{-3} of the positive group median, then an exchange comparing the selected surrogate values with margin δ. Because growth omits the birth factor ε and curvature, this comparison is a controller rule rather than a predicted loss change. Section L gives the exact cascade and zero-median policy.

Removal and materialization apply one index across every represented parameter, buffer, and optimizer-state array, and export strips all candidates to an ordinary dense network with revised widths (sections L.5 and L.6). The reported count uses real tensor extents, whereas proposal costs use pre-edit bundle marginals that can include dormant extents and omit the cross terms of equation (2) (section H.2).

6 Experimental Setup

Every configuration is evaluated with three arms that share a data pipeline, an optimization schedule, an initialization distribution and a training duration: a dense static reference, the online TSR discovery run, and a static-matched control trained from scratch at the widths that run exported (Liu et al., 2019; Frankle & Carbin, 2018). The comparison that tests the allocation is the static-matched arm against the dense reference. The online arm against its own static twin isolates what changing widths during training contributes, and a random-width control at the same count, run on the forecasting side, tests whether the task signal chose better than an arbitrary nonuniform allocation. On the vision side that test is carried by the from-scratch pruning arm of table 1, an allocation of the same count trained from a fresh initialization under the same recipe and chosen by weight magnitude rather than by chance, which is the stronger comparator. This structure is aligned with the claim in a way that a comparison against a separately tuned architecture with a different search budget would not be, which is why width-search methods such as AutoSlim and MorphNet enter as context rather than as controls (section N).

The vision suite is four backbones on four datasets: ResNet-18, VGG-16-BN, MobileNetV2 and EfficientNet-B0 (He et al., 2016; Simonyan & Zisserman, 2014; Sandler et al., 2018; Tan & Le, 2019) on CIFAR-10, CIFAR-100, Tiny-ImageNet-200 and ImageNet-100 (Krizhevsky et al., 2009; Le et al.,

Table 1: Vision. Ref is the dense model, C2 the discovered architecture retrained from scratch, TSR the online discovery run, and ΔP the change in deployed parameters; the two Δ columns are C2 and TSR minus Ref. The right block prunes that same reference to C2's exact deployed count, each arm measured against the dense reference it was trained beside. ℓ_1, BN, Tay and DG are fine-tuned, ℓ_1/s trained from scratch. Higher is better throughout. One seed per cell, every value measured from the archived checkpoint; the ImageNet-100 cells were added after the pruning sweep (sections O and Q).

			top-1 (%)			Δ		structured pruning, Δ				
Backbone	Data	ΔP	Ref	C2	TSR	C2	TSR	ℓ_1	BN	Tay	DG	ℓ_1/s
ResNet-18	C-10	-15.00	95.01	95.08	94.70	$+0.07$	-0.31	-0.24	-0.09	-0.18	-0.05	$+0.05$
	C-100	-15.01	77.90	77.63	77.62	-0.27	-0.28	-0.60	-0.99	-0.17	-1.32	-1.48
	Tiny-IN	-15.00	65.85	65.33	66.14	-0.52	$+0.29$	-0.87	-0.41	-0.95	-1.70	-1.58
	IN-100	-15.00	82.98	82.72	82.46	-0.26	-0.52	–	–	–	–	–
VGG-16-BN	C-10	-15.04	93.78	93.66	93.46	-0.12	-0.32	$+0.30$	$+0.34$	$+0.61$	$+0.29$	$+0.07$
	C-100	-15.04	73.36	73.32	73.39	-0.04	$+0.03$	$+0.27$	$+0.48$	$+0.34$	$+0.52$	-0.37
	Tiny-IN	-15.00	62.32	62.39	62.07	$+0.07$	-0.25	-0.97	-1.25	-0.17	-1.69	-1.46
	IN-100	-15.00	86.40	86.16	86.06	-0.24	-0.34	–	–	–	–	–
MobileNetV2	C-10	-16.26	90.99	90.78	90.72	-0.21	-0.27	$+0.41$	$+0.20$	$+0.47$	$+0.29$	-1.17
	C-100	-15.29	68.09	66.80	68.19	-1.29	$+0.10$	$+0.08$	-0.25	$+0.24$	-0.23	-1.00
	Tiny-IN	-16.96	55.78	54.46	56.30	-1.32	$+0.52$	$+1.03$	$+0.76$	$+0.94$	$+1.13$	-0.24
	IN-100	-18.13	80.98	79.76	80.34	-1.22	-0.64	–	–	–	–	–
EfficientNet-B0	C-10	-15.01	90.78	90.61	90.21	-0.17	-0.57	$+0.18$	$+0.17$	$+0.23$	$+0.18$	$+0.23$
	C-100	-15.00	68.51	67.99	67.60	-0.52	-0.91	-0.15	-0.07	-0.12	-0.11	-1.21
	Tiny-IN	-15.01	57.65	57.54	58.15	-0.11	$+0.50$	$+0.82$	$+0.77$	$+0.88$	$+0.82$	$+0.58$
	IN-100	-16.41	79.92	80.40	80.42	$+0.48$	$+0.50$	–	–	–	–	–
mean, twelve pruned cells						-0.37	-0.12	$+0.02$	-0.03	$+0.18$	-0.16	-0.63

2015; Deng et al., 2009). The four backbones exercise four different coupling structures, residual, plain sequential, depthwise inverted-residual and compound-scaled MBConv, which is what the calculus is being tested on. The forecasting suite is a dilated residual TCN, PatchTST, iTransformer and TSMixer (Bai et al., 2018; Nie et al., 2023; Liu et al., 2024; Chen et al., 2023) on Weather, Electricity and Traffic (Lai et al., 2018; Zhou et al., 2021) at horizon 96, spanning the variate count from 21 to 862 so that allocation is tested as the share of each network fixed by the task interface grows from small to dominant. Section N gives the dimensions, the splits, the widths each control inherits, and why the excluded alternatives were excluded.

The vision budget is held at 85% of the dense parameter count in every configuration. That is a control rather than a tuned value: the question is where a fixed capacity should be placed, so the constraint is held constant across every architecture and dataset instead of being adjusted per configuration, and 15% is therefore the budget under which allocations are compared, not a reduction the method is claimed to achieve. Sensitivity to the budget itself is characterized on the forecasting side, where the target is swept from 0.95 down to each architecture's measured admissibility floor. The additional control at a fixed budget is structured pruning of the same reference to the same deployed count, which we treat as required rather than optional. Vision reports top-1 accuracy, deployed parameters and forward FLOPs, with top-5 accuracy on the ImageNet-derived sets and forecasting reports test MSE. Candidate-bank computation and the online trajectory are reported separately from exported-model cost. Full splits, preprocessing, optimization settings, budgets and per-cell validity checks are in section N.

7 Vision Results

Table 1 reports the complete vision grid, four backbones on four datasets with all three arms archived in every one of the sixteen cells. A single training run discovers a width vector that carries the dense model's accuracy at 15.0–18.1% fewer parameters: the median difference is 0.22 points, and ten of the sixteen differences are smaller than the range the dense reference itself covers across seeds (section S). No dense pre-training stage, proxy task or search budget beyond that run is involved. Three cells cost more than a point, all of them MobileNetV2, the smallest and most parameter-efficient backbone in the suite, whose inverted residuals leave the least redundancy to reclaim. The largest gain, $+0.48$ points at 16.4% fewer parameters, is EfficientNet-B0 on ImageNet-100. Against structured pruning of the same reference to the same count, the online arm lands within the seed spread of a fine-tuned pipeline that costs 40 more epochs and needs a trained network to start from, and the

Table 2: Forecasting at horizon 96, in the format of table 1 with test MSE in place of accuracy, so lower is better and a negative Δ favors the discovered allocation. ΔP is measured at the knee, the tightest budget whose retrained architecture stays within 3% of the reference. The right block prunes that reference to the same count at that knee, each arm as a percentage of the dense reference of the sweep that produced it; C2* is that sweep's own retraining of C2 and is a separate run from the one on the left. ℓ_1 and Tay are fine-tuned, /s from scratch, and best is the strongest pruning arm, which is all that is archived on Weather. † that row's reference is undertrained, so its column entries share one inflated baseline. Frontier and random-width control in section R.

			test MSE				sweep, % of reference					
Backbone	Data	ΔP	Ref	C2	TSR	Δ	C2*	ℓ_1	Tay	ℓ_1/s	Tay/s	best
TCN	Elec.	-16.8	0.3063	0.3034	0.3016	-0.0029	-0.07	-0.36	-0.26	-2.78	$+4.80$	-2.78
	Traffic	-19.3	0.7286	0.7276	0.7604	-0.0010	-0.14	$+2.42$	$+1.29$	$+1.36$	$+0.23$	$+0.23$
	Weather	-85.6	0.2403	0.2372	0.2724	-0.0031	-1.29	–	–	–	–	$+1.12$
PatchTST	Elec.	-34.0	0.1323	0.1327	0.1329	$+0.0004$	$+0.30$	-0.08	-0.08	$+0.53$	$+0.15$	-0.08
	Traffic	-25.0	0.3719	0.3729	0.3721	$+0.0010$	$+0.27$	-0.75	-0.75	$+0.24$	$+0.16$	-0.75
	Weather	-34.0	0.1518	0.1512	0.1753	-0.0006	-0.40	–	–	–	–	-1.45
iTransformer	Elec.	-35.0	0.1333	0.1341	0.1335	$+0.0008$	$+1.28$	-0.38	-0.38	$+0.45$	$+0.98$	-0.38
	Traffic	-35.0	0.3697	0.3787	0.3749	$+0.0090$	$+2.54$	$+0.22$	$+0.68$	$+2.57$	$+2.65$	$+0.22$
	Weather	-35.0	0.1583	0.1614	0.2083	$+0.0031$	$+1.96$	–	–	–	–	-0.13
TSMixer	Elec.	-86.2	0.1706	0.1563	0.1742	-0.0143	$-8.38^\dagger$	-7.33	-7.80	-8.26	-8.73	-8.73
	Traffic	-81.3	0.4896	0.4985	0.5354	$+0.0089$	$+1.82$	$+1.98$	$+2.04$	$+0.74$	$+0.94$	$+0.74$
	Weather	-45.0	0.1609	0.1582	0.1814	-0.0027	-1.68	–	–	–	–	-1.12
mean, eight Electricity and Traffic knees							-0.30	-0.53	-0.66	-0.64	$+0.15$	–

discovered allocation is ahead of a magnitude-ranked one by 0.26 points on average when neither inherits those weights, a margin most cells do not separate from the seed (section Q). Section O gives the per-cell figures.

What the controller exports is a transferable architecture rather than a trajectory-dependent solution: the online arm's median difference against its own retrained twin is -0.03 points. That comparison never involves the dense arm, so it is unaffected by how well the reference was trained, and it says that searching and training at once costs essentially nothing against retraining the same shape from scratch. The controller optimizes the resource it is given. Under the parameter budget forward FLOPs fall in seven of the fifteen measured cells and rise in eight, because parameters concentrate in late, narrow-spatial layers while computation concentrates in early, wide ones. That dependence is actionable rather than incidental, because κ is defined against any additive cost: re-running EfficientNet-B0 and MobileNetV2 under a FLOP budget, with the same controller, schedule and recipe, delivers a 13–15% compute reduction in all four cells at one seed each, including the one where a parameter budget had instead raised FLOPs by 40.2% and where the FLOP-budgeted model now matches the dense reference exactly (section P).

8 Forecasting Results

Table 2 reports the forecasting campaign in the format of table 1, with test MSE in place of top-1 accuracy. One difference in protocol matters for reading it: each configuration is swept across a range of budgets and the table reports its *knee*, the tightest budget whose retrained architecture stays within 3% of the dense reference, a tolerance fixed before the results were read. Its right-hand block is structured pruning at those same knees, on which the discovered and the pruned allocations are level (section Q). Compression at that tolerance spans an order of magnitude, from 17% to 86% fewer parameters, and the sweep locates that figure for each configuration rather than assuming it. The share left plastic by the admissibility conditions of section K orders the backbones and the variate count orders the datasets, so the same TCN reaches a knee at 17% on one dataset and 86% on another. The knee is therefore a measurement of how much capacity the task uses, one that no single fixed budget would have produced. On three Weather cells it is the tightest budget swept, so the reduction reported there is a lower bound (section R).

The random-width control asks whether the task signal chose the allocation or whether any allocation of that size would have served. At the knee the discovered allocation has lower test MSE than a random one of the same parameter count in all eight Electricity and Traffic cells, by 2.3–5.2% of the metric on TSMixer and the TCN, well clear of the seed spread (section S). Where little capacity has

to move, or where the task does not use what is being moved, the margins fall inside that spread, which is what a controller whose only objective is the resource constraint should produce.

Two readings depend on keeping the arms separate. The online arm is measured at its search-converged architecture, reached early on Weather, so the anneal applies most of its removals to a network that has already converged and its error sits above the reference. The retrained arm on the identical architecture lands inside the tolerance at every Weather knee, which localizes that cost to the schedule rather than to the search. In the other direction, TSMixer on Electricity places the retrained arm 8.4% below its own dense reference while the online run only matches it. That reference is undertrained: its recipe is the worst of six settings swept, and a better-tuned dense arm reaches a test MSE of 0.1519, against which the retrained model is 2.9% worse rather than 8.4% better. We exclude the cell rather than count it as a gain. A two-arm design would have reported it as the headline number (section R).

9 Discussion

The question posed in section 1, whether the task signal can improve the allocation of channels within a fixed model family, is answered here for allocation under a budget. TSR discovers allocations inside a single training run rather than through a separate search. Retrained from scratch, they hold the dense model's accuracy to a median of 0.22 points at 15 to 18% fewer parameters in vision, and the run that discovered them ends 0.28 below the dense model on its own. In forecasting the retrained allocation reaches 17% to 86% fewer parameters inside a fixed tolerance. Parity at a reduced budget is the claim, and it is a measured one, since every arm is scored from its archived checkpoint.

The static-matched arm reproduces the online arm's accuracy to a median of 0.03 points over the sixteen vision cells, so what TSR exports is a width vector that trains on its own. The random-width control adds a second distinction, between a task-directed allocation and an arbitrary one of the same size, which the from-scratch pruning arm carries in vision with a stronger comparator and a more modest margin. Where the budget forces a large reallocation of capacity the task needs, the task-directed allocation beats an arbitrary one of the same size by more than the seed spread, and elsewhere the two are indistinguishable, which is what a controller whose only objective is the resource constraint should produce. The released controller carries no accuracy objective. It enforces structural validity exactly and spends the budget where the sensing rule finds capacity worth most, and the descent certificate belongs to its calibrated form. Exported networks that are smaller and not worse are therefore the designed outcome, and the design leaves a direct route open. A small accuracy term in the exchange rule, or a protection interval for recently promoted channels, would push toward allocations that are better, and neither touches the coupling machinery that makes the edits valid. What can be reallocated is set by the coupling structure and the task interface rather than by the sensing rule.

Three boundaries remain. Most cells are single-seed, with the repeats of section S bounding the spread, and the controller's own settings (k, K, ε, δ) were fixed across the campaign rather than swept. The vision budget is a fixed control rather than a swept quantity, so sensitivity to the budget is characterized on the forecasting side. The third is coverage rather than a missing control. Structured pruning at the matched count runs in twelve of the sixteen vision cells, and the four ImageNet-100 cells are not yet covered (section Q). The longer forecast horizon is excluded for the reason the Electricity cell is, that an arm enters the tables only when its controls are complete.

10 Conclusion

Topological Self-Regulation treats channel width as a budgeted, revisable part of a model and gives training the means to revise it. Its two components are separable. A coupling calculus makes a proposed edit valid across every dependent tensor, with exact accounting of the parameters it moves. A sensing rule reads a topological derivative from the ordinary backward pass at a silent candidate and pairs it with a removal saliency at incumbents. The allocation it exports trains on its own, beats a random one of the same size where the budget forces capacity to move, and can be spent on parameters or on compute. A neural network, we believe, should be allowed to decide its own structure.

AI USE STATEMENT

In this work, we used generative AI tools to edit the manuscript for readability and to polish the writing. We have not used generative AI tools to develop the theoretical framework, formulate or prove mathematical claims, propose or refine hypotheses, design or provide feedback on the methodology or experiments, implement methods, or interpret results; generating synthetic data, translation, dataset cleaning, and qualitative or thematic data analysis are not applicable to this work. We have reviewed all AI-assisted work: every AI-edited passage was checked by the authors against the underlying results and revised where needed. We take responsibility for the final content of this work, including text produced with the aid of generative AI.

REFERENCES

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Si-An Chen, Chun-Liang Li, Sercan O Arik, Nathanael Christian Yoder, and Tomas Pfister. TSMixer: An all-MLP architecture for time series forecast-ing. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=wbpxTuXgm0.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. In *International Conference on Learning Representations*, 2016. URL http://arxiv.org/abs/1511.05641.

Xiaoliang Dai, Hongxu Yin, and Niraj K. Jha. NeST: A neural network synthesis tool based on a grow-and-prune paradigm. *IEEE Transactions on Computers*, 68(10):1487–1497, 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019. URL http://jmlr.org/papers/v20/18-598.html.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International conference on machine learning*, pp. 2943–2952. PMLR, 2020.

Utku Evci, Bart van Merrienboer, Thomas Unterthiner, Fabian Pedregosa, and Max Vladymyrov. Gradmax: Growing neural networks using gradient information. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=qjN4h_wwUO.

Scott Fahlman and Christian Lebiere. The cascade-correlation learning architecture. *Advances in neural information processing systems*, 2, 1989.

Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16091–16101, 2023. doi: 10.1109/CVPR52729.2023.01544.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Kenji Fukumizu and Shun-ichi Amari. Local minima and plateaus in hierarchical structures of multilayer perceptrons. *Neural networks*, 13(3):317–327, 2000.

Peiran Gao, Eric Trautmann, Byron Yu, Gopal Santhanam, Stephen Ryu, Krishna Shenoy, and Surya Ganguli. A theory of multineuronal dimensionality, dynamics and measurement. *BioRxiv*, pp. 214262, 2017.

Shangqian Gao, Yanfu Zhang, Feihu Huang, and Heng Huang. Bilevelpruning: Unified dynamic and static channel pruning for convolutional neural networks. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16090–16100. IEEE, 2024.

Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Conference on learning theory*, pp. 797–842. PMLR, 2015.

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 *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1586–1595. IEEE, 2018.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.

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*, pp. 770–778, 2016.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *2017 IEEE international conference on computer vision (ICCV)*, pp. 1398–1406. IEEE, 2017.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International conference on learning representations*, 2021.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMlR, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pp. 95–104, 2018.

Mike Lasby, Anna Golubeva, Utku Evci, Mihai Nica, and Yani Ioannou. Dynamic sparse training with structured sparsity. In *International Conference on Learning Representations*, volume 2024, pp. 32982–33010, 2024.

Yann Le, Xuan Yang, et al. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *International conference on learning representations*, volume 2024, pp. 11116–11140, 2024.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *2017 IEEE international conference on computer vision (ICCV)*, pp. 2755–2763. Ieee, 2017.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rJlnB3C5Ym`.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJGCiw5gl.

Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. *Advances in neural information processing systems*, 1, 1988.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Jbdc0vTOcol.

Antonio André Novotny and Jan Sokołowski. *Topological derivatives in shape optimization*. Springer Science & Business Media, 2012.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 4510–4520. Ieee, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Jan Sokolowski and Antoni Zochowski. On the topological derivative in shape optimization. *SIAM journal on control and optimization*, 37(4):1251–1272, 1999.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PmLR, 2019.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Roman Vershynin. High-dimensional probability. *Cambridge Series in Statistical and Probabilistic Mathematics*, 47, 2019.

James G. Wendel. Note on the gamma function. *American Mathematical Monthly*, 55:563, 1948. URL https://api.semanticscholar.org/CorpusID:124319579.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ju_Uqw384Oq.

Lemeng Wu, Dilin Wang, and Qiang Liu. Splitting steepest descent for growing neural architectures. *Advances in neural information processing systems*, 32, 2019.

