GRAD SCHOOL: TEACHING A STUDENT ITS TEACHER'S CIRCUIT

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge distillation supervises a student on its teacher's answers, not on the computation behind them. We introduce GRaph Attribution Distillation (GRAD), which makes the teacher's circuit a training target: differentiable attribution graphs over MLP neurons are coarsened into supernodes grouped by function, so both models share labelled nodes, and the student is trained, alongside standard KD, to route influence between them as the teacher does. Distilling Llama-3-8B into Llama-3.2-1B on two-digit addition, GRAD improves on standard KD on every family, beats it and three other baselines on the held-out mean (0.55 against at most 0.50), lifts never-trained multiplication from 0.51 to 0.95, and raises a 3B student's held-out mean from 0.83 to 0.88; a scrambled teacher target does not reproduce the gain. Mechanistically, GRAD's student develops features as specific as a label-trained student's without gold labels, and on multiplication it corrects a failure of standard KD, whose students hedge instead of answering in the evaluated input format, the format in which the graph term is computed; a loss that rewards answering in that format produces the same correction. These analyses yield a checklist for objectives that aim to transfer a mechanism.

1 INTRODUCTION

Knowledge distillation trains a small model to reproduce a large one's output distribution (Hinton et al., 2015). Its refinements, from feature hints (Romero et al., 2015) to on-policy objectives (Gu et al., 2024; Agarwal et al., 2024), change what is matched or on which inputs, but the supervision remains a function of what the teacher says, or of its hidden states as vectors, not of how it computes. For a task with a definite algorithm the difference matters: two models can agree on every training output and disagree beyond it. Mechanism-matching objectives address this. Causal distillation matches the teacher under aligned interchange interventions (Wu et al., 2022), and Circuit Distillation aligns components paired by their ablation effects (Wadhwa et al., 2025); both improve on standard distillation. Whether the student thereby acquires the teacher's mechanism, rather than whatever else the auxiliary loss rewards, is rarely tested directly.

We introduce GRaph Attribution Distillation (GRAD), which makes the teacher's attribution graph (Ameisen et al., 2025; Lindsey et al., 2025) a training target (Figure 1). It has three parts (Section 3). Attribution graphs over raw MLP neurons are built differentiably inside the training loop, so gradient reaches the student's weights through its edges. Supernodes come from an ANOVA that scores each neuron's activations over a grid of probe prompts against a short list of functional roles, so the same labelled nodes exist in an 8B teacher and in 1B and 3B students of different width, depth and neuron basis. And a Jensen–Shannon divergence between each supernode's incoming routing profiles in the two models is added to the KD loss. The aim is to teach the student its teacher's circuit: not only what to answer, but which parts of the computation feed which.

Distilling Llama-3-8B into Llama-3.2-1B on two-digit addition, with three seeds per arm, GRAD raises accuracy on held-out multiplication, an operation never trained on, from 0.512 ± 0.117 under standard KD to 0.955 ± 0.007 (Section 4). It has the best held-out mean of five arms, ahead of supervised fine-tuning (SFT) and the TinyBERT and CKA recipes (Jiao et al., 2020; Kornblith et al., 2019); it peaks in about half the steps and keeps the gain. KD given GRAD's compute, $14\times$ as many steps, does not close the gap. A scrambled teacher target, keeping the teacher's numbers but pairing each supernode with the wrong sources, destroys training: two of its three seeds never exceed the

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

untrained student in distribution. The same target, unchanged, raises a 3B student's held-out mean from 0.826 under KD to 0.875.

We then analyse how GRAD changes the student, comparing seed-matched KD, SFT and GRAD students (Section 5), and find two effects. First, the teacher's graph does what soft probabilities cannot: without any gold label, GRAD's sum features become as specific as label training makes them (0.37, against 0.38 for SFT and 0.28 for KD), and its routing ends closer to the teacher's than KD's. Second, on held-out multiplication GRAD corrects a failure of standard KD. The KD student answers only 46.6% of multiplication prompts, hedging with a "?" on the rest, though it is right on 94.7% of those it answers; GRAD answers 99.9% (Figure 4). The correction is made in layers 0–7 at the answer position and lives in the evaluated input format: our protocol, fixed by the KD baseline's own sweep, drops BOS in training but keeps it in evaluation and attribution, so the graph term is the only loss computed in that format, and a loss that directly rewards answering there gives the same held-out gain (0.951 ± 0.005).

Contributions. (i) GRAD, a distillation objective on the teacher's attribution graph, and functional supernodes that give an 8B teacher and 1B and 3B students the same labeled nodes (Section 3). (ii) A controlled study, with three seeds, four baselines, compute-matched and scrambled-target controls and a second student, in which GRAD nearly doubles held-out multiplication accuracy over standard KD (Section 4). (iii) A mechanistic analysis of how GRAD changes the student: internals of label-trained quality without labels, and a held-out gain carried by a commit-or-hedge decision in the evaluated format, with a five-step checklist to analyze any mechanism-matching objective (Section 5).

2 Related Work

Output and representation distillation. Soft-label distillation (Hinton et al., 2015) has been extended to intermediate representations (Romero et al., 2015; Sanh et al., 2019; Sun et al., 2019), attention maps, hidden states and relations between examples (Zagoruyko & Komodakis, 2017; Park et al., 2019; Wang et al., 2020; Jiao et al., 2020), and representational similarity (Kornblith et al., 2019); for language models, reverse-KL and on-policy variants change which distribution is matched, and on whose samples (Gu et al., 2024; Agarwal et al., 2024). We compare against the TinyBERT and CKA recipes as the representation-level references.