Lemeng Wu, Bo Liu, Peter Stone, and Qiang Liu. Firefly neural architecture descent: a general approach for growing neural networks. *Advances in neural information processing systems*, 33: 22373–22383, 2020.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

Jiahui Yu and Thomas Huang. Autoslim: Towards one-shot architecture search for channel numbers, 2020. URL https://openreview.net/forum?id=H1gz_nNYDS.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. 2019. Publisher Copyright: © 7th International Conference on Learning Representations, ICLR 2019. All Rights Reserved.; 7th International Conference on Learning Representations, ICLR 2019 ; Conference date: 06-05-2019 Through 09-05-2019.

Xin Yuan, Pedro Savarese, and Michael Maire. Growing efficient deep networks by structured continuous sparsification. In *International Conference on Learning Representations*, 2021.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 11121–11128, 2023.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

Barret Zoph and Quoc Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=r1Ue8Hcxg.

APPENDIX CONTENTS

A Silent Embeddings and the Port Derivative . . . 16
B Candidate Learning and the Covariance Connection . . . 17
C Growth Direction and Finite-Step Gain . . . 18
D Removal Saliency and Controlled Deletion . . . 19
E A Descent Guarantee for the Calibrated Variant . . . 21
F The Equimarginal Condition and Its Discrete Boundary . . . 22
G Directed Birth and Random Port Directions . . . 23
H Coupling, Normalization, and State-Preserving Edits . . . 24
H.1 Why channel-spanning normalization is different . . . 25
H.2 Exact parameter accounting . . . 25
H.3 Optimizer state and dormancy . . . 26
I Windowed Estimation and Direction Error . . . 26
J Certified Event Counts and Practical Budget Enforcement . . . 28
K Admissibility Conditions . . . 28
K.1 Group discovery and eligibility . . . 29
K.2 Condition (N) and normalization span . . . 29
K.3 Condition (R) and hardcoded reshapes . . . 29
K.4 Task-fixed axes and the empirical probe . . . 30
K.5 Time-series axes and implemented cases . . . 30
L Implementation and Structural-Update Procedure . . . 31
L.1 Candidate-bank layout and dormancy . . . 31
L.2 Measurements between edits . . . 31
L.3 Budget schedule and the two controller regimes . . . 32
L.4 Released ranking and the calibrated exchange variant . . . 33
L.5 Pruning by index . . . 33
L.6 Materialization and candidate replacement . . . 33
L.7 Optimizer state, export, and decision records . . . 34
M Further Relation to Earlier Work . . . 34
N Experimental Protocol Details . . . 35
O Vision Result Details and Artifact Audit . . . 37
P Budgeting Compute Instead of Parameters . . . 38
Q Structured Pruning at a Matched Count . . . 39
R Forecasting Result Details . . . 39
S Seed Replicates . . . 40
T Case studies: two discovery runs . . . 41

A SILENT EMBEDDINGS AND THE PORT DERIVATIVE

The proofs distinguish a structural identity from a differentiable loss model. The identity concerns a directed acyclic computation graph with channel-indexed tensors. A silent candidate is admissible when setting its independent port coordinates to zero leaves every retained output unchanged for every setting of its incoming parameters in the candidate family. This requires all paths into the retained computation to be blocked, including paths through statistics or gates. Index consistency alone is not sufficient.

For derivatives, fix the current weights and let ξ denote a sample, or a complete minibatch if the forward map uses batch statistics. Assume differentiation can be interchanged with expectation. A sufficient local condition is almost-sure differentiability with an integrable envelope for the relevant derivatives. All second-order results below require twice differentiable composed losses with the stated segment bounds. A cubic remainder additionally requires a Lipschitz Hessian. ReLU networks need these conditions checked along the perturbation under consideration, or an appropriate nonsmooth bound in their place.

Lemma A.1 (Embedding and gradient structure)**.** Suppose that an admissible candidate affects the retained computation only through port terms that vanish at $v_c = 0$, and that Condition (N) holds. Then $f_{\theta,w_c,0} = f_\theta$. At a differentiable point,

$$\nabla_\theta \mathcal{L}(\theta, w_c, 0) = \nabla_\theta \mathcal{L}(\theta), \qquad \nabla_{w_c} \mathcal{L}(\theta, w_c, 0) = 0.$$

The outgoing gradient need not vanish.

Proof. Order the operators topologically. At the input, retained coordinates are unchanged. Suppose this remains true up to a given operator. At a linear consumer, every added contribution into a retained coordinate contains a zero port coefficient, so the retained output is unchanged. Channelwise nonlinearities preserve that equality. A permitted normalization reads the same retained values with the same state, so it also returns the same retained values. Other operators satisfy the asserted path-blocking condition. Induction reaches the task output and proves equality of the two functions.

The identity holds as a function of both θ and w_c in the local admissible parameterization. Differentiating it with respect to θ gives the first gradient equality. The right side does not depend on w_c, so differentiating with respect to w_c gives zero. No identity asserts constancy with respect to v_c away from zero, which is why its derivative can be nonzero. These are real-arithmetic identities. Different floating-point kernels after a shape change need not produce bitwise identical results. □

Proposition A.2 (Gradient at a tied port)**.** At the silent point, let s index an independent port parameter and let $\mathcal{U}_s$ be its uses in the graph. If use q contributes $v_s a_{s,q}$ to a preactivation with loss derivative $g_{s,q}$, then

$$(u_c)_s = \mathbb{E}\left[\sum_{q \in \mathcal{U}_s} a_{s,q} g_{s,q}\right]. \tag{A.1}$$

For every fixed d, $\frac{d}{dt}\mathcal{L}(td)|_{t=0} = u_c^\top d$.

Proof. Reverse-mode differentiation sums a parameter's contribution at each place where that parameter is used. At a use q, the derivative of $v_s a_{s,q}$ with respect to v_s, with the local inputs held fixed, is $a_{s,q}$. Multiplication by the downstream adjoint gives $a_{s,q} g_{s,q}$. Summing over uses and then taking expectation proves equation (A.1). Any additional computational paths are already represented in the adjoints and other uses of the same parameter. The scalar chain rule along $v_c = td$ then gives the directional identity.

For a dense layer there is one use per output coordinate per example. For a convolution, if $s = (o, \Delta)$ denotes an output channel and kernel offset, then

$$(u_c)_{o,\Delta} = \mathbb{E}\left[\sum_p a_c[p + \Delta] g_o[p]\right].$$

Each spatial product belongs to a single weight use. Pooling the two factors separately would introduce cross-position products and discard the required alignment. In particular, $a = (1, -1)$ and $g = (1, -1)$ have sum of products 2 but product of sums 0. With several consumers, their coordinate gradients form a joint port vector. Their squared norms add only when their parameter coordinates are disjoint. Tied coordinates must first sum their contributions. □

If the original weights are stationary and $u_c \neq 0$, the embedding is not stationary because moving the port along $-u_c$ is descending. If $u_c = 0$ but the mixed block $\nabla_{v_c}\nabla_{w_c}\mathcal{L}$ is nonzero, the enlarged Hessian has a negative direction and the embedding is a strict saddle. If both blocks vanish, incoming candidate directions are flat to second order, although port curvature can remain. The following proposition makes this escape hierarchy precise.

Proposition A.3 (Escape from a stationary embedding)**.** Suppose the original weights are stationary and the extended loss is twice differentiable. If $u_c \neq 0$, the silent embedding is not stationary. If $u_c = 0$ but $M = \nabla_{v_c}\nabla_{w_c}\mathcal{L} \neq 0$, it is a strict saddle. If both vanish, the Hessian is flat in the w_c directions, but the port Hessian can still have positive or negative curvature.

Proof. If $u_c \neq 0$, the direction with port displacement $-u_c$ and all other coordinates fixed has derivative $-\|u_c\|^2 < 0$. If $u_c = 0$, all first derivatives vanish by lemma A.1 and the assumed stationarity of the original weights. Since the loss at a silent port is independent of w_c, the Hessian restricted to (w_c, v_c) has the form

$$H = \begin{pmatrix} 0 & M^\top \\ M & H_{vv} \end{pmatrix}.$$

Choose b with $M^\top b \neq 0$ and set $a = -sM^\top b$ for $s > 0$. The quadratic form along (a, b) is $-2s\left\|M^\top b\right\|^2 + b^\top H_{vv} b$, which is negative for sufficiently large s. A real symmetric matrix with a negative quadratic form has a negative eigenvalue. If $M = 0$, the block becomes diagonal with a zero incoming block. This says nothing about the sign of H_{vv}. The mixed-block mechanism is related to the redundant-unit saddle geometry studied by Fukumizu & Amari (2000). □

B Candidate Learning and the Covariance Connection

Equation (3) means that dormant incoming weights do not learn from the task loss. A principled feature-learning objective is

$$J(w) = \tfrac{1}{2}\|u(w)\|_2^2, \qquad \nabla_w J = \mathbb{E}\left[(g^\top u(w))\nabla_w a_w\right], \tag{B.1}$$

under a scale constraint or curvature normalization. With squared loss at a linear output, $u(w) = \mathbb{E}[a_w r]$. It equals the centered covariance $\mathrm{Cov}(a_w, r)$ when $\mathbb{E}[r] = 0$, connecting the construction to Cascade-Correlation. The Euclidean norm used here differs from its original ℓ_1 aggregation for multiple outputs. The released controller does not optimize equation (B.1); it evaluates fixed random incoming proposals. The remainder of this section derives the objective, the centering condition, and the scale ambiguity.

Fix the real network while evaluating a candidate. For a simple port, write $u(w) = \mathbb{E}[a_w g]$ and $J(w) = \frac{1}{2}\|u(w)\|_2^2$. At the silent point the real network's adjoint g is independent of the candidate's incoming weights. Differentiation gives

$$\nabla_w J(w) = \left(\mathbb{E}[g\nabla_w a_w^\top]\right)^\top u(w) = \mathbb{E}\left[(g^\top u(w))\nabla_w a_w\right]. \tag{B.2}$$

For multiple consumers, the corresponding contributions are added over the independent port coordinates. This calculation differentiates a correlation with a frozen adjoint. It does not require differentiating through an optimization trajectory of the real network, but it does require a candidate objective beyond the task loss. A detached port gradient read from an ordinary backward pass is not itself a training update for w.

There is a scale ambiguity in optimizing J. For a positively homogeneous feature family, multiplying a_w by $q > 0$ multiplies J by q^2 without changing the represented direction after inverse rescaling of the port. Candidate learning must therefore constrain its parameter family or feature scale, or optimize a curvature-normalized gain. For example, $\mathbb{E}[a_w^2] \leq 1$ bounds the score whenever $\mathbb{E}\|g\|^2 < \infty$, by Cauchy–Schwarz. This is a condition on a well-posed objective, not a claim that a particular constraint has been implemented.

Proposition B.1 (Relation to Cascade-Correlation)**.** For squared loss at a linear output port, let $r = f_\theta - y$. Then $u(w) = \mathbb{E}[a_w r]$. The centered covariance vector is

$$q(w) = \mathrm{Cov}(a_w, r) = u(w) - \mathbb{E}[a_w]\mathbb{E}[r].$$

When $\mathbb{E}[r] = 0$, the two vectors agree. The centered objective $\|q(w)\|_1$ agrees, up to sample normalization and residual sign, with the original Cascade-Correlation score. The Euclidean objective $\|u(w)\|_2$ is generally different for multiple outputs.

Proof. For $\ell = \frac{1}{2}\|f - y\|_2^2$, differentiation with respect to the output gives r, so proposition A.2 gives $u = \mathbb{E}[a_w r]$. Expanding the centered product yields

$$\mathbb{E}[(a_w - \mathbb{E}[a_w])(r - \mathbb{E}[r])] = \mathbb{E}[a_w r] - \mathbb{E}[a_w]\mathbb{E}[r].$$

If an unconstrained output bias is stationary under squared loss, its gradient $\mathbb{E}[r]$ vanishes, giving one sufficient condition for equality. The original score sums the absolute centered covariances over output coordinates (Fahlman & Lebiere, 1989). Consequently, for a scalar output, maximizing its absolute value or its square gives the same ordering under the centering condition. For multiple outputs, ℓ_1 and ℓ_2 need not agree. Covariance vectors $(2, 0)$ and $(1.1, 1.1)$ are ranked in opposite orders by these two norms.

The norm difference also has a geometric interpretation. If outgoing directions satisfy $\|d\|_\infty \le 1$, then $\max_d(-q^\top d) = \|q\|_1$, attained by choosing each coordinate of d opposite to the corresponding sign of q. A Euclidean constraint instead gives $\|q\|_2$. Thus the two candidate scores correspond to different port geometries even when they use the same centered signal. □

C Growth Direction and Finite-Step Gain

Proposition C.1 (Block-normalized implemented birth)**.** Partition a port and its gradient into m disjoint consumer blocks. Define

$$\|d\|_{2,\infty} = \max_{r \le m} \left\|d^{(r)}\right\|_2, \qquad \|u\|_{2,1} = \sum_{r \le m} \left\|u^{(r)}\right\|_2.$$

The minimum of $u^\top d$ over $\|d\|_{2,\infty} \le 1$ is $-\|u\|_{2,1}$. It is attained by $d^{(r)} = -u^{(r)}/\left\|u^{(r)}\right\|_2$ for every nonzero block, with an arbitrary feasible value on zero blocks. Choosing zero on zero blocks gives Euclidean norm $\sqrt{m_+}$, where m_+ is the number of nonzero blocks.

Proof. Blockwise Cauchy–Schwarz gives

$$u^\top d = \sum_r (u^{(r)})^\top d^{(r)} \ge -\sum_r \left\|u^{(r)}\right\|_2 \left\|d^{(r)}\right\|_2 \ge -\sum_r \left\|u^{(r)}\right\|_2.$$

The stated direction attains both inequalities. Its nonzero blocks have unit Euclidean norm and are disjoint, so its squared Euclidean norm is m_+. Scaling by the implementation's ε gives first-order minibatch change $-\varepsilon\|u\|_{2,1}$ and step norm $\varepsilon\sqrt{m_+}$. □

This proposition describes the port-opening geometry used by the released edit routine. Candidate selection nevertheless uses the Euclidean block-concatenated magnitude $\|u\|_2$, averaged as in equation (8). The two orderings can differ when candidates distribute their gradients differently across consumers. They agree when the port has one consumer block.

For comparison with the released surrogate, a calibrated variant uses the joint Euclidean direction $d_c = -u_c/\|u_c\|_2$ and the quantities

$$G_c(t) = t\|u_c\|_2 - \tfrac{1}{2}t^2 h_c, \qquad R_j = |u_j^\top v_j| + \tfrac{1}{2}\beta_j\|v_j\|_2^2, \tag{C.1}$$

where h_c is directional curvature and β_j bounds the incumbent port Hessian over deletion. These are not the scores in equations (8) and (9). They define a stronger reference rule whose remainder bounds are proved here and in section D. If the directional second derivative along the Euclidean birth is at most $\beta > 0$ for $0 \le t \le r$, then the loss change is at most $-t\|u\|_2 + \beta t^2/2$, and the step $t = \min\{\|u\|_2/\beta, r\}$ is descending. A local curvature h_c yields the quadratic model in equation (C.1) but is not a segment bound; Hessian-Lipschitz control or a trust region is required to turn it into a finite-step statement.

Theorem C.2 (Steepest birth and controlled gain)**.** Let $F(v) = \mathcal{L}(\theta, w_c, v)$ be differentiable at zero with $u = \nabla F(0) \ne 0$. The unique minimizer of $u^\top d$ on the Euclidean unit ball is $d^* = -u/\|u\|$. If F is twice differentiable along td^* for $0 \le t \le r$ and its directional second derivative is at most $\beta > 0$, then

$$F(td^*) - F(0) \le -t\|u\| + \tfrac{1}{2}\beta t^2. \tag{C.2}$$

The step $t = \min\{\|u\|/\beta, r\}$ guarantees the nonnegative gain $t\|u\| - \beta t^2/2$.

Proof. Cauchy–Schwarz gives $u^\top d \geq -\|u\| \|d\| \geq -\|u\|$. Equality requires d to have unit norm and be a negative multiple of u, giving uniqueness. For $\varphi(t) = F(td^*)$, the fundamental theorem of calculus applied twice gives

$$\varphi(t) - \varphi(0) = t\varphi'(0) + \int_0^t (t-s)\varphi''(s)\, ds.$$

Substitute $\varphi'(0) = -\|u\|$ and $\varphi''(s) \leq \beta$. The integral is at most $\beta t^2/2$, proving the bound. Minimizing this upper quadratic over $[0, r]$ gives the stated step. Since that step is at most $\|u\|/\beta$, its gain is at least $t\|u\|/2 > 0$ when $r > 0$. □

The local curvature $h = (d^*)^\top \nabla^2 F(0) d^*$ need not be an upper bound away from zero. If $h > 0$, completing the square in the local model gives

$$-t\|u\| + \tfrac{1}{2}ht^2 = \tfrac{h}{2}\left(t - \frac{\|u\|}{h}\right)^2 - \frac{\|u\|^2}{2h}. \tag{C.3}$$

This proves the model optimum used in the main text, but not a finite-step guarantee at an arbitrarily large t. If the Hessian is M-Lipschitz along the segment, then

$$F(td^*) - F(0) = -t\|u\| + \tfrac{1}{2}ht^2 + \mathcal{R}(t), \tag{C.4}$$

$$|\mathcal{R}(t)| \leq \int_0^t (t-s)Ms\, ds = \frac{Mt^3}{6}. \tag{C.5}$$

With only continuous second derivatives, the remainder is $o(t^2)$ rather than necessarily $O(t^3)$. If $h \leq 0$, the expression $\|u\|^2/(2h)$ is not an admissible positive gain. A trust-region model or a positive upper curvature bound avoids that division.

For a linear injection $h_x + A_x v$ into a downstream loss ψ_x, the port Hessian is $\mathbb{E}[A_x^\top \nabla^2 \psi_x A_x]$. Convexity of the loss in the final network output does not imply convexity of this composed function. Indeed, if $\psi_x(h) = \ell(f(h), y)$, then

$$\nabla_h^2 \psi_x = J_f^\top \nabla_f^2 \ell\, J_f + \sum_o \frac{\partial \ell}{\partial f_o} \nabla_h^2 f_o.$$

A Gauss–Newton approximation keeps the first term, which is positive semidefinite for convex output losses. It need not be positive definite or upper-bound the full Hessian. Damping can make an approximate curvature positive, but does not on its own certify a loss bound.

Finally, rescale a fixed candidate activation by $q > 0$ and its corresponding port by $1/q$. In the new coordinates, $u' = qu$ and $h' = q^2 h$, so $\|u'\|^2/(2h') = \|u\|^2/(2h)$. This explains the benefit of curvature normalization under scalar feature rescaling. A fixed Euclidean trust radius or birth step must be transformed as well to preserve the same function-space edit. No invariance under arbitrary port reparameterizations is asserted.

D Removal Saliency and Controlled Deletion

Theorem D.1 (Removal with integral remainder)**.** Let $F_j(v)$ be the loss with an incumbent port set to v, and write v_j for its current value and $u_j = \nabla F_j(v_j)$. Suppose F_j is twice differentiable along $v_j - sv_j$ for $s \in [0, 1]$. Then

$$F_j(0) - F_j(v_j) = -u_j^\top v_j + \int_0^1 (1-s) v_j^\top \nabla^2 F_j(v_j - sv_j) v_j\, ds. \tag{D.1}$$

If $\left\|\nabla^2 F_j\right\|_{\mathrm{op}} \leq \beta_j$ there, then

$$|F_j(0) - F_j(v_j)| \leq |u_j^\top v_j| + \tfrac{1}{2}\beta_j \|v_j\|^2.$$

Proof. Set $\varphi(s) = F_j((1-s)v_j)$. Its first derivative at zero is $-u_j^\top v_j$, and its second derivative is $v_j^\top \nabla^2 F_j((1-s)v_j)v_j$. Taylor's integral identity between 0 and 1 gives equation (D.1). The operator-norm assumption bounds the absolute integrand by $(1-s)\beta_j \|v_j\|^2$. Integration gives the factor $1/2$, and the triangle inequality gives the claim. The same expression is consequently an upper bound on loss increase. □

For a scalar activation a_j entering a single downstream preactivation through $a_j v$, assume that activation is fixed as v is varied. Then $F_j(v) = \mathbb{E}[\psi_x(h_x + a_j v)]$, with h_x collecting the other inputs. Its Hessian is

$$\nabla^2 F_j(v) = \mathbb{E}[a_j^2 \nabla^2 \psi_x(h_x + a_j v)].$$

If $\left\|\nabla^2 \psi_x\right\|_{\mathrm{op}} \le H_{\max}$ throughout the segment, the triangle inequality for the operator norm yields $\beta_j \le H_{\max}\mathbb{E}[a_j^2]$. For a general linear injection $A_x v$, the corresponding bound is $H_{\max}\mathbb{E}\left\|A_x\right\|_{\mathrm{op}}^2$. Multiple coupled consumers, especially at different depths, should instead use the full port Hessian unless their cross terms have been bounded. A single activation-energy factor cannot be assumed to cover those interactions. Repetition of one tied parameter across positions already causes this problem even without multiple consumers. If a shared scalar port v injects activations $a_1, \ldots, a_P$ into P coordinates of a squared-loss output, its port Hessian is $\sum_p a_p^2$, while the activation mean square usually recorded by a hook is $P^{-1}\sum_p a_p^2$. These differ by P when each use contributes independently. For overlapping convolutional taps and multiple port coordinates, $\left\|A_x\right\|_{\mathrm{op}}^2$ also includes correlations between uses. The valid bound depends on the complete injection operator, not on a per-channel mean square alone.

A two-position example gives an exact failure of the mean-square proxy as a removal certificate. Let $a = (1, 1)$, $\psi(h) = \frac{1}{2}\left\|h - (1,1)\right\|^2$ and $h = av$ with current $v = 1$, so that

$$F(v) = \tfrac{1}{2}\left\|av - (1,1)\right\|^2 = (v-1)^2, \qquad F'(1) = 0, \qquad F'' = a^\top a = 2. \tag{D.2}$$

The port gradient vanishes and $H_{\max} = 1$ for the downstream loss, while the per-position activation mean square is $\frac{1}{2}(1^2 + 1^2) = 1$. The two candidate bounds on the cost of setting $v = 0$ therefore disagree,

$$\tfrac{1}{2}\, H_{\max}\mathbb{E}[a^2]\ \|v\|_2^2 = \tfrac{1}{2}, \qquad \tfrac{1}{2}\,(a^\top a)\ \|v\|_2^2 = 1, \qquad F(0) - F(1) = 1, \tag{D.3}$$

and only the second holds. No measurement noise or third-order remainder is involved in this discrepancy.

Proposition D.2 (Gate saliency)**.** For a channel whose effects on the loss are exactly its linear consumer contributions, let a_j denote its activation, including position indices. Then

$$u_j^\top v_j = \mathbb{E}\left[\sum_p a_j[p]\frac{\partial \ell}{\partial a_j[p]}\right]. \tag{D.4}$$

Equivalently, this is the derivative of the loss with respect to a scalar gate multiplying that channel, evaluated at gate value one.

Proof. A gate α changes all contributions from $a_j v_j$ to $\alpha a_j v_j$. Varying the gate is therefore the same as varying the port along $v = \alpha v_j$, whose loss derivative at $\alpha = 1$ is $u_j^\top v_j$. Differentiating the activation representation instead gives the right side of equation (D.4). Equality follows because both calculations differentiate the same scalar path. Absolute value after expectation gives the stated first-order saliency. An average of absolute samplewise products is a different quantity, so the reduction order must match the intended criterion. □

The released statistic takes the absolute value on each minibatch and then averages it over the structural window. If v_j is fixed over that window and $Z_t = u_{t,j}^\top v_j$, the triangle inequality gives

$$\left|K^{-1}\sum_{t=1}^{K} Z_t\right| \le K^{-1}\sum_{t=1}^{K}|Z_t|.$$

Thus the implemented first-order score suppresses cancellation and measures typical absolute gate sensitivity. This inequality does not upper-bound the finite deletion loss. When v_j changes during training, the observations do not even describe derivatives at one fixed port. The curvature conditions of theorem D.1 are still required for a loss-change certificate.

At an exact stationary point of the unregularized task loss, $u_j = 0$ for every freely trained port. This nulls first-order saliency without implying that deletion is harmless. Stationarity of a regularized

objective need not null the task gradient alone. If $\mathbb{E}\,\|g\|^2 \le K^2$ in the simple-port setting, Cauchy–Schwarz gives $|\mathbb{E}[a_j g]^\top v_j| \le K\sqrt{\mathbb{E}[a_j^2]}\,\|v_j\|$. Writing $e_j = \mathbb{E}[a_j^2]\,\|v_j\|^2$, the removal bound becomes

$$|\Delta\mathcal{L}_j| \le K\sqrt{e_j} + \tfrac{1}{2}H_{\max}e_j. \tag{D.5}$$

Thus shrink-then-remove has vanishing loss cost if $e_j \to 0$ while these bounds remain uniform. Exact functional preservation requires a null contribution on every relevant input together with an admissible deletion. A small population loss bound by itself does not imply pointwise equality. Nullity is a sufficient condition, not a necessary one, since downstream insensitivity or cancellation can also preserve the output.

E A Descent Guarantee for the Calibrated Variant

An exchange combines two edits of one network. Separate local predictions can be added only after their dependence has been addressed. We give a sequential formulation that avoids an unmodeled interaction, followed by a bound when both predictions are made at the original state.

Theorem E.1 (Sequential exchange)**.** Let θ^- be the model after an admissible removal and θ^+ the model after adding a candidate to θ^-. Suppose

$$\mathcal{L}(\theta^-) - \mathcal{L}(\theta) \le R, \qquad \mathcal{L}(\theta^+) - \mathcal{L}(\theta^-) \le -G.$$

If $\widehat{G} - G \le e_G$, $R - \widehat{R} \le e_R$, and

$$\widehat{G} - \widehat{R} > \delta + e_G + e_R, \qquad C(\boldsymbol{n}^+) \le B,$$

then the exchange is budget-feasible and $\mathcal{L}(\theta^+) - \mathcal{L}(\theta) < -\delta$.

Proof. Insert the intermediate state to obtain the exact identity

$$\mathcal{L}(\theta^+) - \mathcal{L}(\theta) = \big[\mathcal{L}(\theta^+) - \mathcal{L}(\theta^-)\big] + \big[\mathcal{L}(\theta^-) - \mathcal{L}(\theta)\big].$$

The assumed bounds give at most $-G + R$. The one-sided estimation inequalities give $-G + R \le -\widehat{G} + \widehat{R} + e_G + e_R < -\delta$. The budget condition is checked on the final width vector and establishes feasibility independently of the loss comparison. The gain bound can be obtained from theorem C.2 on the post-removal model, while theorem D.1 supplies the removal bound on the original model. □

Re-evaluation can be performed on a trial copy before either edit is committed. If growth is scored on the original network instead, let $\Phi(p, q)$ represent the joint loss in a common admissible parameterization, with removal displacement p and growth displacement q. Suppose $\big\|D^2_{pq}\Phi\big\| \le M_{pq}$ over the rectangle (sp, tq) for $s, t \in [0, 1]$. Applying the fundamental theorem of calculus once in each variable gives

$$I(p, q) = \Phi(p, q) - \Phi(p, 0) - \Phi(0, q) + \Phi(0, 0) \tag{E.1}$$

$$= \int_0^1 \int_0^1 p^\top D^2_{pq}\Phi(sp, tq)q\, dt\, ds, \tag{E.2}$$

so $|I(p, q)| \le M_{pq}\,\|p\|\,\|q\|$. If removal is bounded by R and the isolated growth gain is at least G, then the combined loss increase is at most $R - G + M_{pq}\,\|p\|\,\|q\|$. This last term must enter the error allowance in equation (7). In a quadratic approximation it is the mixed Hessian contribution, which is of the same order as the separate quadratic terms.

When a Hessian-Lipschitz local model is used instead of a segment upper bound, equation (C.5) supplies another error term. The allowance must also cover any mismatch between the proposed and actual birth step. For a fixed step ε, the model gain is $\varepsilon\,\|u\| - \varepsilon^2 h/2$. Substituting the optimal model gain while materializing with a different fixed step would not satisfy the theorem's assumptions.

The conversion from density to total value is elementary but necessary. If $G = \kappa_{\ell'}\gamma_{\ell'}$ and $R = \kappa_\ell\rho_\ell$, the comparison is $\kappa_{\ell'}\gamma_{\ell'} > \kappa_\ell\rho_\ell + \delta + \eta$. An unweighted comparison of γ and ρ tests a different condition unless the transferred resource amounts are equal. Finally, the exact budget difference may not equal a difference of two costs measured before either edit. Section H derives that distinction directly from tensor dimensions.

F THE EQUIMARGINAL CONDITION AND ITS DISCRETE BOUNDARY

For local channel cost κ_ℓ, define $\gamma_\ell = \max_c G_c/\kappa_\ell$ and $\rho_\ell = \min_j R_j/\kappa_\ell$. In the calibrated fractional model at a tight budget, no improving transfer exists exactly when $\max_\ell \gamma_\ell \le \lambda \le \min_\ell \rho_\ell$ for some $\lambda \ge 0$. The same algebra can organize frozen surrogate scores, but then λ prices the surrogate rather than task loss. Neither interpretation characterizes indivisible channel exchanges or global network optimality; the discrete boundary is given at the end of this section.

We formulate the relaxation explicitly. At a fixed model, let $G_\ell \ge 0$ be an available growth value and $R_\ell \ge 0$ an available removal cost. Let $\kappa_\ell > 0$ be the local cost of a unit at that site. The algebra below also applies if G_ℓ and R_ℓ are frozen nonnegative surrogate scores. In that case it characterizes stability of the surrogate model, not of the task loss. A loss interpretation requires G_ℓ to lower-bound gain and R_ℓ to upper-bound damage under the same edit geometry. The relaxation treats fractional growth $x_\ell \ge 0$ and removal $y_\ell \ge 0$ as independent infinitesimal actions, with model

$$\Delta_{\text{lin}}(x, y) = \sum_\ell R_\ell y_\ell - \sum_\ell G_\ell x_\ell, \qquad \sum_\ell \kappa_\ell (x_\ell - y_\ell) \le 0. \tag{F.1}$$

Sites at a width floor are excluded from the removal action set. The statement below assumes both action sets are nonempty, and maxima and minima range only over available actions. A transfer can be scaled arbitrarily small, so local upper bounds on action magnitudes do not affect the argument. Values and costs are frozen for this comparison.

Theorem F.1 (Equimarginality of the local relaxation)**.** At a tight budget, $\Delta_{\text{lin}} \ge 0$ for every feasible direction in equation (F.1) if and only if there is a $\lambda \ge 0$ satisfying

$$\max_\ell \frac{G_\ell}{\kappa_\ell} \le \lambda \le \min_\ell \frac{R_\ell}{\kappa_\ell}.$$

At a slack budget, local stability additionally requires all available G_ℓ to vanish.

Proof. Suppose such a λ exists. For any feasible x, y,

$$\Delta_{\text{lin}} = \sum_\ell (R_\ell - \lambda\kappa_\ell) y_\ell + \sum_\ell (\lambda\kappa_\ell - G_\ell) x_\ell + \lambda \sum_\ell \kappa_\ell (y_\ell - x_\ell).$$

The first two sums are nonnegative by the assumed inequalities. The last is nonnegative by feasibility and $\lambda \ge 0$. Hence no feasible direction improves the model.

Conversely, if the interval is empty, choose a growth site b and removal site a with $G_b/\kappa_b > R_a/\kappa_a$. For any sufficiently small $q > 0$, set $x_b = q/\kappa_b$ and $y_a = q/\kappa_a$, with all other actions zero. This releases and spends exactly q units of resource. Its modeled loss change is

$$q\left(\frac{R_a}{\kappa_a} - \frac{G_b}{\kappa_b}\right) < 0,$$

contradicting stability. This proves necessity at a tight budget.

With positive slack, sufficiently small pure growth at any site is feasible. If $G_b > 0$, choosing $x_b > 0$ and $y = 0$ gives negative modeled loss change. Thus every growth value must vanish. Conversely, if all growth values vanish, the model change is $\sum R_\ell y_\ell \ge 0$, so no action improves it. □

The shadow-price interpretation is an inequality because a site's proposed growth feature and its cheapest incumbent need not have the same value. Equality of marginal values requires further regularity. For example, if a differentiable relaxed loss $\overline{\mathcal{L}}(\boldsymbol{n})$ has an interior constrained stationary point and differentiable cost with a nonzero constraint gradient, its stationarity equations are $\partial_{n_\ell}\overline{\mathcal{L}} + \lambda\partial_{n_\ell} C = 0$. These imply equal signed marginal loss reduction per resource at interior sites. The candidate gains and conservative removal bounds above are not automatically derivatives of such a globally defined relaxed loss.

The continuous condition is not equivalent to stability of indivisible edits. Consider an available incumbent of cost $\kappa_\ell = 2$ with removal cost $R_\ell = 2$, and an available candidate of cost $\kappa_{\ell'} = 1$ with gain $G_{\ell'} = 1.5$. At a tight budget the exchange is feasible, yet

$$R_\ell - G_{\ell'} = 0.5 > 0 \qquad \text{while} \qquad \frac{G_{\ell'}}{\kappa_{\ell'}} = 1.5 \;>\; 1 = \frac{R_\ell}{\kappa_\ell}. \tag{F.2}$$

Pure growth is infeasible and pure removal is harmful. The action catalog is therefore exchange-stable, even though the growth density 1.5 exceeds the removal density 1. Fractional removal could fund the growth at lower modeled damage, but a whole-channel controller cannot perform that action. This counterexample rules out the unqualified discrete equivalence without additional structure on admissible bundles. It also shows why no universal one-channel correction follows merely from a ratio of costs.

If G is a lower bound on gain and R an upper bound on damage, failure to find an improving surrogate action establishes only stability of those bounds. It does not exclude a better candidate outside the bank or an edit whose true removal cost is below its conservative estimate. These qualifications are intrinsic to using a finite proposal set and local models.

G Directed Birth and Random Port Directions

At fixed $u_c \in \mathbb{R}^d$, a uniform random Euclidean unit direction has expected absolute projection $\|u_c\|_2 \sqrt{2/(\pi d)}(1 + O(d^{-1}))$, so a joint-normalized gradient direction has a $\Theta(\sqrt{d})$ directional advantage. For k random directions, the expected best absolute projection is at most $\|u\|_2 \sqrt{2\log(2k)/d}$, capped by $\|u\|_2$, and reaching a fixed fraction of the optimum with fixed positive probability requires exponentially many directions in d. This comparison concerns the same fixed candidate feature and the Euclidean reference geometry. It does not compare random incoming features with learned ones or describe the block-normalized multi-consumer step. The two results below give the exact spherical calculation and the tail bound.

Theorem G.1 (Exact random projection)**.** Fix $u \in \mathbb{R}^d \setminus \{0\}$ and draw v uniformly from the unit sphere. For $d \geq 2$,

$$\mathbb{E}|u^\top v| = \|u\| \, \frac{\Gamma(d/2)}{\sqrt{\pi}\Gamma((d+1)/2)}, \qquad \mathbb{E}[u^\top v] = 0.$$

The ratio of the optimal descent rate $\|u\|$ to the expected absolute projection is $\sqrt{\pi d/2}\,(1 + O(d^{-1}))$.

Proof. Rotational invariance permits $u = \|u\| \, e_1$ without changing the distribution. The first coordinate V_1 of a uniform spherical vector has density (Vershynin, 2019)

$$f_d(z) = \frac{\Gamma(d/2)}{\sqrt{\pi}\Gamma((d-1)/2)}(1 - z^2)^{(d-3)/2}, \qquad -1 < z < 1.$$

Its symmetry gives $\mathbb{E}V_1 = 0$. For the absolute moment, substitute $r = 1 - z^2$ to obtain

$$\begin{aligned}\mathbb{E}|V_1| &= \frac{2\Gamma(d/2)}{\sqrt{\pi}\Gamma((d-1)/2)} \int_0^1 z(1 - z^2)^{(d-3)/2} \, dz \\ &= \frac{2\Gamma(d/2)}{(d-1)\sqrt{\pi}\Gamma((d-1)/2)} = \frac{\Gamma(d/2)}{\sqrt{\pi}\Gamma((d+1)/2)}.\end{aligned}$$

Multiplication by $\|u\|$ proves the exact identity. The gamma-ratio asymptotic $\Gamma(z + 1/2)/\Gamma(z) = z^{1/2}(1 + O(z^{-1}))$ (Wendel, 1948), with $z = d/2$, gives $\mathbb{E}|u^\top v| = \|u\| \sqrt{2/(\pi d)}(1 + O(d^{-1}))$. Inverting its factor yields the stated ratio. For $d = 1$, the sphere consists of $\{-1, 1\}$ and the absolute projection equals $\|u\|$ directly. □

The absolute projection corresponds to allowing a sign correction after measuring a random direction. Specifically, replacing v by $-\operatorname{sign}(u^\top v)v$ gives nonnegative descent rate $|u^\top v|$. Without that measurement and correction, the signed first-order gain averages to zero. Neither statement compares two different incoming features, since both hold with the same fixed u.

Theorem G.2 (A finite-dimensional bound for a random bank)**.** For $d \geq 2$, k independent uniform directions v_i, and any $a > 0$,

$$\mathbb{E} \max_{i \leq k} |u^\top v_i| \leq \|u\| \min\left\{1, \sqrt{\frac{2\log(2k)}{d}}\right\}, \tag{G.1}$$

$$\mathbb{P}\left(\max_{i \leq k} |u^\top v_i| \geq a\,\|u\|\right) \leq 2k \exp(-da^2/2). \tag{G.2}$$

For a fixed $a \in (0, 1)$, attaining this fraction with probability at least $p_0 > 0$ requires $k \geq (p_0/2)e^{da^2/2}$.

Proof. By rotational invariance, it suffices to bound $Z = V_1$. Its even moments follow by the substitution $r = z^2$ in the density integral,

$$\mathbb{E}Z^{2m} = \frac{\Gamma(m+1/2)\Gamma(d/2)}{\sqrt{\pi}\Gamma(m+d/2)} = \frac{(2m-1)!!}{d(d+2)\cdots(d+2m-2)} \le \frac{(2m-1)!!}{d^m}.$$

Odd moments vanish. Since $|Z| \le 1$, its exponential series can be integrated term by term. Therefore

$$\mathbb{E}e^{tZ} \le \sum_{m\ge 0} \frac{t^{2m}(2m-1)!!}{(2m)!d^m} = \sum_{m\ge 0} \frac{(t^2/(2d))^m}{m!} = e^{t^2/(2d)}.$$

Chernoff's inequality gives $\mathbb{P}(Z \ge a) \le \inf_{t>0} e^{-ta+t^2/(2d)} = e^{-da^2/2}$. Symmetry and a union bound over k directions prove equation (G.2). This argument in fact does not require independence for the upper bound.

For the expectation, let $M_k = \max_i |Z_i|$. For $t > 0$,

$$e^{tM_k} \le \sum_{i=1}^{k} (e^{tZ_i} + e^{-tZ_i}), \qquad e^{t\mathbb{E}M_k} \le \mathbb{E}e^{tM_k} \le 2ke^{t^2/(2d)}.$$

Take logarithms and minimize $\log(2k)/t + t/(2d)$ at $t = \sqrt{2d\log(2k)}$. The resulting bound, together with $M_k \le 1$, proves equation (G.1). Rearranging equation (G.2) under the requirement that its left side be at least p_0 gives the necessary bank size. Exact alignment with a fixed direction has probability zero in finite independent banks when $d \ge 2$, so the operational statement concerns a fixed fraction of the optimal projection. □

H Coupling, Normalization, and State-Preserving Edits

A coupling group G carries an abstract channel index set I_G. Every occurrence of that index in a tensor is represented by a block map

$$\rho_b : I_G \times \{1, \dots, m_b\} \longrightarrow \text{tensor-axis positions.} \tag{H.1}$$

The multiplicity m_b records repeated coordinates such as the flattened spatial block belonging to one convolutional channel. Concatenations add block offsets without merging the identities of their different inputs. A residual addition instead identifies the relevant index sets. Divisibility constraints restrict which index edits are admissible, and grouped operators may require balanced changes in distinct blocks rather than merely a total-size constraint.