Distilling mechanisms. Two objectives instead target the teacher's causal structure. Causal distillation (Wu et al., 2022) trains the student to reproduce the teacher's outputs under aligned interchange interventions. Circuit Distillation (Wadhwa et al., 2025), the closest work to ours, pairs teacher and student components with similar ablation effects and aligns their representations, improving on standard distillation for entity tracking and theory of mind. GRAD differs in what it matches (attribution routing between functional supernodes, not component representations) and in how it aligns the models (an automatic functional ANOVA, not ablation-based matching; Section 3.3). Above all, we pair the objective with a mechanistic audit and controls: a scrambled teacher target, a content-free loss computed on the graph term's inputs, and evaluation in every input format (Section 5). To our knowledge such controls are absent from prior work on mechanism-matching distillation. We offer them as a contribution: an auxiliary loss can act through where it is computed rather than what it matches, so the check applies to any such objective, ours included.

What distillation transfers. What a student acquires often differs from what its objective names. Students can generalise well while matching their teachers poorly (Stanton et al., 2021), which teacher properties transfer must be measured (Ojha et al., 2023), and chain-of-thought rationales help distillation even when placed after the label and permuted, so their reasoning is not what helps (Wadhwa et al., 2024).

Circuits, attribution and arithmetic. Circuits are found by causal mediation and activation patching (Vig et al., 2020; Meng et al., 2022; Wang et al., 2023; Conmy et al., 2023), by attribution patching, its linear approximation (Nanda, 2023; Syed et al., 2024), or over learned features (Bricken et al., 2023; Marks et al., 2025). Attribution graphs (Ameisen et al., 2025; Lindsey et al., 2025) put these on a common footing: a weighted graph of direct effects computed in one pass, coarsened by hand into labelled supernodes. We make this a training target by attributing to raw MLP neurons rather than transcoder features (Dunefsky et al., 2024), and by replacing manual supernodes with

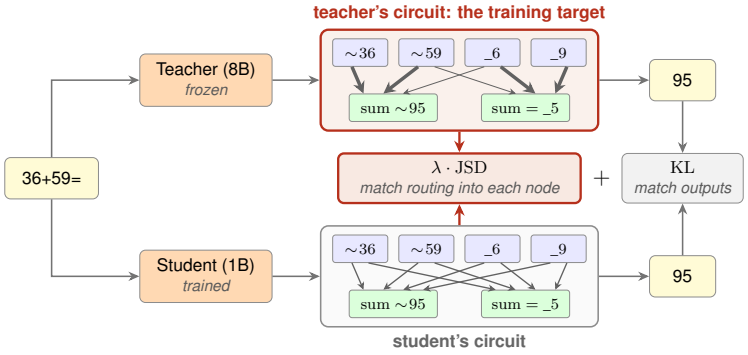

Figure 1: **GRAD teaches a student its teacher's circuit.** Both models see the same prompt. Supernodes are functional categories found independently in each model by ANOVA over all two-digit prompts (Section 3.3), so the 8B teacher and the 1B student share the same labelled nodes; on top of standard KD, the student is trained to match the teacher's routing into each supernode. Edge widths are schematic: teacher sharp, student diffuse (compare Figure 2).

ones computable in both models (Section 3.3). In arithmetic, attention moves operands to the final token and MLPs compute the result (Stolfo et al., 2023); numbers are encoded as Fourier features (Zhou et al., 2024) or on a helix (Kantamneni & Tegmark, 2025); and Lindsey et al. (2025) trace our "`36+59=`" prompt. Nikankin et al. (2025) find that our teacher does arithmetic with a bag of heuristic neurons, a distributed mechanism that routing over a few supernodes may not capture.

What training changes, and how evaluations mislead. Fine-tuning often wraps or enhances existing mechanisms rather than building new ones (Jain et al., 2024; Prakash et al., 2024), and model stitching localises such changes (Lenc & Vedaldi, 2015; Bansal et al., 2021). The first (BOS) token acts as an attention sink that models depend on (Xiao et al., 2024; Barbero et al., 2025); scores swing with prompt format (Sclar et al., 2024); first-token probabilities can disagree with generated answers (Wang et al., 2024); and accuracy-only scoring rewards answering over abstaining (Kalai et al., 2025). A held-out gain can therefore be a shortcut (Geirhos et al., 2020). None of this work considers training losses that see different input formats, which is where Section 5 locates our gain.

3 GRAD: Graph Attribution Distillation

3.1 Overview

Standard knowledge distillation constrains only the *output* of the student. We add a constraint on the *internal computation*: for a prompt we build an attribution graph over MLP neurons in each model, coarsen it into a small set of labeled supernodes that exist in both models, and require the student to route causal influence between those supernodes in the same proportions as the teacher (Figure 1).

Let $\mathcal{T}$ be the frozen teacher and $\mathcal{S}$ the student with parameters $\theta_{\mathcal{S}}$. For a prompt x the pipeline of Sections 3.2–3.3 (edge weights in Appendix A) produces one $K \times K$ supernode adjacency matrix per model, $W^{\mathcal{T}}(x)$ and $W^{\mathcal{S}}(x;\theta_{\mathcal{S}})$, whose row t says where supernode t draws its input from. Supernodes are identified by a label ("the ones digit of the first operand", "the value of the sum") rather than by neuron index, so the rows and columns of the two matrices correspond although the models differ in width, depth and neuron basis. Dividing row W_t by its ℓ_1 norm gives supernode t's *routing profile*, the fraction of its incoming causal influence contributed by each other supernode, and the graph loss is the mean Jensen–Shannon divergence between the two models' profiles over the supernodes $\mathcal{A}$ present in both, the teacher's detached; the JSD is symmetric, penalizing the student both for missing an edge the teacher uses and for inventing one it does not, and bounded, so one edge on which the models place disjoint mass cannot dominate. The full objective adds it to the temperature-scaled KD term over the positions $\mathcal{P}$ whose next token is real, with $p_p^{\mathcal{T}} = \mathrm{softmax}(z_p^{\mathcal{T}}/\tau)$ and $p_p^{\mathcal{S}}$ the two models' output distributions at position p:

$$\mathcal{L}(\theta_{\mathcal{S}}) = \underbrace{\frac{\tau^2}{|\mathcal{P}|}\sum_{p\in\mathcal{P}} \mathrm{KL}\big(p_p^{\mathcal{T}} \,\|\, p_p^{\mathcal{S}}\big)}_{\mathcal{L}_{\mathrm{KD}}} + \lambda \underbrace{\frac{1}{|\mathcal{A}|}\sum_{t\in\mathcal{A}} \mathrm{JSD}\left(\frac{W_t^{\mathcal{T}}(x)}{\|W_t^{\mathcal{T}}(x)\|_1} \,\middle\|\, \frac{W_t^{\mathcal{S}}(x;\theta_{\mathcal{S}})}{\|W_t^{\mathcal{S}}(x;\theta_{\mathcal{S}})\|_1}\right)}_{\mathcal{L}_{\mathrm{graph}}}. \quad (1)$$