Proposition H.1 (Consistency of a group edit)**.** Suppose every dependency imposed by the represented operators is encoded in its index maps, and an edit respects their block and divisibility constraints. Applying the same restriction or extension of I_G through all its maps preserves those shape constraints. If the added or removed coordinates have no effect on retained computations and Condition (N) holds, the edit also preserves the represented function.

Proof. For deletion, choose a subset $J_G \subset I_G$ of retained indices and restrict every occurrence to $\rho_b(J_G \times \{1, \dots, m_b\})$. Two tensor axes identified by an operator then retain identical abstract indices, even when their physical layouts differ. Residual operands consequently remain aligned. At a concatenation, each retained block receives the offset determined by the lengths of the preceding retained blocks. Flattened channels retain all associated positions. Divisibility and within-group conditions hold by the assumed admissibility of the edit. The same reasoning in reverse applies to extension with new indices. This proves shape consistency for the encoded operator rules.

To prove functional preservation, apply induction through the graph as in lemma A.1. Retained parameters and state agree by their index maps. By assumption, the removed contributions are null or the added contributions are blocked. No altered statistic enters a retained coordinate. Each retained operator output therefore agrees with its previous value, yielding equality at the final output. The dependency-coverage assumption is essential. A successful traced example cannot establish it for an unrepresented data-dependent branch. □

For a depthwise convolution (Howard et al., 2017) with multiplier one, input channel i is paired with filter i. A deletion removes that pair, updates both channel counts and the group count, and

restricts downstream consumers consistently. With multiplier greater than one, the associated filter block must be edited as a whole. A general grouped convolution with a fixed number of groups requires compatible changes within its groups. An arbitrary single-channel removal does not satisfy this requirement merely because another tensor can be sliced.

Squeeze-and-excitation (Hu et al., 2018) introduces an additional path. Write a block schematically as $y_i = x_i\sigma(b_i + \sum_r W^{(2)}_{ir}\phi(b_r + \sum_j W^{(1)}_{rj}\overline{x}_j))$. Appending a channel c to x changes retained gates if $W^{(1)}_{r,c}$ is nonzero. Zeroing those new consumer coefficients preserves the inputs to retained gate computations. Ordinary downstream consumers of y_c must also be zeroed. The candidate channel's own gate may be initialized without affecting retained outputs, provided these paths stay blocked. Spatial averaging within a channel does not mix it with other channels. This calculation explains why an excitation branch must be represented among the consumers rather than treated as an inert channelwise transformation.

H.1 Why channel-spanning normalization is different

Let n retained channel values have mean μ and variance $\sigma^2 = n^{-1}\sum_i (h_i - \mu)^2$. Append a value a and put $q = a - \mu$. Direct expansion of the new mean and variance gives

$$\mu' = \mu + \frac{q}{n+1}, \qquad \sigma'^2 = \frac{n}{n+1}\sigma^2 + \frac{n}{(n+1)^2}q^2. \tag{H.2}$$

To see the second identity, expand retained deviations as $(h_i - \mu) - q/(n+1)$, sum their squares, and use $\sum_i (h_i - \mu) = 0$. The new channel's deviation is $nq/(n+1)$. Adding these squared deviations and dividing by $n+1$ gives the formula.

Consequently, a zero outgoing port does not prevent a new channel from changing every retained normalized value. If the stabilizing constant is $\epsilon_N > 0$, the affine-normalized output is $z_i = \gamma_i(h_i - \mu)/\sqrt{\sigma^2 + \epsilon_N} + \beta_i$. For uniformly bounded q, retained centered values and affine scales, and variance denominators uniformly bounded away from zero, Taylor expansion of the inverse square root gives

$$z'_i - z_i = \gamma_i \left[-\frac{q}{(n+1)\sqrt{s}} - \frac{(h_i - \mu)(\sigma'^2 - \sigma^2)}{2s^{3/2}} \right] + O(n^{-2}), \qquad s = \sigma^2 + \epsilon_N. \tag{H.3}$$

This is $O(n^{-1})$ under those bounds. There is no width-only numerical guarantee when an appended activation is unbounded or the denominator is arbitrarily small.

A masked normalizer that computes statistics on the retained set alone restores silence while a candidate remains outside that set. Promotion enlarges the set and changes the statistics according to equation (H.2). Thus masked sensing does not by itself make promotion exactly function-preserving. Per-channel normalization avoids this particular coupling because the statistic for a retained channel has no dependence on the added one. These observations justify freezing unsupported axes rather than claiming universal normalization compatibility.

H.2 Exact parameter accounting

The reported count uses tensor extents and excludes candidates. Proposal costs use pre-edit bundle marginals, which can include dormant extents and omit cross terms from equation (2). Recounting after commitment does not make the proposal exact. Candidate computation remains discovery cost, and parameter savings need not imply proportional speedups. The exact accounting is as follows.

For a tensor T, let $d_{T,a}(\boldsymbol{n})$ be the deployed extent of its axis a at the width vector $\boldsymbol{n}$. Counting each shared parameter tensor once, the deployed parameter count is

$$C(\boldsymbol{n}) = \sum_T \prod_a d_{T,a}(\boldsymbol{n}). \tag{H.4}$$

The exact cost of a proposed growth is $\Delta C^+_G = C(\boldsymbol{n} + q_G e_G) - C(\boldsymbol{n})$, where e_G selects group G's entry of the width vector and q_G is the number of indices added, with an analogous finite difference for removal. If a group changes only one axis of a tensor linearly, its one-index increment is that axis's multiplicity times the product of the other extents. If it changes two axes of the same tensor, cross terms must be included. A square tensor makes the gap explicit. Write its count as $C(a, b) = ab$ and evaluate on the diagonal $a = b = n$. Adding one index to each axis costs

$$C(n+1, n+1) - C(n, n) = 2n + 1, \qquad \text{whereas} \qquad \frac{\partial C}{\partial a} + \frac{\partial C}{\partial b}\bigg|_{a=b=n} = b + a = 2n, \tag{H.5}$$

so the two first-order marginals miss the corner entry that both edits touch.

Even distinct neighboring groups interact. For a matrix of count ab whose two axes are edited at once,

$$(a+1)(b-1) - ab = b - a - 1, \qquad \text{whereas} \qquad \kappa_a - \kappa_b = b - a, \tag{H.6}$$

so the pre-edit marginals overstate the saving by exactly one entry. Exact post-edit counting avoids this ambiguity. It also accommodates changes to grouped-convolution storage without assuming a fixed formula for every operator.

Candidate stripping requires the same care. If both axes of a matrix carry k candidates, subtracting each axis's candidate block from the stored count leaves

$$(a+k)(b+k) - k(a+k) - k(b+k) = ab - k^2, \tag{H.7}$$

which falls short of the deployed count ab by the candidate-by-candidate corner k^2. Reading the real extent on every axis gives the correct result directly. Buffers are maintained by edits but do not belong in trainable parameter count.

H.3 Optimizer state and dormancy

Let P be the restriction map that keeps selected entries of a parameter tensor. For momentum state m, the update $m^+ = \mu m + g$ obeys $Pm^+ = \mu Pm + Pg$. Thus applying P to both the parameter and its momentum retains the same accumulated state on every surviving coordinate. Adam's first and second moment updates are coordinatewise, so their arrays can be restricted in the same way. Replacement tensor objects must be registered with the optimizer under their new identities.

For appended entries, state can be initialized at zero while surviving entries retain their history. This preserves stored moments, but a tensor-wide Adam step counter gives new entries the existing bias-correction age. Exactly reproducing an independently fresh Adam parameter would require per-coordinate age handling. Future gradients also change after a functional edit. State preservation therefore does not imply that the subsequent optimization trajectory remains unchanged.

Finally, let Π set all dormant port entries to zero. The update $\theta_{t+1} = \Pi(\mathrm{OptStep}(\theta_t))$ restores silence before the next forward pass regardless of the port's nonzero gradient. A forward pass must not occur between the optimizer step and this projection. Projection of parameters alone does not clear optimizer moments on those entries. If a dormant entry is promoted, its state needs an explicit policy so accumulated port momentum does not masquerade as a fresh birth. No exact optimizer-state initialization for newly active entries is implied merely by re-zeroing the parameter values.

I Windowed Estimation and Direction Error

At a fixed state, independent minibatch gradients Z_t with $q = \mathbb{E}\,\|Z_t\|_2$ and $\mathrm{Var}(\|Z_t\|_2) \le \tau^2$ give

$$\mathbb{E}\bar{q} = q, \qquad \mathbb{E}(\bar{q} - q)^2 \le \tau^2/K, \qquad q \ge \|\mathbb{E}Z_t\|_2\,, \tag{I.1}$$

so the released window estimates gradient activity, not a population gradient vector or a finite loss decrease, and drift and candidate selection add staleness and selection effects. By contrast, the vector mean $\widehat{u} = K^{-1}\sum_t Z_t$ obeys $\mathbb{E}\,\|\widehat{u} - u\|_2^2 \le \sigma^2/K$ when the gradient-noise second moment is at most σ^2, and if $\|\widehat{u} - u\|_2 \le e$ its normalized direction has true first-order rate at least $\|u\|_2 - 2e$. The results below give the fixed-state analysis and its adaptive qualifications.

Proposition I.1 (Estimation target of the implemented norm window)**.** At a fixed state, let $Z_1, \dots, Z_K$ be independent minibatch port gradients with mean u. Put $Q_t = \|Z_t\|_2$, $q = \mathbb{E}Q_t$, and suppose $\mathrm{Var}(Q_t) \le \tau^2$. Then the implemented per-candidate statistic $\widehat{q} = K^{-1}\sum_t Q_t$ satisfies

$$\mathbb{E}\widehat{q} = q, \qquad \mathbb{E}(\widehat{q} - q)^2 \le \frac{\tau^2}{K}, \qquad q \ge \|u\|_2\,.$$

Consequently, $\widehat{q}$ estimates expected minibatch gradient magnitude, not the magnitude of the population gradient.

Proof. Linearity gives unbiasedness. Independence makes the variance of the average at most τ^2/K. Jensen's inequality for the Euclidean norm gives $\mathbb{E}\,\|Z\|_t \ge \|\mathbb{E}Z_t\| = \|u\|$. □

For a finite bank, selecting the largest $\widehat{q}_c$ introduces the usual winner's bias. A simultaneous Chebyshev bound follows by applying $\mathbb{P}(|\widehat{q}_c - q_c| \geq e) \leq \tau_c^2/(Ke^2)$ to each candidate and summing the failure probabilities. Along training, both the conditional distribution of Z_t and the network state move, so the window is more naturally interpreted as a temporal average of gradient activity. This is precisely the quantity used for ranking by the released controller.

Proposition I.2 (Mean-gradient estimation at a fixed state)**.** Let $Z_1, \ldots, Z_K \in \mathbb{R}^d$ be independent port-gradient observations from a fixed network and candidate, with mean u and $\mathbb{E}\left\|Z_i - u\right\|^2 \leq \sigma^2$. For $\widehat{u} = K^{-1}\sum_i Z_i$,

$$\mathbb{E}\left\|\widehat{u} - u\right\|^2 \leq \frac{\sigma^2}{K}, \qquad \mathbb{P}(\|\widehat{u} - u\| \geq e) \leq \frac{\sigma^2}{Ke^2}.$$

If only a coordinate variance bound σ_0^2 is known, one may take $\sigma^2 = d\sigma_0^2$.

Proof. Write $X_i = Z_i - u$. On expanding the squared norm of $K^{-1}\sum_i X_i$, cross inner products have zero expectation because observations are independent and centered. The remaining K diagonal terms sum to at most $K\sigma^2$, yielding the mean-square bound after division by K^2. Markov's inequality applied to the nonnegative variable $\left\|\widehat{u} - u\right\|^2$ gives the tail bound. Summing coordinate variances gives $\mathbb{E}\left\|Z_i - u\right\|^2 \leq d\sigma_0^2$ without requiring coordinate independence. □

For a relative error $e = \alpha\left\|u\right\|$ with failure probability at most p, the sufficient window size is $K \geq \sigma^2/(p\alpha^2\left\|u\right\|^2)$. This controls the vector error directly. A coordinatewise union bound must account for the conversion from coordinate errors to the vector norm and is not needed here.

Along a training trajectory, let u_t be the conditional mean gradient at the state used for observation Z_t. If the noise terms are martingale differences with conditional second moment at most σ^2, their cross expectations still vanish. Relative to the final state u_K,

$$\widehat{u} - u_K = \frac{1}{K}\sum_{t=1}^{K}(Z_t - u_t) + \frac{1}{K}\sum_{t=1}^{K}(u_t - u_K).$$

The first term has the same mean-square bound. If u is L_u-Lipschitz in the full sensing state z_t, the norm of the second term is at most $L_u K^{-1}\sum_t \left\|z_t - z_K\right\|$. The state includes candidate parameters and normalization state when these change. This separates sampling noise from staleness without assuming a learning-rate-only bound on parameter motion.

The released controller computes

$$\max_c \frac{1}{K}\sum_t \left\|Z_{t,c}\right\|,$$

with the maximum taken after each candidate's temporal average. By proposition I.1, the individual averages target $\mathbb{E}\left\|Z\right\|_c$ at a fixed state. Selecting their maximum introduces an additional selection effect. This statistic differs both from the temporal average of per-step maxima and from $\max_c \left\|\mathbb{E}Z_{t,c}\right\|$. A scalar norm average also cannot specify a port direction, which is why the edit routine uses the selected candidate's most recent gradient blocks.

Lemma I.3 (Descent using an estimated direction)**.** Suppose $\|\widehat{u} - u\| \leq e$ and $\widehat{u} \neq 0$. For $\widehat{d} = -\widehat{u}/\left\|\widehat{u}\right\|$,

$$-u^\top \widehat{d} \geq \left\|u\right\| - 2e.$$

Under the directional upper curvature bound β, a step t therefore has loss change at most $-t(\left\|u\right\| - 2e) + \beta t^2/2$.

Proof. Put $r = \widehat{u} - u$. Then

$$-u^\top \widehat{d} = \frac{u^\top \widehat{u}}{\left\|\widehat{u}\right\|} = \left\|\widehat{u}\right\| - \frac{r^\top \widehat{u}}{\left\|\widehat{u}\right\|} \geq \left\|\widehat{u}\right\| - e \geq \left\|u\right\| - 2e.$$

The last inequality is the reverse triangle inequality. Substitute this first-derivative bound into the integral upper bound used in theorem C.2 to obtain the finite-step statement. In particular, a sufficiently small positive step is descending when $e < \left\|u\right\|/2$. □

For a finite collection of decisions, per-decision failure probabilities must sum to the desired run-level failure probability. With an adaptively chosen or unbounded number of decisions, one may allocate $p_m = 6p/(\pi^2 m^2)$ to decision m, since $\sum_m p_m = p$. Each bound must hold conditionally on the preceding history, and it must cover selection among candidates, for example by a simultaneous bound over the bank. A single fixed-candidate confidence statement is not automatically valid after selecting its maximum.

J Certified Event Counts and Practical Budget Enforcement

The global conclusions available for the calibrated variant are conditional. If certified structural events each decrease a nonnegative loss by at least $\delta/2$ and all other transitions have total positive variation V_+, their number is bounded by equation (J.1) below. Forced feasibility removals and tolerance-based practical edits are not certified events, so the bound applies to the calibrated trajectory and not to the number of operations emitted by the released controller. The theorem is followed by the separate question of budget attainability.

Theorem J.1 (Event bound with intervening loss increases)**.** Consider a finite trajectory of nonnegative losses. Suppose N certified structural events each decrease the loss by at least $\delta/2$, where $\delta > 0$. Let V_+ be the sum of positive loss changes from all other transitions, including stochastic weight steps and feasibility prunes. Then

$$N \leq \frac{2(\mathcal{L}_0 + V_+)}{\delta}. \tag{J.1}$$

In particular, $N \leq 2\mathcal{L}_0/\delta$ if all other transitions are non-increasing.

Proof. Partition the increments into certified events and all other transitions. The first group sums to at most $-N\delta/2$. The second group sums to at most its positive variation V_+, since discarding its negative increments can only increase that sum. Telescoping therefore gives $\mathcal{L}_{\text{final}} \leq \mathcal{L}_0 - N\delta/2 + V_+$. Nonnegativity of the final loss implies $0 \leq \mathcal{L}_0 - N\delta/2 + V_+$, which rearranges to equation (J.1). □

The bound belongs to the calibrated variant and is meaningful for a descent-controlled trajectory. It is not an unconditional count of operations in the released annealed controller. A forced prune can pay a positive loss cost. Pure removal accepted below a tolerance need not decrease loss, and a zero-port embedding changes no loss at all. Such events cannot be counted as decreases of $\delta/2$. A newborn-protection rule could prevent immediate re-pruning for a chosen duration, but it is neither present in the audited branches nor sufficient by itself to rule out later cycles. Once structural adaptation is disabled, subsequent training is ordinary optimization on the fixed final shape, with whatever guarantees that optimizer separately satisfies.

Budget enforcement has a different invariant. If each committed loss-informed proposal is checked using its exact post-edit count, a model already within a fixed budget remains within it after those proposals. A shrinking schedule changes the feasible set between updates. A model can therefore exceed the newly scheduled target without having accepted an infeasible proposal. The feasibility phase repairs that discrepancy only if enough legal removals are available.

A basic necessary capacity check compares the target with the minimum deployed count reachable under the group floors and constraints. When all modeled cost dimensions are monotone in widths and their minima are jointly admissible, evaluating those minimum widths gives that floor count. More complicated coupled constraints require an admissible minimum construction. If B^* is below this count, the target is impossible under the available edit family. Even above the floor, per-update caps and a finite adaptation window can delay its attainment. If at most m channels may be removed at each of K remaining updates, any plan requiring more than mK such removals is impossible within that window. This is a schedule constraint, not a property of the loss signal.

These two forms of reasoning serve different purposes. The descent bound describes accepted improvements under controlled errors. Exact parameter accounting describes resource feasibility. Neither substitutes for an empirical assessment of the accuracy retained by the final architecture.

K Admissibility Conditions

A coupling group describes which axes must change together. Admissibility asks whether the resulting operation preserves the task interface and the conditions under which candidates can remain silent. These questions are distinct. A collection of tensors can have compatible shapes while an

added channel changes a normalization statistic or an input dimension is no longer appropriate for the data. The following checks define the required screening, with branch-specific implementation coverage stated below.

K.1 Group discovery and eligibility

The model is traced with PyTorch's symbolic tracer and its tensor shapes come from an example input. A union–find construction identifies channel axes constrained to have equal size by the represented operators. Each group stores slots identifying a module parameter or buffer, its axis, the role of that axis, and any multiplicity needed to map one channel to several physical positions. Producer output axes, consumer input axes, and normalization coordinates are thereby associated with the same abstract index set.

A potentially plastic group must contain both a producer and a consumer. A group with no producer corresponds to an external input. A group with no consumer generally represents an output whose width belongs to the task, such as a classifier head. Neither is an available capacity dimension. This eligibility check is followed by normalization and reshape checks rather than being treated as sufficient evidence that a group is safe to edit.

At residual additions, operand channels share indices. Concatenation instead preserves the distinct incoming blocks and requires their offsets to be updated when a block changes size. Flattening can map a channel to several consumer coordinates. A convolutional channel flattened over S positions, for example, maps to the block $\{iS, \dots, (i+1)S - 1\}$. These mappings explain why a group needs more information than its current width. Their consistency, when all maps are represented, is established in proposition H.1.