3.2 ATTRIBUTION GRAPHS OVER MLP NEURONS

One forward pass becomes a directed weighted graph whose nodes are localized computational units and whose edge A_{ij} is the *direct* linear effect of node j on node i, holding fixed everything off the direct path, following the circuit-tracing construction (Ameisen et al., 2025; Lindsey et al., 2025). A node i is a triple (ℓ_i, p_i, n_i) of layer, token position and neuron index. It carries a *write vector* $v_i^{\text{out}} = a_i \cdot W_{\text{down}}[:, n_i]$, what the neuron adds to the residual stream, and a *read direction* $v_i^{\text{in}} = \partial a_i / \partial h$, the residual-stream direction it responds to (for Llama's gated MLP, Appendix A). Edges are gradient–activation products (Nanda, 2023). For a target node i we form the linearized pre-activation $f_i = \langle h_{p_i}^{(\ell_i)}, \text{sg}[v_i^{\text{in}}] \rangle$, with the read direction stop-gradiented so that only the *input* to the neuron is differentiated, and set

$$A_{ij} \;=\; \Big\langle \nabla_{m_{p_j}^{(\ell_j)}} f_i, \;\; v_j^{\text{out}} \Big\rangle, \tag{2}$$

where $m^{(\ell_j)}$ is the output of MLP block ℓ_j. Token-embedding nodes use the embedding as v^{out}, and logit nodes use $f_c = \langle h_{-1}^{(L)}, W_U[:, c] - \overline{W_U} \rangle$ with the mean-subtracted unembedding column, for the smallest set of top tokens covering 0.95 of $\text{softmax}(z/\tau)$. Since $n_{\text{pos}} \times d_{\text{mlp}}$ candidates per layer is far too many, we keep the top 10% per layer by write-vector norm in the teacher and 1% in the student, and zero the BOS position so the attention sink is never selected.

We attribute to raw neurons rather than to the transcoder features of the published pipeline (Dunefsky et al., 2024). Equation (2) needs only a node's residual-stream contribution to be linear in its activation, which neurons satisfy exactly, and it avoids the transcoder's approximation error, which must otherwise be carried as error nodes. The cost is polysemanticity, and that is what the supernode step repairs; in the published work, too, the interpretable unit is a group of features rather than a feature, and the graph term of (1) is defined over groups only.

3.3 SUPERNODES: GROUPING NEURONS BY FUNCTION

Raw neuron graphs are too large and too model-specific to compare across a 1B and an 8B model. Where the published work draws supernodes by hand from feature visualizations, we group neurons by a functional criterion computed in each model from one list of candidate roles, so that the same labels exist in both. The list is task-specific; we give it for two-digit addition, whose mechanisms in language models are well studied (Section 2).

Every neuron is evaluated on all 10^4 prompts "a+b=" with $a, b \in \{0, \ldots, 99\}$, giving an *activation grid* $T_i \in \mathbb{R}^{100 \times 100}$ with $T_i[a, b]$ its activation on that prompt (the node panels of Figure 2). For a prompt with operands (a^*, b^*) and sum $s^* = a^* + b^*$, each category c is a binary mask M_c over the grid, one supernode each:

category	$M_c[a,b] = 1$ iff	category	$M_c[a,b] = 1$ iff
first-operand value	$a = a^*$	sum value	$a + b = s^*$
first-operand ones digit	$a \equiv a^* \pmod{10}$	sum ones digit	$a + b \equiv s^* \pmod{10}$
second-operand value	$b = b^*$		
second-operand ones digit	$b \equiv b^* \pmod{10}$		

A neuron's score for category c is the fraction of the variance of T_i that M_c explains, the one-way ANOVA η^2 (equivalently its squared correlation with the mask):

$$s_i^c \;=\; \frac{\big(\langle T_i - \bar{T}_i,\; M_c - \bar{M}_c \rangle\big)^2}{\|T_i - \bar{T}_i\|_2^2 \;\cdot\; \|M_c - \bar{M}_c\|_2^2} \;\in\; [0, 1]. \tag{3}$$

Neurons are ranked by *specificity*, the margin over the best competing category, $\delta_i^c = s_i^c - \max_{c' \neq c} s_i^{c'}$, and supernode S_c is the top n_{per} neurons by δ_i^c. The margin is what keeps the supernodes functionally disjoint: a neuron responding to a broad band of a scores well on several correlated categories at once, and only the margin separates a ones-digit detector from a magnitude detector that happens to correlate with one.

The two sum categories are defined by what their neurons *write* rather than read: among the neurons with positive mask score, candidates are ranked by how closely the direct logit attribution

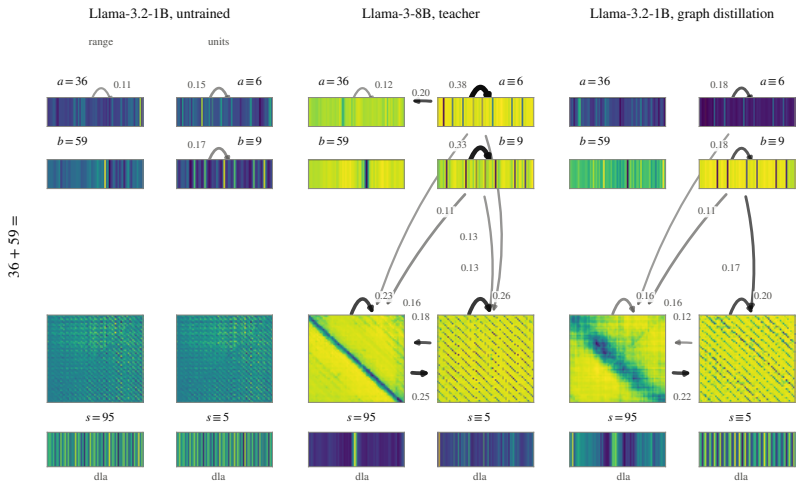

Figure 2: Supergraphs for "36+59=" with the six ANOVA supernodes of Section 3.3 and the edge weights of (4), drawn source to target with width proportional to the weight and omitted below 0.10; $\equiv$ is modulo 10. The teacher's sum grids are sharp – $s = 95$ an anti-diagonal, $s \equiv 5$ a periodic lattice – and it routes 0.11–0.26 into them from the operand supernodes, where the untrained student's grids are noise and its largest such edge is 0.03. GRAD recovers both, at 0.11–0.22, from the teacher alone: its sum features are as specific as those of a student fine-tuned on gold labels and more specific than standard KD's (Table 2). The right panel is that student, not a separate run.

$\mathrm{softmax}(v_i^{\mathrm{out}\top} W_U)$, restricted to the number tokens 0–200, matches in KL a point mass at s^* (sum value) or the uniform distribution over $\{n : n \equiv s^* \pmod{10}\}$ (sum ones digit). The same score also picks, for each sum target, the neurons that write it most directly; these two output-side supernodes are drawn in the bottom row of Figure 2 for reading and are not part of the trained target. The figure shows the same six-node structure in all three models, sharp in the 8B teacher and diffuse in the untrained 1B student.

4 Experiments

4.1 Setup

Models and task. We distil Meta-Llama-3-8B-Instruct into Llama-3.2-1B-Instruct on two-digit addition, prompts "a+b=" with $a, b \in \{0, \ldots, 99\}$; Section 4.3 repeats the comparison with Llama-3.2-3B-Instruct as the student.

Held-out families. Generalization is measured on four families never seen in training: *three-operand* and *four-operand addition* (sums of two-digit numbers), *three-digit addition*, and *multiplication* (a two-digit by a one-digit number), each scored by greedy decoding and exact match on a fixed 2000-prompt subset of its test split.

Baselines. Standard KD from the teacher's logits; KD plus the TinyBERT recipe (Jiao et al., 2020) and KD plus linear CKA (Kornblith et al., 2019) on the residual streams, the representation-level references; and supervised fine-tuning on the gold answers (SFT). GRAD adds the graph term of (1) at $\lambda = 3$ with the six supernodes of Section 3.3, ten neurons each; a scrambled-target control is defined in Section 4.2.

Training and evaluation. Every arm shares one trainer, data order, optimizer protocol and evaluation, and carries three seeds. The protocol (bf16, AdamW at 10^{-5} with warmup and cosine decay, batch 32, 100 steps; Table 13) was selected by sweeping the standard-KD baseline alone and inherited by every other arm; the representation weights and λ come from their own sweeps (Appendix B). Training sequences omit the BOS token, the one departure from the standard convention; evaluation and attribution keep it, so the graph term is the only loss computed on inputs formatted as at evaluation, which Section 5 shows matters.

Table 1: Accuracy on the training family (two-digit addition) and the four held-out families, each seed read at its own best in-distribution step (checkpoint selection on the training family's test split, which uses no out-of-distribution data); seed mean ± sd over three seeds. Bold is the best mean in a row; italics are below the untrained student.

	untrained student	teacher	standard KD (3 seeds)	KD + TinyBERT (3 seeds)	KD + CKA (3 seeds)	SFT (3 seeds)	GRAD ($\lambda = 3$, 3 seeds)
2-digit add (train)	0.670	0.960	0.989 ± 0.003	0.987 ± 0.004	0.987 ± 0.003	0.995 ± 0.001	**0.999 ± 0.001**
selected step, per seed	—	—	50/35/40	50/35/40	45/35/40	40/45/55	70/35/30
3-operand add	0.199	0.884	0.440 ± 0.027	0.433 ± 0.029	0.444 ± 0.007	**0.487 ± 0.005**	0.466 ± 0.032
4-operand add	0.045	0.769	0.120 ± 0.010	0.121 ± 0.011	**0.126 ± 0.005**	0.108 ± 0.002	0.122 ± 0.010
3-digit add	0.487	0.922	0.608 ± 0.026	0.601 ± 0.026	0.603 ± 0.018	**0.670 ± 0.016**	0.661 ± 0.011
2×1-digit mult	0.124	0.894	0.512 ± 0.117	0.478 ± 0.101	0.502 ± 0.101	0.725 ± 0.031	**0.955 ± 0.007**
held-out mean	0.214	0.868	0.420 ± 0.044	0.408 ± 0.040	0.419 ± 0.032	0.498 ± 0.004	**0.551 ± 0.015**
peak held-out mean (step)	—	—	0.431 ± 0.025 (30)	0.421 ± 0.023 (30)	0.426 ± 0.023 (30)	0.498 ± 0.005 (40)	**0.586 ± 0.004** (20)
2-digit add (train), last step	—	—	0.985 ± 0.002	0.982 ± 0.003	0.979 ± 0.003	0.993 ± 0.001	**0.999 ± 0.001**