The calculus is broader than the released tracer. The audited index-bundle builder handles ordinary convolutional and linear producer–consumer axes, residual identifications, affine normalization coordinates, and the implemented flatten multiplicity. It records concatenation metadata but does not apply those offsets when building edit bundles. It also lacks a complete rule for general grouped convolutions, recurrent state axes, and attention's activation-to-activation constraints. Such groups are not covered merely because tracing succeeds. They require an explicit index-map rule or exclusion by an admissibility check. Consequently, proposition H.1 is a conditional result about a complete bundle, not a claim that the present tracer recognizes every operator in the theoretical catalog.

K.2 Condition (N) and normalization span

Condition (N) excludes normalizers that compute retained-channel statistics over the candidate-bearing axis. BatchNorm and InstanceNorm (Ioffe & Szegedy, 2015; Ulyanov et al., 2016) compute their statistics separately for each channel, so appending a channel does not change the statistic assigned to a retained channel. In contrast, LayerNorm (Ba et al., 2016) across the edited feature axis reads both retained and candidate values when forming its mean and variance. Ordinary GroupNorm (Wu & He, 2018) has the same obstruction within a group whose membership changes.

Setting the candidate's outgoing weights to zero cannot block this statistical path. The new value has already altered retained normalized activations before reaching an outgoing consumer. Equation (H.2) gives the exact mean and variance changes, followed by the conditions for an $O(1/n)$ perturbation. That asymptotic statement is not exact function preservation. A masked normalizer can keep candidates outside the statistics during sensing, but promotion changes the membership and needs its own treatment. A normalizer spanning an unrelated axis is not excluded merely because of its module type. The relevant question is which values determine each retained statistic.

K.3 Condition (R) and hardcoded reshapes

A group must also remain compatible with reshapes that assign a semantic meaning to its width. Multi-head attention (Vaswani et al., 2017) provides a concrete example. A projection may be reshaped by a fixed view into $(B, L, h, d_{\text{head}})$, requiring its width to equal hd_{head}. Removing an arbitrary coordinate can violate this requirement even if neighboring linear layers have been sliced consistently.

For example, a projection of shape $(2, 6, 32)$ can be viewed as $(2, 6, 4, 8)$. Removing one channel leaves a tensor the fixed view cannot accept,

$$2 \cdot 6 \cdot 31 = 372 \;\neq\; 384 = 2 \cdot 6 \cdot 4 \cdot 8. \tag{K.1}$$

The resulting failure is a reshape mismatch, not a failure of the loss signal. Some such constraints can be inferred statically, but a trace that records only ordinary producer-consumer slots need not represent the internal head arithmetic or activation-to-activation matrix products. Quantized head edits require explicit support for the corresponding block layout.

K.4 Task-fixed axes and the empirical probe

Task dimensions can pass ordinary shape and normalization checks while still being ineligible for capacity search. In a forecasting model, the number of variates and the lookback length determine which input arrays the network accepts. They are not interchangeable hidden widths. These examples explain why eligibility cannot be inferred from normalization type alone, even though the current main paper focuses on the general structural mechanism.

The forecasting harness uses a fresh reconstruction of the model for each proposed group, removes one index, and runs a forward pass on a fixed input. A failed probe excludes that group before it receives candidates. The original model is not progressively damaged while testing the other groups. The resulting allowlist restricts every subsequent structural proposal. The audited vision harness does not run this probe. It rejects LayerNorm and GroupNorm groups during bank attachment, but otherwise relies on the tracer's implemented operator rules and its architecture tests. The difference is deliberate: the four vision families use only operators for which the index calculus has an explicit rule (convolution, linear, residual addition, per-channel normalization, depthwise convolution, squeeze-and-excitation, flatten), each covered by an invariant test that resizes every group and checks the forward pass and the exact count, so the probe would repeat what the test suite already establishes; the forecasting backbones introduce attention, whose head arithmetic the tracer cannot see, and the probe is the sound way to exclude those groups. A new architecture family should run the probe until its operators have rules and tests of their own.

This procedure catches the attention mismatch above and task-interface failures exposed by the example. A successful probe establishes compatibility for that attempted edit and input. It does not certify an untraced branch, all future widths, or all possible inputs. Likewise, a failed single-channel probe does not prove that every larger quantized edit is impossible. It establishes that the particular edit family being used is unsupported there. The controller should therefore remain within the validated operator rules and use the probe as an additional operational check.

K.5 Time-series axes and implemented cases

For forecasting input $x \in \mathbb{R}^{B\times L\times C}$ and output in $\mathbb{R}^{B\times T\times C}$, the lookback L, variate count C, and horizon T belong to the task interface. They are not width variables even when a tensor operation would permit them to be sliced. The reduction axis used by a saliency statistic is also architecture-dependent. A convolutional hidden activation has layout (B, D, L) and channel axis 1, whereas linear features have layout (B, L, D) and channel axis 2. The forecasting branch implements this distinction in its hooks. The vision branch's hook assumes axis 1, which is correct for its convolutional models but should not be reused unchanged for sequence-valued linear layers.

In the channel-independent TCN, each variate is reshaped into a separate univariate example. The hidden convolutional width D is plastic, residual additions couple the corresponding block axes, and BatchNorm is per hidden channel. RevIN computes statistics over time separately for each sample and variate before the hidden TCN. It therefore does not span the plastic hidden axis and does not violate Condition (N). In the channel-mixing TCN, the input variate axis and the forecast head's output size remain task-fixed even though intermediate convolutional widths may be edited.

For the audited PatchTST and iTransformer implementations, the residual width d_{model} is frozen because LayerNorm spans it and multi-head reshapes impose additional arithmetic constraints. The feed-forward hidden width d_{ff} is the supported plastic axis. Its producer is the first feed-forward linear map and its zero port is the corresponding input column block of the second map. TSMixer similarly exposes feature-mixing hidden widths, while its variate and time-mixing axes are task-fixed. These are implementation-specific admissibility conclusions, not a claim that every TCN, Transformer, or mixer admits the same edit family. The one-index probe further restricts the actual allowlist used by the time-series harness.

L Implementation and Structural-Update Procedure

This section collects the execution details supporting the short methodology in the main paper. The procedure distinguishes the first-order controller of section 5 from the stronger conditions required by the descent certificate. It does not assume that inexpensive sensing implements candidate optimization or exact curvature estimation.

L.1 Candidate-bank layout and dormancy

In the released controller, the most recent minibatch gradient, rather than the scalar window average, supplies the block directions used at birth. Each optimizer step is immediately followed by projecting candidate ports to zero. Projection does not clear optimizer moments, and candidate gradients can affect global clipping; the complete execution order follows.

For a group with n_ℓ real channels, a bank of k candidates extends the associated tensors with k additional indices. The overhead is that of k extra channels per editable group and nothing else: the bank adds $k \sum_\ell \kappa_\ell$ parameters, where κ_ℓ is the per-channel parameter cost of group ℓ, and the same fraction of forward and backward computation, since a candidate is an ordinary channel whose outgoing weights happen to be zero. For the archived runs at $k = 4$ this is 266,396 parameters on ResNet-18 (2.4% of the dense reference) and 284,056 on VGG-16-BN (1.9%), computed exactly from the replayed group costs of section T; memory grows by the same fraction plus the optimizer moments of those rows. Sensing costs no additional pass, because the port derivative is read from the gradients the training step already produces, and the structural update itself runs once every $K = 100$ steps. Incoming producer rows are Kaiming-initialized. Every independent consumer coordinate through which a candidate can affect retained outputs is set to zero. A candidate port may span several tensors. In a block with squeeze-and-excitation, it must include connections into the gate-generating branch as well as ordinary downstream consumers, as derived in section H.

The parameter gradient at these zero entries is the measured port derivative. It must be read after backpropagation and before an optimizer update replaces the current state. An ordinary update would generally move the entries away from zero. All dormant ports are therefore re-zeroed after every optimizer step and once more after a batch of structural edits. No subsequent forward pass should intervene before this projection. The projected model retains silence even though its stored optimizer state may contain nonzero port moments. If an optimizer applies a coupled transformation to the entire gradient, such as global-norm clipping, unmasked candidate port gradients can rescale updates of the real weights even while the represented function remains unchanged. The function-preserving embedding guarantees equality of raw task gradients on real coordinates, not equality of every optimizer update. This effect should be tested or avoided when comparing discovery training with ordinary training.

Incoming candidate rows receive no task-loss gradient under exact dormancy. Random initialization followed only by task updates therefore does not realize the feature-learning objective of section B. Weight decay or other parameter updates can still move those rows without learning the correlation objective. Candidate optimization, if enabled, must be implemented and accounted for separately. The measured outgoing direction remains useful even with a random incoming proposal, and the directional theorem holds conditionally on that feature.

L.2 Measurements between edits

Let K be the structural-update interval. The released controllers use a default interval of 100 training steps. If $u_{t,c}$ is the minibatch port gradient for candidate c at step t, the code retains the candidate identity and computes

$$\widehat{s}_{\ell,c} = \frac{1}{K} \sum_{t=1}^{K} \|u_{t,c}\|_2 , \qquad c_\ell^* \in \arg\max_{c \in \mathcal{C}_\ell} \widehat{s}_{\ell,c}. \tag{L.1}$$

Thus the implemented statistic is a maximum after per-candidate temporal averaging, not a temporal average of per-step maxima. It is also not $\left\| K^{-1} \sum_t u_{t,c} \right\|_2$, since directions that cancel still contribute positively to equation (L.1). The code stores norms rather than port vectors. Candidate selection uses the window, while materialization reads the selected candidate's gradient remaining from the most recent backward pass. It therefore does not open the port along a window-averaged or population-mean direction. Section I distinguishes the bias of norm and maximum operations from the estimation error of an averaged vector.

For disjoint consumer-coordinate blocks, the group port is their concatenation and its norm satisfies

$$\|u_c\|_2^2 = \sum_{r \in \mathrm{cons}(\ell)} \left\| u_c^{(r)} \right\|_2^2 .$$

Summing or averaging the individual norms would define a different score. If a parameter is shared between consumers, its gradient contributions must instead be accumulated into that independent coordinate before its square is taken.

Incumbent measurements combine the absolute port weight-gradient product with a curvature proxy. The implemented windowed first-order statistic is

$$\widetilde{s}_j^{(1)} = \frac{1}{K} \sum_{t=1}^{K} \left| \sum_{q \in \mathrm{port}(j)} v_{t,j,q} \frac{\partial \widehat{\mathcal{L}}_t}{\partial v_{t,j,q}} \right|, \qquad s_j = \widetilde{s}_j^{(1)} + \frac{\widetilde{H}}{2} \widehat{\mathbb{E}[a_j^2]} \, \|v_j\|_2^2 . \tag{L.2}$$

Here $\widehat{\mathcal{L}}_t$ is the minibatch loss, $\widetilde{H}$ is the configured curvature coefficient rather than a verified bound, and the port can change during the window. Even for a fixed port, averaging absolute minibatch products differs from the absolute product with a mean gradient. The former is at least the latter by Jensen's inequality, but neither is a finite deletion bound without curvature control.

The activation statistic is accumulated over batch and non-channel positions, preserving the actual channel axis. For a convolutional tensor (B, C, L) that axis is 1. For a linear-layer activation (B, L, D) it is the final axis. Reducing over the wrong axis can silently produce a time-position statistic instead of a channel statistic. The mean square in this practical formula is a ranking proxy for tied convolutional ports, not generally the port-Hessian factor in equation (C.1). Even a single convolutional weight is used at several positions. The two-position counterexample in section D shows that this proxy can be smaller than the actual deletion loss at a stationary point despite a correctly bounded downstream Hessian. A certified bound instead needs a verified $\beta_j \geq \sup_{s \in [0,1]} \left\| \nabla^2 F_j(v_j - s v_j) \right\|_{\mathrm{op}}$, for example a bound using the full injection operator $H_{\max} \mathbb{E} \, \|A_x\|_{\mathrm{op}}^2$ where that representation applies. Several coupled consumers require control of their cross terms as well.

The released code uses a default curvature coefficient of 1.0, with a finite-difference calibration option. That option estimates local directional curvature for candidate births. It does not by itself upper-bound every incumbent's port Hessian along the full deletion segment. A chosen or calibrated coefficient is therefore not automatically a valid β_j. It is also coordinate-dependent, so its numerical value is not intrinsically comparable across parameterizations. Scores computed with it can rank channels, but the removal guarantee applies only when the bound's assumptions actually hold. This separates the practical saliency from its theoretical interpretation.

L.3 Budget schedule and the two controller regimes

For an initial deployed count C_0, target B^*, and positive anneal duration t_{end}, the scheduled budget is

$$B(t) = C_0 + \min\{t/t_{\mathrm{end}}, 1\}(B^* - C_0), \qquad t_{\mathrm{end}} = \max\{1, \lfloor \alpha T_{\mathrm{total}} \rfloor\}. \tag{L.3}$$

The vision runs use $\alpha = 0.5$ and the forecasting runs $\alpha = 0.3$ (section N). The schedule and the structural-update interval determine different timescales. The schedule describes the intended resource ceiling, while edits occur only at the structural updates.

At an update with $C(\boldsymbol{n}) > B(t)$, the controller enters the feasibility regime. It selects an eligible low-saliency incumbent in each group and orders the groups by estimated damage per parameter released. It proposes removals until a projected count meets the scheduled budget or the per-update cap is reached. Width floors restrict the available incumbents, and the implementation permits at most one removal from a given group in an update. This avoids shifting indices underneath additional same-group selections. The projection subtracts pre-edit bundle costs. Because those costs are evaluated on stored tensors that include dormant candidates and do not include simultaneous-edit cross terms, the projection is not generally the exact finite difference of deployed counts. The exact deployed count is recomputed after commitment, but there is no automatic repair within the same update if the projection was inaccurate.

Once $C(\boldsymbol{n}) \leq B(t)$, the controller enters the optimality regime. The stated cascade considers pure growth using available slack, then pure removal of a relatively weak incumbent, followed by a budget-feasible exchange. For positive group-median saliency, with the released tolerance $\tau_{\mathrm{dead}} = 10^{-3}$, the relative removal criterion is

$$\frac{s_j}{\mathrm{median}_{i \in \ell} \, s_i} < \tau_{\mathrm{dead}}. \tag{L.4}$$

The released implementation skips this pure-removal test when the median is zero. A relative threshold compares a channel with its group. It neither guarantees a negative loss change nor forces the model to satisfy a smaller budget. Feasibility removals deliberately bypass that quality criterion when the resource constraint requires a reduction.

Before adaptation, the requested target should be checked against the minimum reachable count under the admissible groups and their width floors. Even a reachable target may require more edits than the scheduled adaptation window permits. Section J derives these resource limitations separately from the loss-decrease argument. The controller does not inherit an all-times hard-budget guarantee merely from choosing a decreasing target schedule.

L.4 Released ranking and the calibrated exchange variant

The practical growth and removal densities use candidate measurements and incumbent scores divided by their local parameter costs. In the first-order form,

$$\widetilde{\gamma}_\ell = \frac{\max_c \|u_c\|_2}{\kappa_\ell}, \qquad \widetilde{\rho}_\ell = \frac{\min_j s_j}{\kappa_\ell}.$$

These ratios propose sites for growth and removal. A fixed birth displacement ε predicts linear gain $\varepsilon \|u_c\|_2$. With curvature included, the gain becomes $G_c(\varepsilon)$ from equation (C.1). The scalar $\|u_c\|_2$ alone has not incorporated that displacement.

Consequently, the released norm-based acceptance comparison is a surrogate allocation rule. The certificate in equation (7) belongs to the calibrated variant and additionally includes measurement errors and the interaction between edits. A theoretical optimal-step gain cannot be substituted while taking a different fixed birth step. This distinction permits the practical controller to be analyzed on its own terms without assigning it a guarantee that belongs to another acceptance test.

The implementation sets the proposed growth value to $\|u_c\|$, while materialization uses a fixed $\varepsilon = 10^{-3}$ by default. It therefore omits the factor ε from the first-order gain and does not subtract birth curvature. The field it logs as a loss bound is a surrogate difference, not a certified one. The same applies to feasibility: the parameter marginal sums per-axis sizes on the stored candidate-bearing tensors, while the exact finite difference must use real extents and include cross terms. Recomputing $C(\boldsymbol{n})$ after an edit reports what happened but does not retroactively certify the proposal.

L.5 Pruning by index

A removal applies one index to each represented parameter, buffer, and optimizer-state array. For restriction P, momentum satisfies $P(\mu m + g) = \mu P m + P g$. Materialization promotes a bank row, opens its blocks by equation (L.5), inserts state, and refills the slot. Replacement slots can retain old state, and neither branch protects newborns. Export strips all candidates, yielding an ordinary dense network with revised widths, without claiming that every edit decreased loss.

Removing index j from group ℓ applies that deletion through every slot map of the group. Producer rows are restricted along their output axis. Consumer coordinates reading that index are removed along the input axis, including all positions associated with its multiplicity. Affine normalization parameters and their running statistics undergo the same restriction. A depthwise convolution additionally updates its filter count and group attributes consistently with the surviving input channels.

A count-only request cannot identify the removed channel. If channel 3 is deleted but a normalization tensor is merely truncated at its end, the surviving activations are paired with the wrong affine parameters and running statistics. The error can preserve tensor shapes while changing the function. The shared index map prevents this mismatch by making the identity of the removed coordinate part of the edit. The mathematical consistency argument and the details for grouped operators are given in section H.

L.6 Materialization and candidate replacement

The released operation that promotes a selected candidate into the real index set opens every nonzero consumer block along

$$v_c^{(r)} = -\varepsilon\, u_c^{(r)} / \left\| u_c^{(r)} \right\|_2 . \tag{L.5}$$

Its first-order change is $-\varepsilon \sum_r \left\| u_c^{(r)} \right\|_2$ and its joint step norm is $\varepsilon\sqrt{m}$ when all m blocks are nonzero, as shown in proposition C.1. The routine uses the gradient left by the most recent minibatch backward

pass, not a vector averaged over the sensing window. If a consumer gradient is absent, it falls back to a small random block, for which no measured-direction guarantee applies. The calibrated Euclidean variant instead uses the single joint denominator $v_c = -\varepsilon u_c / \|u_c\|_2$. For a single consumer the two rules coincide.

A fresh candidate with random incoming weights and a zero port replenishes the bank so it retains k dormant proposals. The promoted feature changes the represented function when its port opens. Only its preceding silent embedding is exactly function-preserving under the stated admissibility conditions.

Newly active channels initially have small outgoing weights and can consequently receive small saliency. A protection interval would keep them out of the removal candidate set while their ports develop, but neither audited branch records birth ages or enforces such an interval. Immediate reversal is therefore possible in the released controllers. Ports that remain dormant are re-zeroed after the edit, and the deployed count is recomputed using the resulting real extents.

L.7 Optimizer state, export, and decision records

A structural edit replaces parameter tensors. Momentum and other coordinatewise state arrays must be mapped along the same axes and associated with the replacement parameter objects. Otherwise the optimizer can retain state attached to a tensor no longer used by the model. Existing coordinate histories are preserved by restriction. The released reindexer's policy is exact and is the one used for every reported run: the promoted coordinate is inserted into every optimizer-state array with zero moments, so a newly real channel starts optimization afresh, and the candidate slot it vacates is refilled with a fresh random row that retains the moments accumulated in that slot. Those retained moments can act only on dormant coordinates, and every dormant port is projected to zero after each optimizer step, so they never reach the represented function; on the incoming rows they carry only the weight-decay history of a row that receives no task gradient. Promotion therefore never transfers accumulated port momentum into a real channel, which is the property that matters for the stability of a birth. Section H derives the desired state mapping and the additional age issue for Adam.

The exported network contains real channels only. Its reported count is obtained from the real extent of each tensor axis, deduplicating shared parameters. Subtracting candidate counts independently at neighboring sites can double-count the candidate-by-candidate corner of a tensor. Likewise, a simultaneous removal and growth can change the cost of their shared tensor by a cross term. The exact accounting in equation (H.4) handles both effects, although the released proposal logic does not yet use that finite-difference calculation. Candidate computation remains part of the discovery cost even though candidates are excluded from export. A parameter reduction does not by itself certify reduced FLOPs or latency, which is why the budgeted resource is a choice rather than a convention (section P).