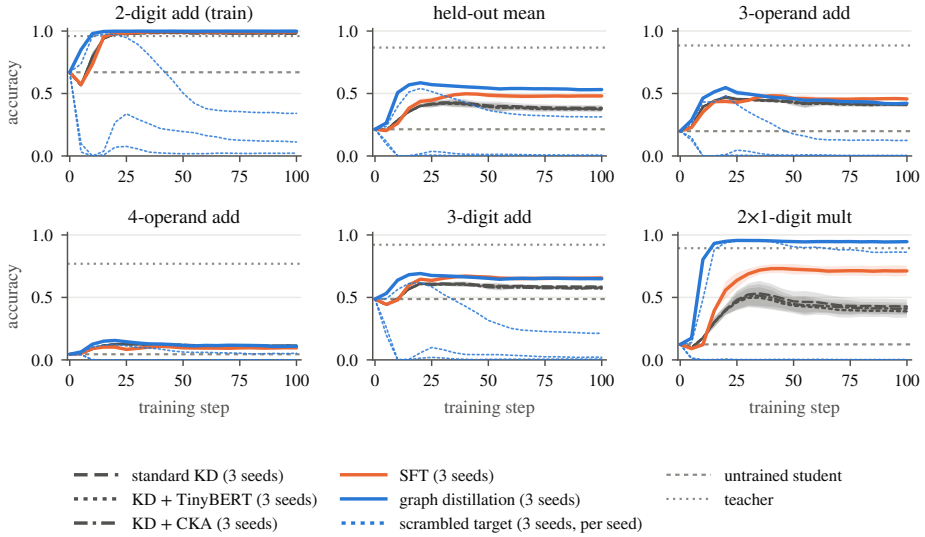

Figure 3: Accuracy against training step: in-distribution, the mean over the four held-out families, and each family. The three KD-based baselines share the grey of the reference method and differ in line style; SFT is orange; GRAD is blue. Seed means with a ±1 sd band for the three-seed arms. The dashed rule is the untrained student and the dotted rule the teacher.

4.2 Generalization from two-digit addition

The gain is on the held-out operation. Every objective saturates the training family (Table 1), so the arms separate out of distribution, and there the pattern is sharp. On multiplication, an operation no arm ever trains on, GRAD reaches 0.955 ± 0.007, where standard KD reaches 0.51 and gold-label fine-tuning 0.73; on the three addition families it improves on standard KD and is at parity with the best baseline, and every arm stays far below the teacher. Section 5 analyzes how the multiplication gain arises.

Representation matching does not reproduce it. The TinyBERT and CKA recipes constrain the same residual stream the graph term reads from and do not move from standard KD on any family. SFT, which fits the gold answers, sits between KD and GRAD under the shared protocol and is stronger under its own best tokenization convention (Section 5).

It arrives early and stays. GRAD reaches its best held-out mean at step 20, against steps 30 to 40 for every baseline, and does not give the multiplication result back: at step 100 it is still at 0.946 where the baselines have eased to 0.39–0.71 (Figure 3). Its seed spread on that family is an order of magnitude below its baselines' (± 0.007 against ± 0.10–0.12), and the graph gradient stays near-orthogonal to the KD gradient throughout (Appendix C).

Sensitivity. Multiplication is flat in λ from 1 to 10 (0.950–0.956) and the held-out mean moves by at most 0.03 from 1 to 6; only at $\lambda = 0.3$, where the graph term owns under 2% of the update, does

Table 2: Student internals on held-out two-digit addition, one seed each. Without gold labels, GRAD reaches the specificity of label-trained SFT. *Specificity*: mean ANOVA margin of the top ten sum-value neurons (Section 3.3); *routing JSD*: mean Jensen–Shannon divergence between the student's and the teacher's routing profiles (lower is closer; membership noise 0.004–0.005).

teacher signal	untrained –	KD soft probabilities	GRAD + teacher's graph	SFT gold labels	teacher –
sum-value specificity	0.20	0.28	0.37	0.38	0.85
routing JSD to teacher	0.069	0.050	0.044–0.051	0.046	–

multiplication fall to 0.854 (Appendix B). With gradient clipping always binding, that share, not λ itself, is the quantity to tune. Supernode size matters more: three neurons per supernode is too thin a target and multiplication falls to the KD level, ten and thirty are equivalent, and the teacher's pre-selection fraction is immaterial.

The gap is not a compute gap. A GRAD step costs $14\times$ a standard-KD step with the teacher's graphs cached, but giving KD that compute does not close the gap: stretched to 280 steps it stays inside the seed spread of its 100-step runs on multiplication, and at $1{,}400$ steps it falls below them (Table 12, Appendix D).

The target's content matters. The scrambled control keeps the teacher's numbers and per-row sparsity but pairs every supernode with the wrong sources. It does not merely fail to help: two of its three seeds never exceed the untrained student in distribution, the third peaks at 0.885 on multiplication against the real target's 0.955, and scored against the real target its loss rises while the real arm's falls (Figure 8, Appendix B). Section 5 shows what that content does.

4.3 The same target on a larger student

A second student needs only its own pass over the probe grid. On Llama-3.2-3B-Instruct, under the identical protocol, λ and supernode size, the untrained student already reaches 0.812 on multiplication and KD takes it to 0.991, leaving no headroom there; GRAD's advantage moves to the operand-count families on which KD still trails the teacher: 0.912 ± 0.023 against 0.852 ± 0.031 on three-operand addition, above the teacher's 0.884, and 0.618 ± 0.037 against 0.503 ± 0.056 on four-operand addition, for a held-out mean of 0.875 ± 0.015 against 0.826 ± 0.021 (Table 9, Figure 7). The 1B setting is transported unchanged, and GRAD again reaches the training family's ceiling in about half the steps.

5 Understanding GRAD's Gains

Section 4 shows that GRAD generalizes better than its baselines and that the gain depends on the teacher's content. Here we ask how it changes the student: what moves inside the network, and what produces the held-out gain. We compare seed-matched students of the $\lambda = 1$ arm, whose standard-KD and SFT counterparts were kept: three seeds per arm, seed s sharing initialization and data order across arms, read at step 100 (GRAD reaches 0.949 ± 0.002 on multiplication, against 0.951 at $\lambda = 3$; Table 6).

Without gold labels, GRAD's internals reach label-trained quality. The teacher's graph shapes the student's representations beyond what its output distribution does (Table 2). The top-10 sum-value specificity of the student's neurons rises from 0.20 untrained to 0.28 under KD and to 0.37 under GRAD, level with SFT's 0.38, which fits the gold answers directly. Routing moves the same way: on held-out addition prompts the mean JSD to the teacher's routing profiles falls from 0.069 untrained to 0.050 under KD and 0.044 under GRAD on the same prompts, against 0.046 for SFT. Soft probabilities alone leave the internals short of what labels produce; the teacher's graph closes that gap without any. The teacher itself remains sharper still (0.85).