Each structural update emits a decision record containing the selected site or sites, the chosen indices, the measured scores, and the count before and after a committed edit. Rejected proposals and no-ops retain a reason, such as insufficient budget, a width floor, or an inadequate estimated gain. These records distinguish a controller that has reached a stable allocation from one that cannot act because of its schedule or admissible edit set.

M Further Relation to Earlier Work

Learning new features before committing them to a network has a long history. Cascade-Correlation trains candidates against residual errors and installs a selected feature (Fahlman & Lebiere, 1989). TSR carries this idea to interior sites by replacing the output residual with the derivative at the candidate's outgoing connections. The relation is to covariance-based feature selection, rather than an unconditional equality of objectives. The original method centers its signals and aggregates absolute covariances, whereas the Euclidean port geometry used here leads to an ℓ_2 objective. The precise correspondence is derived in section B.

Gradient-informed growth is also central to GradMax (Evci et al., 2022). Function-preserving constructions such as Net2Net (Chen et al., 2016) establish how a larger parameterization can retain an existing solution, while splitting methods and Firefly examine useful directions for expanding a network (Wu et al., 2019; 2020). TSR builds on this perspective by using a port derivative to connect feature proposals with removal decisions under a shared resource constraint. A silent embedding preserves the function. Opening its port along a nonzero descent direction deliberately changes that function.

On the removal side, skeletonization measures the sensitivity to a unit's presence (Mozer & Smolensky, 1988), and Taylor pruning uses activation-gradient products to rank channels (Molchanov et al., 2017). Second-order methods account for the curvature neglected by such linear estimates (LeCun et al., 1989; Hassibi & Stork, 1992). Network Slimming learns which channels to retain through sparsity-inducing regularization (Liu et al., 2017). DepGraph addresses the architectural dependencies that determine which parameters must be removed together (Fang et al., 2023). Our index calculus uses the same need for consistent dependency handling, extended to growth and to the state carried by the optimizer.

Width allocation is itself a form of architecture optimization. MorphNet combines resource-aware shrinking with expansion (Gordon et al., 2018), and AutoSlim searches channel configurations within a slimmable network (Yu & Huang, 2020). These are relevant comparisons because they also redistribute capacity within a chosen family. TSR makes its structural decisions from the evolving task gradient during a discovery run. Dynamic sparse training provides a related example at the level of individual connections, as in RigL (Evci et al., 2020). Channel edits instead yield a dense model with altered dimensions. Their effect on inference cost depends on where the dimensions change, which is why the resource under budget is an argument to the TSR controller rather than a fixed convention (section P).

These lines of work differ most sharply in what is allowed to change and in the object ultimately deployed. Channel pruning removes dimensions from a trained network (Han et al., 2015; Li et al., 2016; He et al., 2017), whereas growth methods enlarge a model when new features appear useful. TSR combines these directions as budgeted exchanges, so a proposed addition must be considered together with the cost of retaining or removing other capacity. Dynamic sparse training reallocates a mask over individual weights and can preserve the ambient tensor shapes throughout training. TSR changes channel dimensions and exports an ordinarily dense network, which makes its resource cost depend on the affected tensor axes and their coupled consumers. This is why an unstructured sparsity result is informative context but not a direct control for width reallocation.

Resource-aware width methods are closer in spirit, but they use different evidence and experimental protocols. MorphNet encourages a resource objective through regularization, while AutoSlim searches subnetworks within a slimmable supernetwork (Gordon et al., 2018; Yu et al., 2019; Yu & Huang, 2020). Whether a pruned architecture's value lies in its shape or in the weights that found it is itself contested (Liu et al., 2019; Frankle & Carbin, 2018), which is the question the static-matched control is designed to answer for width reallocation. TSR is designed to make discrete decisions from the evolving task gradient and to preserve a transparent correspondence between a discovered width vector and a retrainable static control. This correspondence is important for interpreting performance differences. If the online model alone improves, the result may reflect the optimization path rather than the allocation. If a static model at the discovered widths also holds its accuracy under the same training recipe, the allocation itself is implicated. Conversely, a deficit in the static control is evidence against a universal width-improvement claim. The present paper therefore treats these methods as conceptual neighbors while using dense, static-matched, and, where available, random-matched controls to identify the source of an observed effect.

N Experimental Protocol Details

The vision benchmark characteristics extend the motivation in section 6. CIFAR-10 and CIFAR-100 each use 32×32 images and the official 50,000/10,000 train/test split, but differ by a factor of ten in label cardinality. Tiny-ImageNet-200 uses 200 classes of 64×64 images with 100,000 training and 10,000 validation examples. ImageNet-100 is a fixed 100-class subset with approximately 126,689 training and 5,000 validation images at 224×224. CIFAR training uses random 32×32 crops with four-pixel padding and horizontal flips. Tiny-ImageNet uses random 64×64 crops with eight-pixel padding and horizontal flips. ImageNet-100 uses random resized 224×224 crops and horizontal flips, followed at evaluation by resizing to 256 and taking a 224×224 center crop. Every transform is shared across its three arms. CIFAR and Tiny-ImageNet use a 3×3, stride-one stem without the ImageNet max-pooling reduction where applicable. ImageNet-100 retains the standard 7×7, stride-two stem and max pool. VGG uses global adaptive pooling, while MobileNetV2 and EfficientNet-B0 reduce the initial stride on the lower-resolution datasets.

Vision models train for 100 epochs with cross-entropy, SGD with Nesterov momentum 0.9, initial learning rate 0.1, weight decay 5×10^{-4}, and per-step cosine decay. The usual batch size is 128, reduced to 64 for ImageNet-100. The discovery run uses $k = 4$ candidates per eligible group, a

100-step structural window, minimum width 8, and a target equal to 85% of the reference parameter count. The ceiling reaches this target halfway through training. Automatic mixed precision is applied consistently across matched arms when enabled. These values describe the archived campaign rather than a tuned optimum for each dataset. In particular, the 85% target is held fixed across every vision configuration by design, as section 6 states; the archived vision campaign contains no sweep over it, and none of the conclusions below rest on that value being a limit.

For forecasting, Weather, Electricity, and Traffic (Lai et al., 2018; Zhou et al., 2021) contain respectively 21, 321, and 862 variates, and all three use chronological 70/10/20-percent partitions. Validation and test windows may read the preceding split for input context, but their targets remain inside the held-out interval. Per-variate means and standard deviations are fitted on the training partition only. TCN and TSMixer use lookback 96. PatchTST and iTransformer use lookback 336, matching their standard long-range configuration. Forecast horizons are 96 and 720, with the four-backbone compression frontier run first at horizon 96.

The TCN has four residual dilated-convolution blocks with dilations $1, 2, 4, 8$, hidden width 64, and BatchNorm. PatchTST uses patch length 16, stride 8, $d_{\text{model}} = 128$, three blocks, eight heads, and $d_{\text{ff}} = 512$. iTransformer uses variate tokens with the same model width, block count, and head count, with $d_{\text{ff}} = 256$. TSMixer uses four blocks and feature-mixing width 256. TSR edits the TCN channel groups, the per-block feed-forward widths of PatchTST and iTransformer, and the feature-mixing hidden widths of TSMixer. Residual dimensions spanned by LayerNorm, attention head arithmetic, input variates, and lookback axes remain fixed according to section K. A preliminary TCN ablation also compares channel-mixing, channel-independent, and no-RevIN variants, but it does not replace the four-backbone benchmark. Two natural alternatives were excluded from the backbone set: DLinear (Zeng et al., 2023) has no hidden width for TSR to reallocate, and the data-dependent FFT shapes in TimesNet (Wu et al., 2023) do not fit the present static tracing assumptions. RevIN is used where specified by the architecture configuration (Kim et al., 2021).

Forecast models train for 50 epochs with MSE, Adam, weight decay 10^{-4}, and per-step cosine decay. PatchTST and iTransformer use learning rate 10^{-4}; TCN and TSMixer use 10^{-3}. Batch sizes are 32 for Weather, 16 for Electricity, and 8 for Traffic. TSR uses $k = 4$, a 100-step structural window, minimum width 8, and a budget schedule ending after 30% of training. The compression frontier ranges from mild reduction to each architecture's measured admissibility floor: budget ratios 0.95 to 0.66 for PatchTST, 0.95 to 0.56 for iTransformer, and 0.85 to 0.10 for TCN. TSMixer uses dataset-specific ranges because its feature-mixing cost depends on the variate count. Infeasible architecture–dataset budgets are rejected before training rather than silently under-delivered.

The benchmark choices follow from the controls rather than from scale. The four vision datasets vary classification difficulty and image scale while keeping repeated matched-control training feasible, which is why ImageNet-100 is used in place of ImageNet-1K; the paper therefore makes no claim about full-scale ImageNet. The four backbones were chosen because their operator differences exercise the coupling mechanism, which is more informative than adding closely related residual networks, and only groups admitted by the calculus are plastic, with unsupported side paths left fixed. On the forecasting side ETT is omitted because its seven variates and short held-out interval give the smallest models and the noisiest single-seed comparisons, the regime in which a budgeted reallocation is least informative; LayerNorm and task-interface constraints leave different fractions of each backbone plastic, so every budget sweep is restricted to the reachable range of its own admissible groups. Pruning-only methods, unstructured dynamic sparsity and one-shot width search (Han et al., 2015; Evci et al., 2020; Yu et al., 2019) remain contextual comparisons because they use a different action space or amortize a different search procedure; they cannot replace the dense, static-matched and random-matched controls that identify the source of an observed gain.

The dense reference fixes every width. The online arm includes candidate sensing and structural edits, while its deployed count excludes dormant candidates. The static-matched arm rebuilds the recorded width vector through the same coupling map and trains from a fresh initialization. Forecasting additionally defines a random-width control by drawing per-group factors uniformly from $[0.5, 1.5]$ and scaling them by binary search until its count is within one percent of the discovered model. It uses the same rebuilding code and inherited optimizer configuration. This control is reported only when its checkpoint exists; it is not imputed for the archived vision campaign. The archived vision controls take the widths at the selected discovery checkpoint, whereas the forecasting controls take the final converged widths, because a forecasting run's best validation epoch can precede the completion of its budget schedule.

Every reported matched comparison checks the width source, deployed parameter equality, achieved budget, and exact zero magnitude of dormant ports. Parameter count is computed from exported tensor extents. FLOPs are reported as a diagnostic rather than inferred from the parameter reduction. The archived campaign contains one seed for several configurations; the repeats that do exist are collected in section S. Single-seed entries are individual controlled runs, not estimates of variance, and they do not support statistical equivalence claims. CIFAR evaluation uses the official test split during checkpoint selection because the loader has no separate validation split. We therefore describe those values as held-out accuracies rather than untouched final-test estimates. Multi-seed means, standard deviations, or equivalence tests must be restricted to configurations for which independent runs are actually available.

O Vision Result Details and Artifact Audit

Table O.1 gives the resource and top-5 measurements omitted from the main table. The static-matched and online models have the same deployed count and shape in every row, though the online arm's stored tensors are larger than that count because they carry the candidate ports that the deployed count excludes. We define $\Delta F = 100(F_{\mathrm{C2}}/F_{\mathrm{ref}} - 1)$, so a negative value denotes a FLOP reduction and a positive value a FLOP increase. Top-5 accuracies are measured from the archived checkpoints together with the top-1 values, on the full held-out split under each run's own transform and on the graph each run was trained with. The ResNet-18 CIFAR-10 entry has no recomputed FLOP value.

Table O.1: Detailed vision resources and top-5 accuracy. Parameter counts are exact exported counts; ΔF is negative for a FLOP reduction and positive for an increase. The top-5 column lists Ref/C2/TSR, measured from the archived checkpoints.

Backbone	Data	P_{ref}	P_{C2}	ΔF (%)	Top-5 (%)
ResNet-18	C-10	11,173,962	9,497,686	–	99.83/99.91/99.85
	C-100	11,220,132	9,536,305	$+1.80$	93.85/93.84/93.88
	Tiny-IN	11,271,432	9,580,578	-19.61	85.00/84.66/84.96
	IN-100	11,227,812	9,543,190	$+12.04$	95.90/95.58/95.88
VGG-16-BN	C-10	14,728,266	12,513,661	$+27.06$	99.79/99.75/99.68
	C-100	14,774,436	12,553,018	$+27.73$	91.31/91.26/91.67
	Tiny-IN	14,825,736	12,601,243	$+47.50$	81.51/81.34/81.92
	IN-100	14,774,436	12,557,980	$+17.61$	97.04/96.94/97.22
MobileNetV2	C-10	2,236,682	1,872,999	-11.08	99.74/99.72/99.79
	C-100	2,351,972	1,992,323	-11.34	90.76/90.73/91.27
	Tiny-IN	2,480,072	2,059,393	-42.02	80.45/79.87/80.79
	IN-100	2,351,972	1,925,595	-14.46	95.30/95.26/95.04
EfficientNet-B0	C-10	4,020,358	3,417,087	$+40.17$	99.70/99.62/99.68
	C-100	4,135,648	3,515,277	-3.56	90.95/90.84/90.64
	Tiny-IN	4,263,748	3,623,936	$+3.52$	81.34/81.79/81.63
	IN-100	4,135,648	3,456,824	-14.31	95.12/95.04/95.04

The distribution behind the headline is as follows. Thirteen of the sixteen differences fall within 0.8 points of the dense reference and three land above it; the three that cost more than a point are MobileNetV2 on CIFAR-100, Tiny-ImageNet and ImageNet-100, at -1.29, -1.32 and -1.22, and excluding them the mean difference is -0.14 points. The online arm exceeds the dense reference in 6 of the 16 cells and its own static twin in 7, over a range of -0.40 to $+1.84$ points; its two largest positive values, $+1.84$ and $+1.39$, both fall on MobileNetV2, the backbone where the budget bites hardest. Forward FLOPs range from a 42.0% decrease for MobileNetV2 on Tiny-ImageNet to a 47.5% increase for VGG-16-BN on the same dataset.

On ImageNet-100 the reference parameter counts of VGG-16-BN, MobileNetV2 and EfficientNet-B0 coincide with their CIFAR-100 counts, because these three backbones change only the initial stride between the two resolutions and no channel width. ResNet-18 does coincide in every width but its stem, a 7×7 convolution here against a 3×3 one on the smaller images, which accounts for the $7{,}680$-parameter difference between its two rows.

The table also explains why a claim of joint improvement must name its resource. C2 reduces parameters in every row but reduces FLOPs in only seven of the fifteen measured. VGG moves capacity toward computationally expensive feature maps in all four of its cells. MobileNetV2 reduces both resources in all four. The controller used parameter cost, so neither outcome is anomalous, and replacing that cost with an audited FLOP cost changes the outcome accordingly (section P).

Every accuracy in table 1 and here is measured directly from the archived checkpoint, by loading each of the three arms and scoring it on the full held-out split under the transform its own run was trained with, rather than read from any training summary.

P BUDGETING COMPUTE INSTEAD OF PARAMETERS

The cost C of equation (2) and the per-index marginal κ derived from it are defined against any additive resource. Section 7 reports what a parameter budget does to compute; this appendix reports what the same controller does when it is given a compute budget instead. Nothing else changes: the schedule, the structural interval, the candidate count, the training recipe and the seed are those of section N, the annealed cap is 85% of the dense model's forward FLOPs rather than 85% of its parameters, and the densities of equation (11) divide by κ_{flops} rather than κ_{params}. The two backbones are the ones whose parameter-budgeted runs sit at opposite ends of table O.1: EfficientNet-B0, which produced the largest FLOP increase on CIFAR, and MobileNetV2, which reduced both resources at once.

Table P.1: The same controller under the two budgets, EfficientNet-B0 and MobileNetV2, 85% cap in the budgeted resource. ΔP and ΔF are the changes in deployed parameters and forward FLOPs of the retrained architecture against the dense reference; Δ is C2 minus Ref in top-1 accuracy. Every accuracy is measured on the full held-out split by the procedure of section O, so the two budgets and table 1 are directly comparable.

Backbone	Data	Budget on	ΔP (%)	ΔF (%)	Ref	C2	TSR	Δ
EfficientNet-B0	C-10	parameters	-15.01	$+40.17$	90.78	90.61	90.21	-0.17
		FLOPs	-23.59	-14.11	90.78	90.78	90.00	0.00
	C-100	parameters	-15.00	-3.56	68.51	67.99	67.60	-0.52
		FLOPs	-20.98	-13.08	68.51	67.38	68.67	-1.13
MobileNetV2	C-10	parameters	-16.26	-11.08	90.99	90.78	90.72	-0.21
		FLOPs	-16.69	-15.00	90.99	90.43	90.54	-0.56
	C-100	parameters	-15.29	-11.34	68.09	66.80	68.19	-1.29
		FLOPs	-13.33	-15.36	68.09	67.78	67.92	-0.31

Budgeting compute reduces compute. All four cells land between 13.1 and 15.4% below their dense reference in forward FLOPs, including the EfficientNet-B0 CIFAR-10 cell where a parameter budget had raised FLOPs by 40.2%, a swing of 54 points in the budgeted resource; that cell also retrains to 90.78%, the dense reference's own accuracy to the second decimal. The accuracy cost of the switch is between zero and 1.13 points, and on MobileNetV2 CIFAR-100 it runs the other way: the worst CIFAR cell in table 1 at -1.29 points under a parameter budget becomes -0.31 under a FLOP budget, with more compute saved rather than less.

What the two backbones do with the switch differs, and the difference is the mechanism. Splitting each network's editable groups at its midpoint and summing widths, EfficientNet-B0 under a parameter budget *adds* channels to the early half, $+5.2\%$ on CIFAR-10 and $+4.9\%$ on CIFAR-100, while taking 7.0% and 7.6% from the late half: parameters are cheap early and expensive late, so the controller moves capacity forward, into exactly the wide, high-resolution layers where computation is spent. Under a FLOP budget that sign reverses, the early half losing 4.7% and 3.4% and the late half 14.4% and 14.7%. MobileNetV2 shows the complementary case. Its two budgets place capacity almost identically, -5.0% against -4.7% early and -15.3% against -15.1% late on CIFAR-10, because its parameter cost and its compute cost are already distributed alike, which is why its parameter-budgeted runs reduced FLOPs by 11% without being asked to. The allocations diverge exactly where the two resources do, and agree where they do not.

Two measurements deserve their exact statement. First, MobileNetV2 lands on its cap, 0.00% and 0.42% inside it, while EfficientNet-B0 overshoots by 1.04% and 2.26%. The controller ranks and projects with κ_{flops}, a per-index marginal read from the traced shapes, whereas the reported figure is an exact forward count of the exported network; the two agree on MobileNetV2's depthwise groups and disagree on EfficientNet-B0's squeeze-and-excitation branches, where the normalizer and bias arithmetic is not part of the marginal. This is the compute-currency form of the accounting distinction in section H.2, and a controller projecting with an exact finite difference would land on the cap in both. Second, these are four cells of two backbones at one seed each, run to show that the cost function is substitutable and to measure what the substitution buys, not to establish a compute-reduction result across the suite.

Q Structured Pruning at a Matched Count

The control that places the compression claim against the standard alternative is structured pruning of the same dense reference down to the same deployed count. It is built on the coupling groups of section H rather than on an external library's dependency analysis, so both arms edit exactly the same plastic set with the same width floor and the same exact accounting, and the only thing pruning cannot do is add capacity. Four criteria rank channels for removal: the ℓ_1 norm of a channel's producer weights (Li et al., 2016), the mean absolute BatchNorm scale of its group (Liu et al., 2017), the first-order activation–gradient product (Molchanov et al., 2017), which is the controller's own removal statistic, and DepGraph's group norm through its released implementation (Fang et al., 2023). Removal is greedy and iterative, ranked globally after per-group mean normalization, and stops at the matched count; every pruned model lands within one channel's cost of its target. The right-hand blocks of tables 1 and 2 report the vision cells and the forecasting knees.