On multiplication, GRAD corrects a failure of standard KD: it commits to an answer. We split exact-match accuracy into *commitment*, whether the continuation begins with a number, and accuracy among committed answers (Figure 4; Table 14). The KD student answers only 46.6% of multiplication prompts, emitting the untrained model's instruction-style hedge on the rest, although it is right on 94.7% of those it answers; GRAD answers 99.9% and is right on 95.0%. The declined prompts are not the hard ones: on them GRAD is right 93.3% of the time (seed 1). The teacher

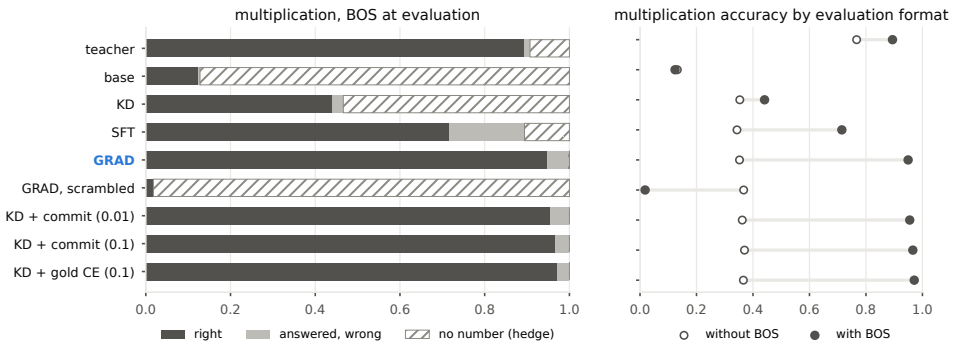

Figure 4: How GRAD improves multiplication. *Left:* greedy outputs with BOS at evaluation, split into right, answered but wrong, and no number (the hedge "`?\nTo find the answer`"); students are three-seed means. *Right:* accuracy without BOS at evaluation (open), the format the KD term trains on, against with BOS (filled); seed-matched KD and GRAD students differ by at most 0.005 without BOS, on every family.

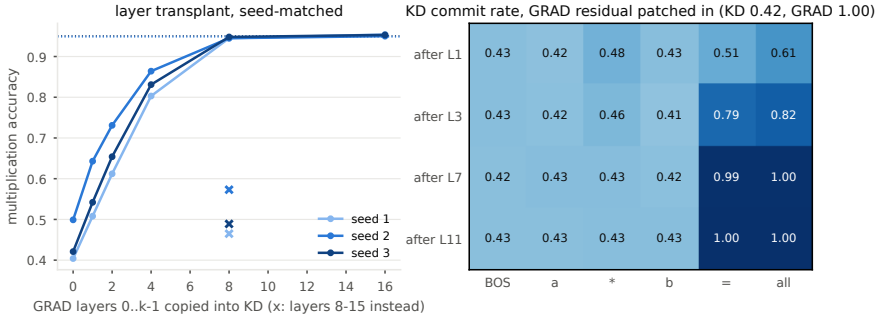

Figure 5: Where the correction is made. *Left:* copying GRAD's layers $0, \ldots, k-1$ into its seed-matched KD student and scoring multiplication with BOS; layers 0–7 reproduce the full gain in all three seeds, layers 8–15 (crosses) almost none. *Right:* overwriting the KD student's residual stream after layer L at one position with GRAD's (seed 1; seed 2 is alike); only the answer position carries the decision.

hedges too, on 9% of prompts, which is why GRAD can exceed it; the teacher is right on 98.6% of those it answers. On the addition families KD already commits on at least 98% of prompts, and GRAD's advantage there comes from answers that are more often right (0.473 against 0.438 among committed three-operand answers).

The correction lives in the evaluated input format. Evaluated without BOS, the format the KD term trains on (Section 4.1), seed-matched KD and GRAD students agree to within 0.005 on every family, while with BOS they differ by 0.45–0.55 on multiplication: the graph gradient changes the student's function in the one context in which it alone is computed.

It is made early, at the answer position. Copying GRAD's layers 0–7 into its seed-matched KD student reproduces the gain (multiplication 0.945, 0.945, 0.948 against GRAD's 0.949, 0.948, 0.951), while layers 8–15 give 0.47–0.57 against KD's 0.40–0.50 and no single block suffices (Figure 5), in the manner of model stitching (Lenc & Vedaldi, 2015; Bansal et al., 2021). Patching GRAD's residual stream into KD (Meng et al., 2022; Heimersheim & Nanda, 2024) at the = position after layer 7 raises KD's commitment from 0.42 to 0.99 (0.54 to 0.999 in seed 2); the same patch at BOS, an operand or the operator moves it by at most 0.08. The decision is computed at the final token by the first half of the network, the kind of targeted change fine-tuning often makes on top of existing mechanisms (Jain et al., 2024).