Two training modes follow, and the distinction decides what each comparison means. In the fine-tuned mode the pruned network keeps the reference's trained weights and is fine-tuned for 40 epochs, which is the practical pipeline and costs 140 epochs in total including the dense reference it requires. In the from-scratch mode the pruned width vector is reinitialized and trained for 100 epochs under the recipe C2 itself uses (Liu et al., 2019). Only the second isolates the allocation, because only there do both arms begin without the dense model's weights.

Fine-tuned pruning is ahead of the retrained arm, by 0.21 to 0.55 points on average depending on the criterion, and it ends above its own dense reference in seven of the twelve cells. That is the correct expectation rather than a surprise, because the arms are not matched in what they consume: a fine-tuned pruned model has been given a 100-epoch dense reference plus 40 further epochs, and it starts from weights that have already solved the task, whereas C2 is a 100-epoch discovery run followed by a 100-epoch retrain that never sees those weights.

The arm matched on cost is the online one, a single 100-epoch run and therefore 40 epochs cheaper than the fine-tuned pipeline and free of any dense pre-training. It reaches that pipeline's accuracy: 0.14 points behind ℓ_1 on average, 0.09 behind network slimming, 0.04 ahead of DepGraph and 0.30 behind Taylor, all within the seed spread of section S. It is also the stronger of the two TSR arms here, averaging 0.12 points below its reference against the retrained arm's 0.37, and finishing above the dense reference in five of twelve cells against two. A model that matches a pruning pipeline while costing less than it and requiring no trained network to start from is the practical statement this table supports.

Where the allocations themselves are compared, at matched count with neither arm inheriting weights, the discovered widths are ahead. Against ℓ_1 from scratch, C2 is better in seven of the twelve cells, by a mean of 0.26 points and a median of 0.18. The advantage is not uniform: it exceeds a point on ResNet-18/CIFAR-100, ResNet-18/Tiny-ImageNet and VGG-16-BN/Tiny-ImageNet, and reverses in five cells, most sharply on the two Tiny-ImageNet cells of the compact backbones. At one seed per cell, with most margins inside the spread of section S, the aggregate supports the claim that a task-directed allocation is better on average than a magnitude-ranked one, not that it wins in any particular cell.

The forecasting knees of table 2 behave differently, and the difference is informative. Over the eight Electricity and Traffic knees the two allocations are level: C2 is better than ℓ_1 fine-tuned in three cells and than ℓ_1 from scratch in four, with mean differences of 0.24 and 0.35 percent of the metric, which is the same order as the seed spreads reported in section S. On the four Weather knees, where three seeds are available, C2 is ahead of the best pruning arm on the TCN and TSMixer and behind it on the two transformers. The vision advantage therefore does not carry over to the tightest budgets, where a larger share of the network must move; that boundary is the honest limit of the allocation claim, and a wider sweep at those budgets is the experiment that would settle it.

One cell deserves its name. MobileNetV2 on CIFAR-100 is among the three cells below a point in table 1, at -1.29 points, and pruning from scratch reaches -1.00 there. Both allocations land in the same place, so the shortfall belongs to the budget rather than to the search, and the same cell recovers to -0.31 when the budget is placed on compute instead (section P).

R Forecasting Result Details

Table R.1 gives the random-width control at each knee reported in table 2. C3 draws a per-group width allocation at random and rescales it until its deployed count matches the discovered model to

within one percent, then trains it from scratch under the discovered model's recipe. It is the control that separates a task-directed allocation from an arbitrary nonuniform one at the same budget; the dense reference cannot do this, because it differs from C2 in size as well as in shape.

Table R.1: Random-width control at the knee. C3 is a random allocation matched to C2's parameter count within one percent. Test MSE at horizon 96; lower is better, so a negative C2 − C3 favors the discovered allocation. The final column expresses that difference as a percentage of C2. The TCN's Weather knee is absent because its random control missed the parameter target in both seeds and fails the validity check.

Backbone	Data	C2	C3	C2 − C3	rel. (%)
TCN	Elec.	0.3034	0.3132	−0.0097	−3.2
	Traffic	0.7276	0.7445	−0.0169	−2.3
PatchTST	Elec.	0.1327	0.1332	−0.0006	−0.45
	Traffic	0.3729	0.3733	−0.0005	−0.13
iTransformer	Elec.	0.1341	0.1342	−0.0001	−0.07
	Traffic	0.3787	0.3791	−0.0004	−0.11
	Weather	0.1614	0.1604	+0.0010	+0.6
TSMixer	Elec.	0.1563	0.1645	−0.0082	−5.2
	Traffic	0.4985	0.5203	−0.0219	−4.4
	Weather	0.1582	0.1582	0.0000	0.0
PatchTST	Weather	0.1512	0.1504	+0.0008	+0.5

The knees of table 2 distribute as follows. On Electricity and Traffic, TSMixer reaches 81–86% fewer parameters while remaining inside the band, the two transformers reach 25–35%, and the TCN leaves the band by 17–19%, the only cells whose reachable compression resembles the vision setting. On Weather, whose 21 variates leave far more of each network unclaimed by the task interface, the same TCN reaches a knee at 86%, TSMixer at 45%, and the transformers at 34–35%; for the latter three the knee is the tightest budget in the sweep, so accuracy never left the band. The online arm's own error on Weather sits 13–32% above the reference, further than on the other datasets, which is the cost of searching and training at once under an aggressive anneal; a longer anneal, or a protection interval for recently edited groups, addresses it directly.

Three qualifications belong with this table. The first is the seed: where repeated runs exist, the standard deviations are of the same order as the smaller margins, and section S gives them per cell. The knee is one row per configuration selected by a fixed tolerance; over the complete frontier the sign of C2 − C3 is close to even, so the eleven-cell result should be read as a statement about the chosen operating point rather than about every budget. Finally, the TCN's random control missed its parameter target at six tighter Electricity and Traffic budgets and at the Weather knee itself. The cause is arithmetic rather than a failure of the search: C3 draws a random factor per group and binary-searches one global scale, but each width is an integer bounded below by the floor of 8, so at a target of 32K parameters the achievable counts form a coarse lattice and no scale lands within one percent of the target. Those cells are excluded here rather than reported with a marker, and the Weather knee's C2 and online arms, whose counts passed every check, are reported in table 2. The Weather frontiers of the three other backbones never leave the tolerance band down to the tightest budget swept, so their knees are the floor of the sweep rather than a measured limit.

The complete frontier, every budget from 0.95 down to each architecture's admissibility floor, is generated from the checkpoints by the table-generation script together with the per-cell validity checks on parameter equality, control matching, and exact port dormancy. Reference runs are single-seed in at least one cell, so the reference's own seed variance is not reflected in the differences above.

S Seed Replicates

The main tables report one run per cell. This appendix gives the repeats that were run to establish whether the differences in those tables are separable from the seed: on the vision side both CIFAR cells of MobileNetV2, the backbone that carries the largest shortfalls, and on the forecasting side every Weather knee. All values are measured by the procedures of section O and section R; nothing here is inherited from a training summary.

Table S.1 measures the noise floor of the vision protocol rather than one cell of it. MobileNetV2 was chosen for that purpose because it is the smallest of the four backbones and carries all three of the differences above one point in table 1, so its spread is a conservative estimate for the suite rather

Table S.1: Vision seed replicates. MobileNetV2 on both CIFAR datasets, three seeds, reference and static-matched arm trained independently under the shared recipe at the counts of table O.1. Top-1 accuracy on the full held-out set; Δ is C2 minus Ref. The seed-42 row is the one reported in table 1.

Data	Seed	Ref	C2	Δ
C-10	42	90.99	90.78	-0.21
	43	90.82	91.13	$+0.31$
	44	90.71	90.84	$+0.13$
	mean $\pm$ s.d.	90.84 ± 0.14	90.92 ± 0.19	$+0.08 \pm 0.26$
C-100	42	68.09	66.80	-1.29
	43	68.17	67.56	-0.61
	44	68.08	67.40	-0.68
	mean $\pm$ s.d.	68.11 ± 0.05	67.25 ± 0.40	-0.86 ± 0.37

than a favorable one. The dense reference is the stable quantity: it reproduces to a standard deviation of 0.14 points on CIFAR-10 and 0.05 on CIFAR-100, a total range of 0.28 and 0.09 across three seeds. The discovered architecture is the more variable arm, at 0.19 and 0.40 points, as expected of a smaller model trained from scratch.

Read against table 1, this settles which of its rows are effects and which are parity. Ten of the sixteen differences are at most 0.28 points, inside the range the reference covers on its own, and are reported as parity for that reason; a sign that changes across seeds, as the CIFAR-10 difference does at $+0.08 \pm 0.26$, is the signature of such a row rather than a defect in it. All three differences above one point are more than an order of magnitude larger than this spread. One of them is the CIFAR-100 cell repeated here: it is negative in all three seeds and averages -0.86 points, smaller than the -1.29 of the reported seed but far outside the spread. The other two, on Tiny-ImageNet and ImageNet-100, were not repeated, so those comparisons carry the CIFAR spread of the same backbone across datasets and are indicative rather than measured. Seed 42 is the less favorable draw for the method in both cells, and it is the one the main table reports.

Table S.2: Forecasting seed replicates at the Weather knees, horizon 96. Test MSE, mean $\pm$ s.d. over the repeats; lower is better. The TCN row is the cell whose random control misses its parameter target and is therefore excluded from table R.1, reported here for the spread only.

Backbone	Red. (%)	TSR	C2	C3
PatchTST	34.0	0.1753 ± 0.0017	0.1512 ± 0.0019	0.1504 ± 0.0012
iTransformer	35.0	0.2083 ± 0.0050	0.1614 ± 0.0011	0.1604 ± 0.0019
TSMixer	45.0	0.1814 ± 0.0060	0.1582 ± 0.0027	0.1582 ± 0.0019
TCN	85.6	0.2724 ± 0.0135	0.2372 ± 0.0077	0.2439 ± 0.0095

The forecasting repeats, table S.2, are read the same way. On Weather the C2 spread of 0.0011 to 0.0027 is larger than every C2 $-$ C3 margin in table R.1, which is why those cells are reported as ties rather than as wins, and the same holds for the TCN cell excluded there. On Electricity the available repeats give a C2 spread of ± 0.0007 for iTransformer and a C3 spread of ± 0.0098 for the TCN; the first is an order of magnitude below the TSMixer and TCN margins of 0.0082 and 0.0097 that carry the random-control result, and the second is of the same order as the TCN margin it bounds, so that one cell is the weakest of the eight. The online arm, measured at a search-converged architecture rather than retrained, carries the largest spread in three of the four cells.

Two gaps remain. The repeats estimate the spread of the protocol, not of every cell, so a per-cell interval is not available for the eleven vision cells and the Electricity and Traffic budgets away from the knee that were run once. And the forecasting reference is single-seed in at least one cell, so its own variance does not enter the differences reported there.

T CASE STUDIES: TWO DISCOVERY RUNS

The tables report outcomes. This appendix opens two of the runs behind them and shows the mechanism at work: ResNet-18 on CIFAR-100, the run drawn in figure 2, and VGG-16-BN on CIFAR-100, a plain chain with no residual coupling and therefore no trunk groups. Both used the same recipe (a parameter ceiling annealed to 85% of the dense count over the first half of 100 epochs,

$k = 4$ candidates per group, one structural update every 100 steps) and both exported an architecture 15.0% smaller than the dense reference. Every width, score and cost below is replayed from the archived decision log; the replay reproduces the logged deployed parameter count before and after all 389 updates of each run, the annealed ceiling at every logged decision, and the exported checkpoint's tensor shapes. The exported checkpoint is the best-validation epoch rather than the last, so figures that tally edits stop at the export step (37,400 for ResNet-18, 37,000 for VGG-16-BN) and the trajectory figures mark it. In every figure ResNet-18 is on the left or top and VGG-16-BN on the right or bottom. These are single runs, one seed each; they illustrate the mechanism and are not evidence of generality.

T.1 The architecture's trajectory

Each structural update leaves the model at a point in width space. Taking the editable groups' widths in $\log_2$ relative to the dense reference as coordinates, the run traces a path through a 12-dimensional space (13 for VGG-16-BN), and a principal-component analysis of that path is a latent representation of the architecture as it is being edited. It describes where capacity sits, not what the network computes.

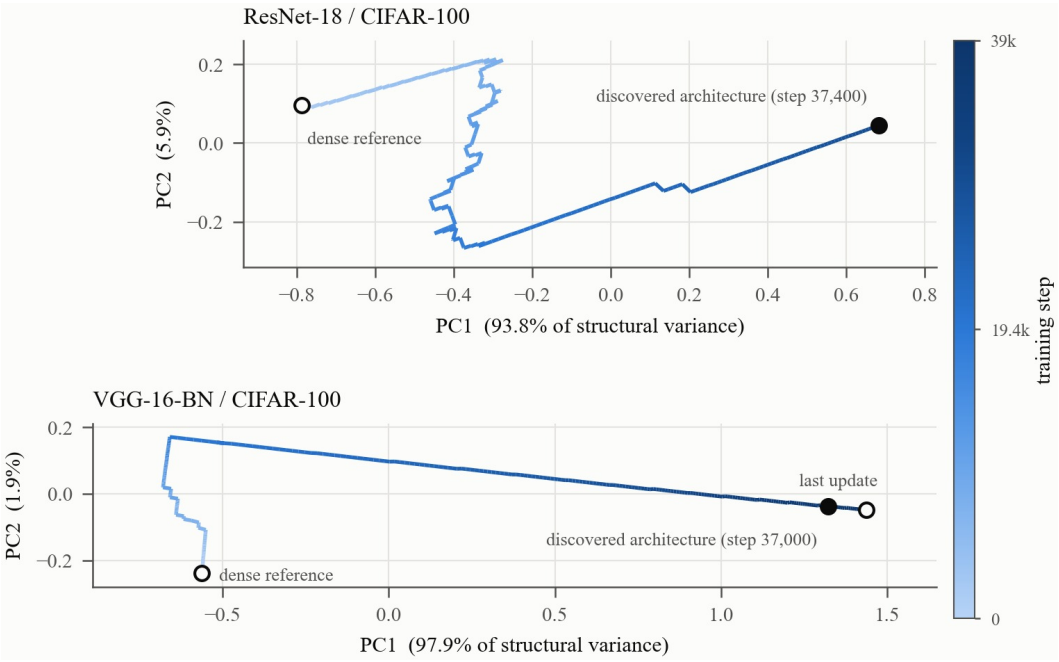


Figure 3: The path through width space, in each run's first two principal components, on equal axes and colored by training step. ResNet-18 (top) carries 93.8% of its structural variance on PC1; the path wanders while the ceiling is falling and then moves in a straight line once it settles. VGG-16-BN (bottom) carries 97.9% on PC1, and its anneal phase barely leaves the neighborhood of the reference. The filled marker is the exported architecture; where the run continued past it, the open marker is the last logged update. The components describe these two trajectories, not a general basis.

T.2 Where the edits went

T.3 The controller's signals over training

The log records, for each window, the candidate the controller selected and the incumbent it removed, so the points below are the accepted edits, not the population they were chosen from. Shading spans the updates that entered the feasibility regime.

T.4 Score against cost

T.5 What the three arms represent

The figures above describe where capacity sits. The last two ask what the networks holding that capacity compute, by running the same 2,048 CIFAR-100 test images through the dense reference,

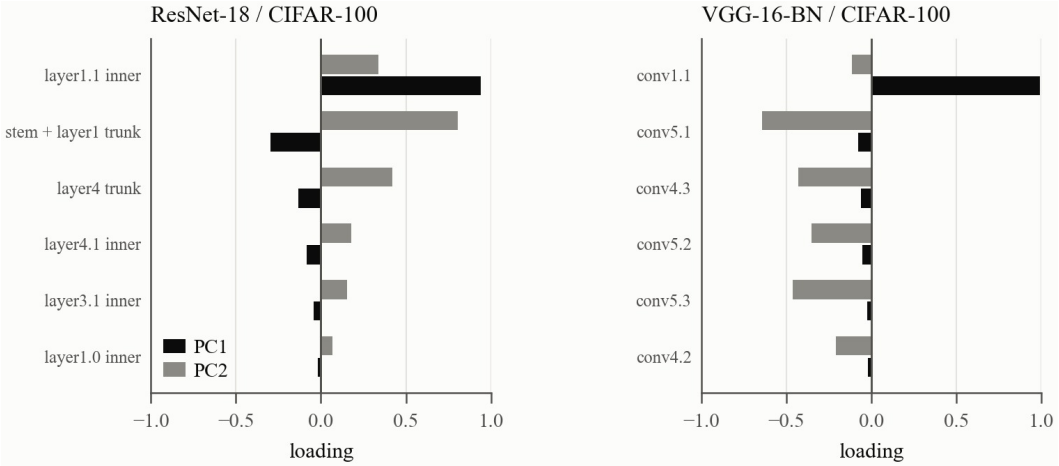


Figure 4: What the principal axes are made of: loadings of the six groups with the largest $|\mathrm{PC1}|$. In both runs PC1 is a single site set against the deep groups: layer1.1's inner width ($+0.94$) against the stem/layer1 trunk (-0.30) and the layer4 trunk (-0.13) for ResNet-18; conv1.1 ($+0.99$) against the block-4 and block-5 convolutions for VGG-16-BN. To first order each discovery is one transfer, from the deep, expensive groups to one early, cheap one.

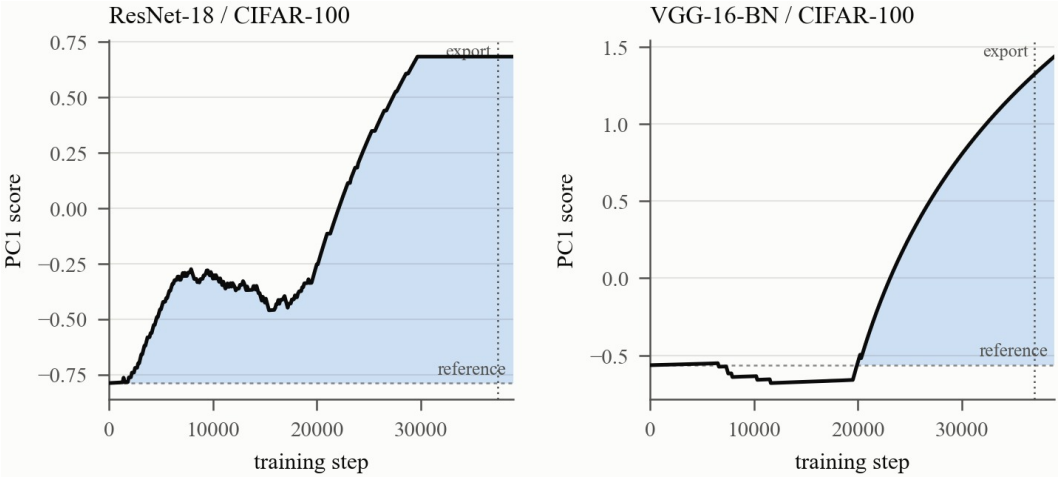


Figure 5: The dominant coordinate over training; the dotted line marks the export step. ResNet-18 climbs in two stages, stalling and partly reversing while the ceiling is still falling and resuming once it settles. VGG-16-BN is flat until the ceiling reaches its target at step 19,500 and then rises monotonically, still rising when training ends.

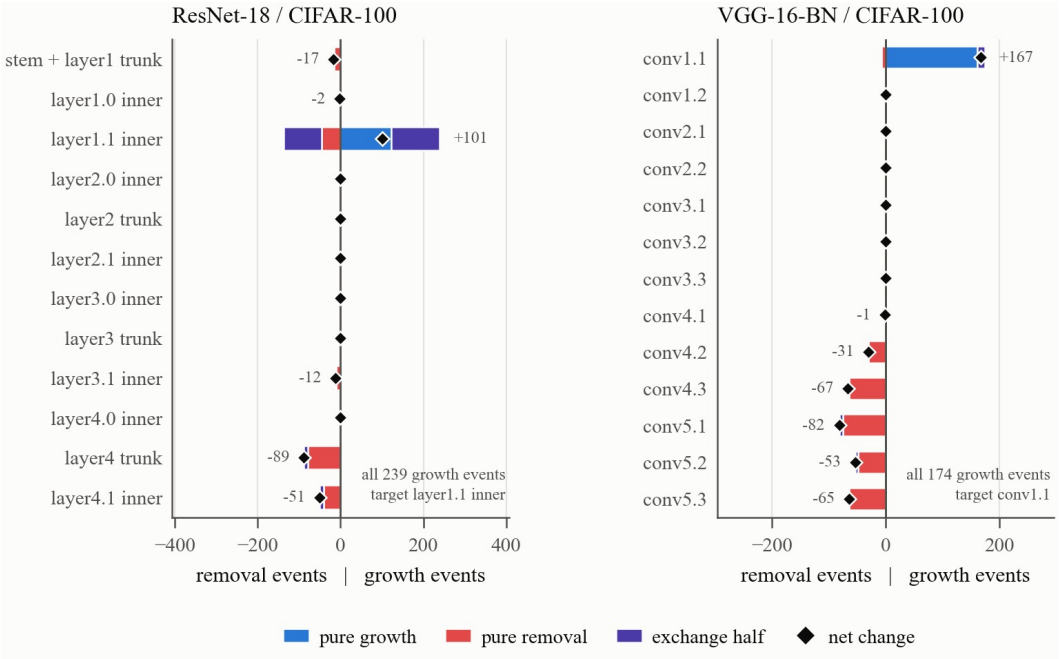


Figure 6: Edits per coupling group up to the exported checkpoint: removals to the left of zero, growth to the right, split into pure edits and the two halves of an exchange; the marker is the net width change. Every growth event in both runs targets a single group, 239 at layer1.1's inner width and 174 at conv1.1, and in ResNet-18 that group is also its most-pruned site (138 removals), consistent with the absence of a newborn-protection interval in the released controller (section L.6). ResNet-18 exchanges freely (132 up to export); VGG-16-BN almost never does (13), moving capacity through pure removals during the anneal and pure growths after it.

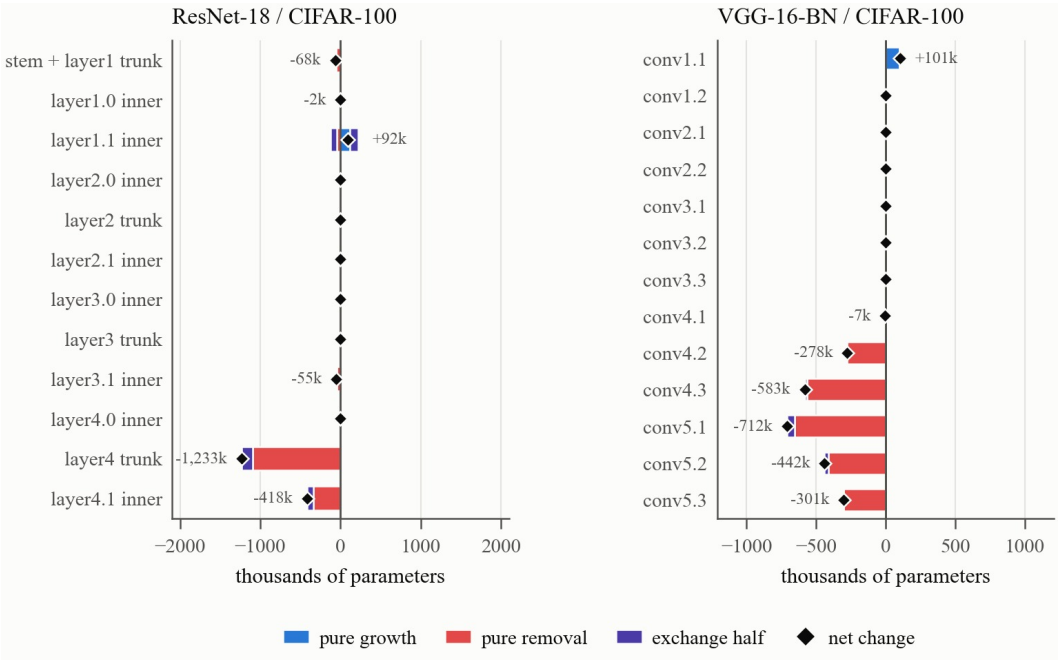


Figure 7: The same edits weighted by the exact parameter cost of the index moved. One channel costs 1,154 parameters at layer1.1 and 13,970 at the layer4 trunk of ResNet-18; 606 at conv1.1 and 9,201 at conv5.1 of VGG-16-BN. Counts and parameters therefore disagree: the growth site receives the most events but a small share of the parameters, and the reduction is paid almost entirely by the deep, wide groups (−1.23M at the layer4 trunk, −712k at conv5.1). Event counts are not a proxy for capacity moved.

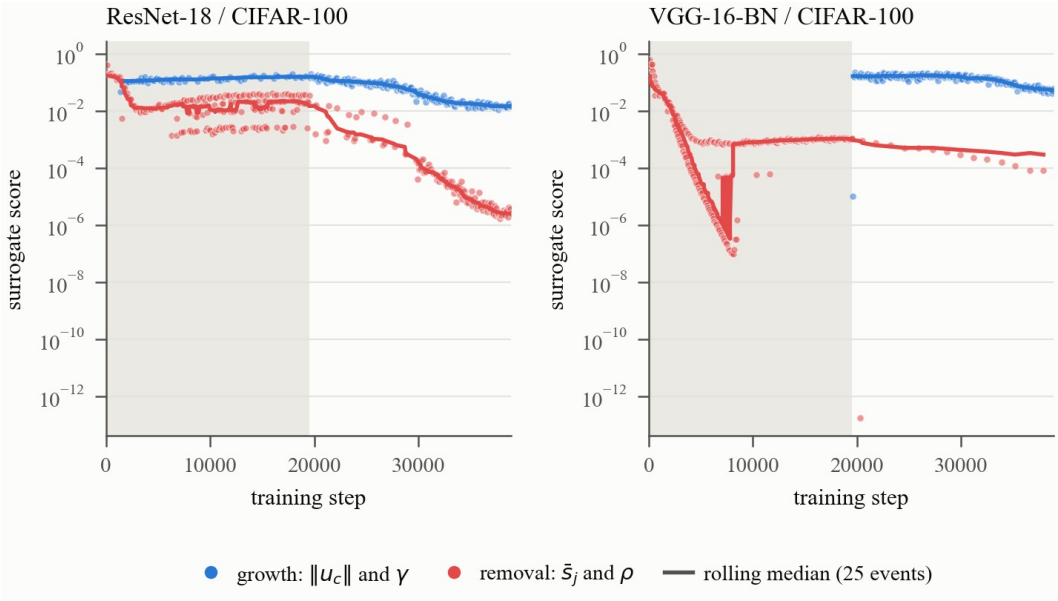


Figure 8: The surrogate score of each accepted edit against training step, with a rolling median over 25 events per side. The removal score $\bar{s}_j$ falls about four decades in ResNet-18 and two in VGG-16-BN as the inner problem converges, which is why the released controller's pure-removal test is relative to each group's own median rather than an absolute floor (equation (L.4)): any fixed threshold is eventually crossed by every channel.

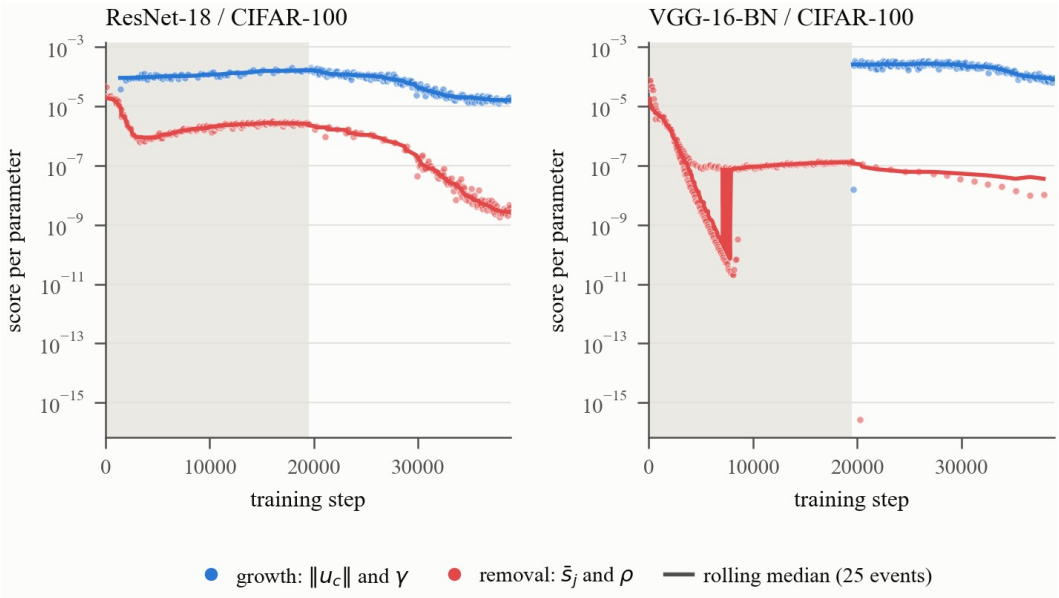


Figure 9: The same edits as densities, score per parameter of the index moved, which is the quantity the controller ranks by. The growth density it accepts sits far above the removal density it pays throughout: the median ratio γ/ρ over accepted exchanges is about 650 in ResNet-18 (132 exchanges) and about 6,000 in VGG-16-BN (14), and in both runs the gap widens rather than closing. Neither run approaches the equimarginal condition of theorem F.1; both stop because training ends.

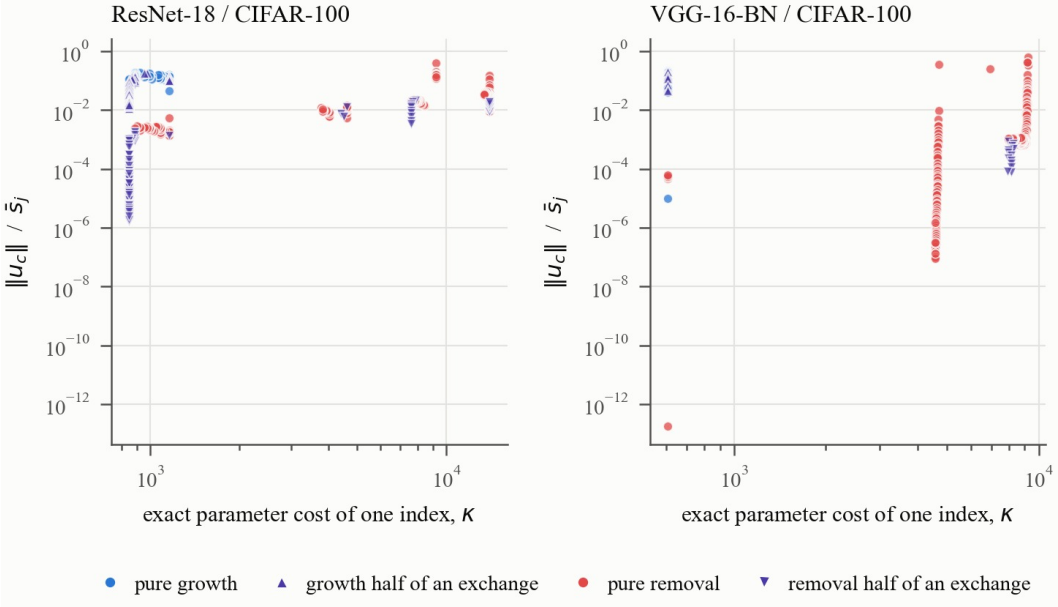


Figure 10: Surrogate score against the exact parameter cost of the index moved, for accepted edits. Growth signals $\|u_c\|$ and removal scores $\bar{s}_j$ share an axis because Case C compares them directly; they remain surrogates in different units, not commensurable loss changes. In VGG-16-BN the costs separate into two clusters an order of magnitude apart, the 606-parameter channels of conv1.1 where every growth lands and the several-thousand-parameter channels of blocks 4 and 5 where nearly every removal lands; ResNet-18's costs spread over the same range with its exchanges concentrated at the cheap end.

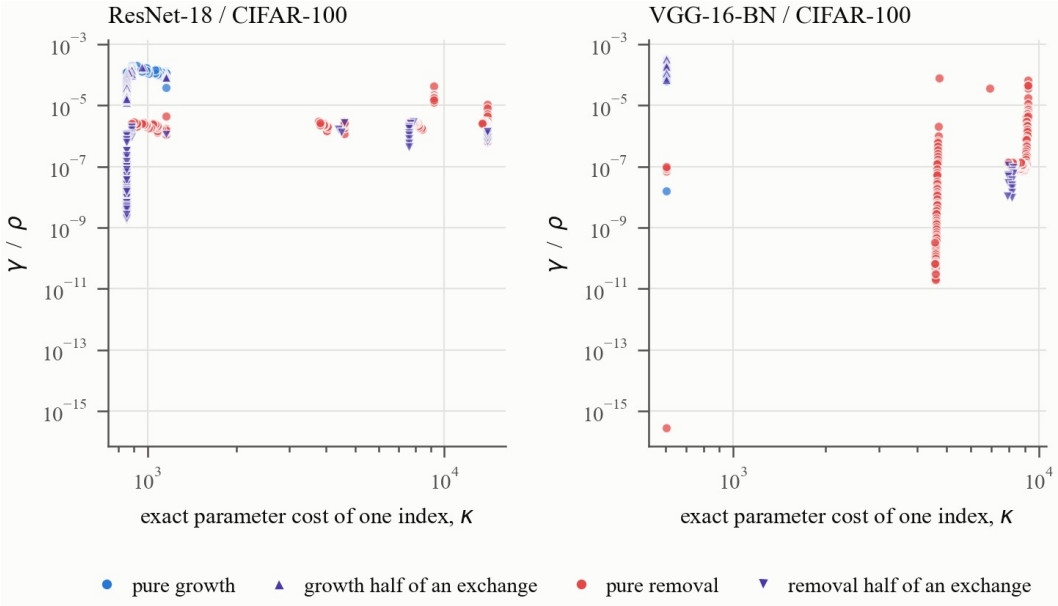


Figure 11: Score per parameter against cost. Dividing by cost lifts the cheap site's edits and lowers the expensive sites', which is the asymmetry the exchange rule ranks on: in both runs the accepted growths sit at the low-cost end and above the accepted removals. Accepted edits only.

the static-matched arm (C2) and the online arm, and comparing their hidden representations at probed stages, with spatial locations treated as samples. One seed per arm; the two matched arms share widths but not an initialization, so similarity below one between them includes ordinary seed-to-seed variation and is not by itself evidence that plasticity changed the representation.

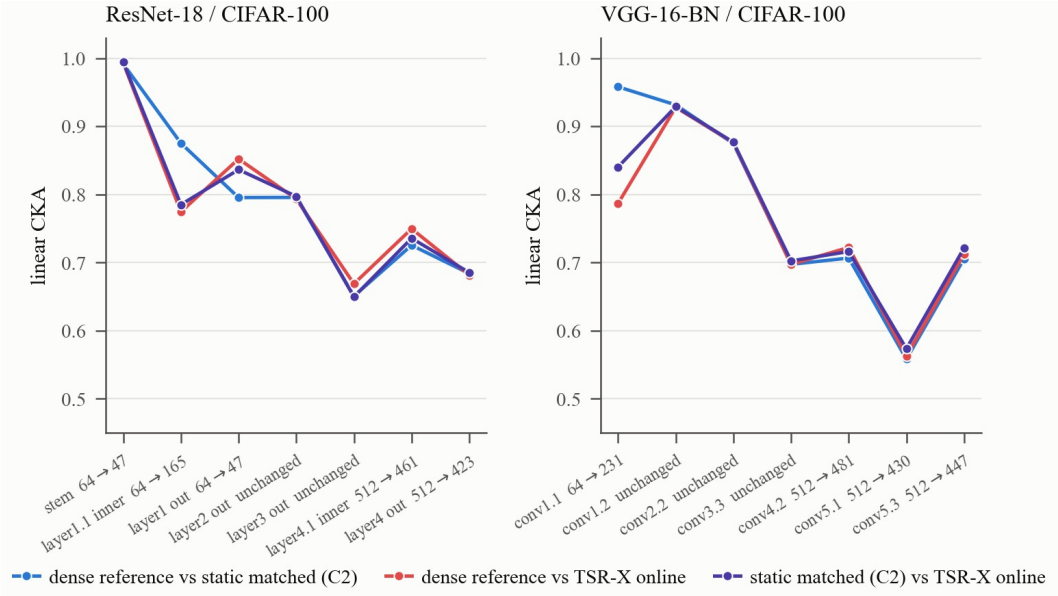


Figure 12: Linear CKA (Kornblith et al., 2019) between the three arms at each probed stage; the tick labels give the stage's reference → exported width. CKA is invariant to the width of the representations compared, which is what makes arms with different channel counts comparable. Similarity falls with depth in both runs and is lowest at the most-pruned deep stage of VGG-16-BN (conv5.1, 0.56 to 0.57), but the reference-vs-C2 pair is as far apart there as the pairs involving the online arm, so that divergence is seed variation at least as much as it is pruning.

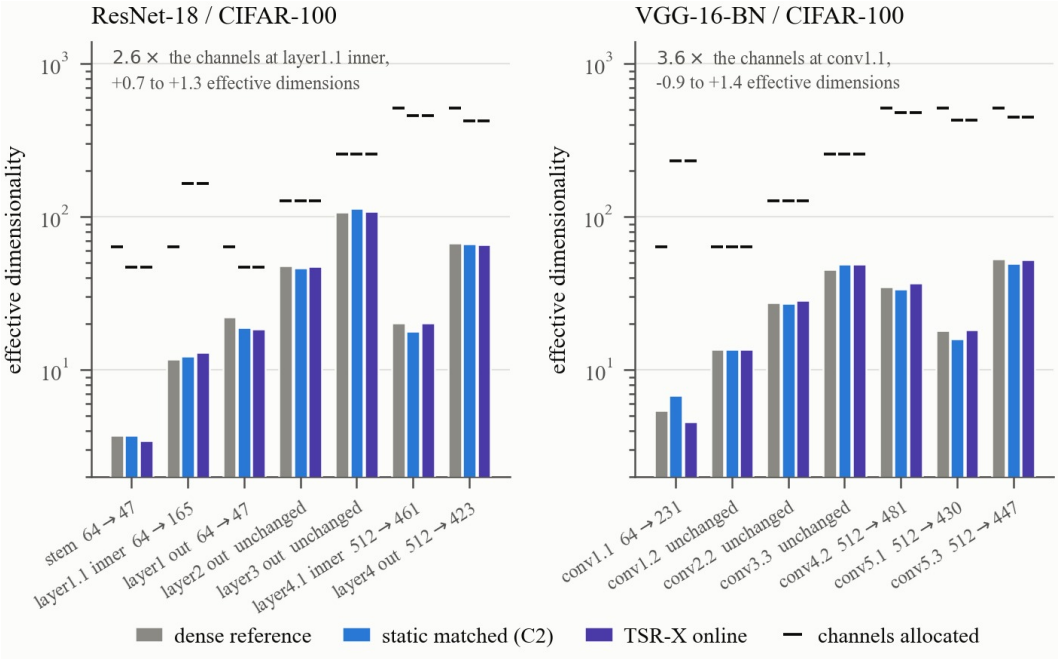


Figure 13: Effective dimensionality (participation ratio of the feature covariance spectrum, Gao et al., 2017) of each arm at the same stages, against the channels allocated (dashes). Every stage uses a small fraction of its width: layer4.1's inner width spans about 20 dimensions across 512 channels and conv5.1 about 18, which is the empirical form of the redundancy the reduction exploits. The grown sites are no exception. $2.6\times$ the channels at layer1.1 bought $+0.7$ to $+1.3$ effective dimensions; $3.6\times$ at conv1.1 bought -0.9 to $+1.4$.