Targeting commitment directly gives the same held-out gain. If the multiplication gain is the decision to answer in the evaluated format, a loss that rewards answering there should reproduce

Table 3: Controls, with BOS at evaluation, step 100. *Commit* adds to KD a loss $-\log \sum_{n \in \mathcal{N}} p(n \mid \langle \text{bos} \rangle, x)$ at the answer position over the number tokens $\mathcal{N}$, computed on the training batch with BOS prepended; *gold CE* adds cross-entropy on the gold answer on the same batch. Weights in parentheses. KD, SFT and GRAD are the three-seed $\lambda = 1$ students of this section; the scrambled target is one seed at $\lambda = 1$ (Section 4.2 has three at $\lambda = 3$).

arm	two-digit	three-operand	four-operand	three-digit	multiplication
teacher	0.960	0.882	0.770	0.922	0.894
base	0.670	0.199	0.046	0.487	0.124
KD	0.984	0.429	0.114	0.585	0.441 ± 0.051
SFT	0.994	0.430	0.087	0.669 ± 0.012	0.715 ± 0.050
GRAD	0.999	0.473	0.128	0.656	0.949
GRAD, scrambled	0.158	0.000	0.001	0.033	0.018
KD + commit (0.01) (3 seeds)	0.999	0.522	0.140	0.679	0.951
KD + commit (0.1)	1.000	0.557	0.150	0.706	0.966
KD + gold CE (0.1)	1.000	0.579	0.166	0.756	0.971

it. We add to standard KD a *commitment* loss, $-\log$ of the total probability of number tokens at the answer position on the BOS-prefixed batch, which says nothing about which number is right (Table 3). At weight 0.01 it matches GRAD on every family (multiplication 0.951 ± 0.005 over three seeds), and at 0.1 it exceeds it; gold-answer cross-entropy in the same format does slightly better again. None changes the function without BOS. The scrambled target pushes the same decision the other way, putting 0.74 of the answer-position probability on the hedge: the teacher's content sets the direction of the decision, which is why it matters in Section 4.2, and λ and supernode size set its strength (Appendix B).

Where the effect holds, and why. The advantage needs the tokenization mismatch: with BOS in the training sequence, standard KD reaches 0.390 on multiplication and GRAD at $\lambda \in \{1, 3, 10\}$ reaches 0.386, 0.361 and 0.287 (Table 8); with no BOS anywhere the two tie (0.374 against 0.379, one seed each). With BOS, the teacher itself puts 0.255 of its answer-position probability on the hedge, against 0.004 without, and KD trained with BOS distills that hedge into the evaluated format, which the graph term at the weights tried does not outweigh; with no BOS, neither term is computed in that format; under BOS-free training only the graph term is. Gold labels in the evaluated format give the same correction: SFT with BOS reaches 0.929 on multiplication, against 0.725 under the shared protocol the KD baseline's sweep selected.

6 Conclusion

Attribution graphs can be a training signal. GRAD, which trains a student to route influence between functional supernodes as its teacher does, improves held-out generalization well beyond standard KD, carries to a larger student, and without gold labels gives the student internals of label-trained quality. Its largest held-out gain corrects standard KD's failure to commit to an answer in the evaluated format, so mechanism-matching objectives can act through where they are computed as well as what they match. Five cheap checks tell these apart: evaluate in every input format a loss sees, split accuracy into commitment and correctness, transplant layers between seed-matched runs, patch residual streams by position, and run a control that targets the suspected behavior directly.

Limitations and next steps. The study covers one training family and one teacher with two students of one model family, with only standard KD on the larger student; the largest gain relies on a nonstandard tokenization convention, for the reason Section 5 gives; the analysis uses the $\lambda = 1$ arm, which matches $\lambda = 3$ on multiplication, with one-seed internal measurements; and the supernode categories are written for the task, with membership and pre-selection non-differentiable. Next are GRAD under a consistent tokenization convention, the analysis on the larger student, and tasks beyond arithmetic with derived categories.

AI use statement

Generative AI (Claude) was used as a coding and writing assistant throughout: to implement and refactor the training, attribution, analysis and plotting code, to audit the input pipeline, to draft and revise the text of this paper from the authors' experiment logs, and to generate the tables and figures from the recorded histories. Every experiment was designed and run by the authors, every AI-written

line of code and text was reviewed by them, and every number in the paper is produced by a script from a recorded training history or analysis output rather than transcribed. The authors take full responsibility for the content.

ETHICS STATEMENT

The training and evaluation data are synthetic arithmetic prompts generated by a script; both models are public checkpoints. No human subjects, personal data or annotators were involved. The method adds a loss built from the teacher's attribution graphs to standard distillation and raises no concern beyond those of distillation generally.

REPRODUCIBILITY STATEMENT

Every training run records its full configuration, per-step losses and gradient metrics, and per-evaluation accuracies in a history file; the tables and figures are generated from those files by the scripts named in Appendix E, which also lists every hyperparameter (Table 13). Appendix A documents the adaptations needed to build a differentiable attribution graph inside a Hugging Face training loop for Llama, Appendix B the sweep that fixed the protocol, and Appendix F the procedures of the analysis in Section 5, whose scripts are released with the rest. Code and histories will be released.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations (ICLR)*, 2024.

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/methods.html.

Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Federico Barbero, Álvaro Arroyo, Xiangming Gu, Christos Perivolaropoulos, Michael Bronstein, Petar Veličković, and Razvan Pascanu. Why do LLMs attend to the first token? In *Conference on Language Modeling (COLM)*, 2025.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable LLM feature circuits. In *Advances in Neural Information Processing Systems*, 2024.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2:665–673, 2020. doi: 10.1038/s42256-020-00257-z.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching, 2024.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. NIPS 2014 Deep Learning Workshop.

Samyak Jain, Robert Kirk, Ekdeep Singh Lubana, Robert P. Dick, Hidenori Tanaka, Edward Grefenstette, Tim Rocktäschel, and David Scott Krueger. Mechanistically analyzing the effects of fine-tuning on procedurally defined tasks. In *International Conference on Learning Representations (ICLR)*, 2024.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4163–4174, 2020.

Adam Tauman Kalai, Ofir Nachum, Santosh S. Vempala, and Edwin Zhang. Why language models hallucinate, 2025.

Subhash Kantamneni and Max Tegmark. Language models use trigonometry to do addition, 2025. ICLR 2025 Workshop on Building Trust in Language Models and Applications.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529, 2019.

Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, 2022.

Neel Nanda. Attribution patching: Activation patching at industrial scale. https://www.neelnanda.io/mechanistic-interpretability/attribution-patching, 2023.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. Arithmetic without algorithms: Language models solve math with a bag of heuristics. In *International Conference on Learning Representations (ICLR)*, 2025.

Utkarsh Ojha, Yuheng Li, Anirudh Sundara Rajan, Yingyu Liang, and Yong Jae Lee. What knowledge gets distilled in knowledge distillation? In *Advances in Neural Information Processing Systems*, volume 36, pp. 11037–11048, 2023.

Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-tuning enhances existing mechanisms: A case study on entity tracking. In *International Conference on Learning Representations (ICLR)*, 2024.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *International Conference on Learning Representations (ICLR)*, 2015.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter, 2019. 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing (EMC^2), NeurIPS 2019.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *International Conference on Learning Representations (ICLR)*, 2024.

Noam Shazeer. GLU variants improve transformer, 2020.

Samuel Stanton, Pavel Izmailov, Polina Kirichenko, Alexander A. Alemi, and Andrew Gordon Wilson. Does knowledge distillation really work? In *Advances in Neural Information Processing Systems*, volume 34, pp. 6906–6919, 2021.

Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4323–4332, 2019. doi: 10.18653/v1/D19-1441.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 407–416, 2024. doi: 10.18653/v1/2024.blackboxnlp-1.25.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Somin Wadhwa, Silvio Amir, and Byron C. Wallace. Investigating mysteries of CoT-augmented distillation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6071–6086, 2024. doi: 10.18653/v1/2024.emnlp-main.349.

Somin Wadhwa, Silvio Amir, and Byron C. Wallace. Circuit distillation, 2025. Preprint; check for a published venue before camera-ready.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *International Conference on Learning Representations (ICLR)*, 2023.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5776–5788, 2020.

Xinpeng Wang, Bolei Ma, Chengzhi Hu, Leon Weber-Genzel, Paul Röttger, Frauke Kreuter, Dirk Hovy, and Barbara Plank. "My answer is C": First-token probabilities do not match text answers in instruction-tuned language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 7407–7416, 2024.

Zhengxuan Wu, Atticus Geiger, Joshua Rozner, Elisa Kreiss, Hanson Lu, Thomas Icard, Christopher Potts, and Noah Goodman. Causal distillation for language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 4288–4295, 2022. doi: 10.18653/v1/2022.naacl-main.318.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations (ICLR)*, 2024.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations (ICLR)*, 2017.

Tianyi Zhou, Deqing Fu, Vatsal Sharan, and Robin Jia. Pre-trained large language models use Fourier features to compute addition. In *Advances in Neural Information Processing Systems*, 2024.

A Attribution Graphs for Llama

The published construction assumes a transcoder replacement model and a hook-based forward pass (Ameisen et al., 2025). Building it for Llama inside a training loop, with the student's graph carrying gradients back into $\theta_{\mathcal{S}}$, required the following changes.

Supernode edge weights.

The supergraph's edges are the supernode adjacency of Ameisen et al. (2025),

$$W_{ts} \;=\; \text{supernode_adjacency}(t,s) \;=\; \frac{\displaystyle\sum_{n_t \in N_t} \text{frac_external}(n_t) \cdot \sum_{n_s \in N_s} A'_{n_t,n_s}}{\displaystyle\sum_{n_t \in N_t} \text{frac_external}(n_t)}, \tag{4}$$

where N_t and N_s are the member nodes of supernodes t and s, A' is the graph adjacency with each node's inbound weights normalized to sum to one over all sources (neurons, token embeddings and logits), and $\text{frac_external}(n_t)$ is the fraction of a member's absolute inbound weight that comes from outside its own supernode. Weighting members by frac_external keeps a supernode from dominating its own inbound profile, and normalizing by their sum makes the edge a weighted mean rather than a count of members (Ameisen et al., 2025). Because A' normalizes over all nodes while W spans only supernode columns, its rows do not sum to one, hence the ℓ_1 normalization in (1).

Gated MLPs. Llama's MLP is a SwiGLU block (Shazeer, 2020) with no single input matrix. With residual-stream input h at the node's position,

$$a_i = \sigma(g_i)\, u_i, \qquad v_i^{\text{in}} = \frac{\partial a_i}{\partial h} = \sigma(g_i)\, W_{\text{up}}[n_i,:] \;+\; u_i\, \sigma'(g_i)\, W_{\text{gate}}[n_i,:], \tag{5}$$

where $g_i = \langle W_{\text{gate}}[n_i,:], h\rangle$, $u_i = \langle W_{\text{up}}[n_i,:], h\rangle$ and σ is SiLU. The read direction is therefore a product-rule sum of two terms that itself depends on the activation, rather than a fixed row of W_{in}.

Differentiability. The pre-selection top-k is detached (indices are not differentiable), but activations and read/write vectors remain differentiable, so gradients flow $\mathcal{L}_{\text{graph}} \to \text{supernode_adjacency} \to A' \to A \to v^{\text{out}} \to W_{\text{down}}$ back into the model. Supernode *membership* is recomputed each step but treated as fixed structure; only the aggregation (4) is rebuilt differentiably.

Shared attribution targets. Both models are attributed to the *same* logit token set, the teacher's. Otherwise the two graphs would be built over different output nodes, and would be non-stationary early in training when the student's own output distribution is still wrong. The teacher's targets are computed once per prompt and cached (Appendix D).

Edges. Every edge (2) is the full Jacobian through attention patterns and normalization denominators alike; we do not apply the direct-path linearization of the published method, whose stop-gradients would otherwise have to be kept out of the forward pass that carries the student's gradient.

Precision. The graph gradient arrives one prompt at a time, since each prompt's graph is backpropagated and freed before the next is built (Appendix D), into a gradient buffer that already holds the far larger KD gradient. With bf16 master weights each of those $\sim$32 partial sums can fall below half a unit in the last place and be rounded away, and the loss grows as λ shrinks; fp32 master weights with a bf16 autocast forward retain the full gradient at every λ. The paper's runs use bf16 master weights, where both objectives do best; there the loss is about 2% of the graph gradient at the working weight and 30% at $\lambda = 0.3$, which is why λ was not pushed lower.

B Protocol Selection and Ablations

The optimizer protocol of Section 4.1 was fixed by sweeping the standard-KD baseline alone over learning rate, tokenization convention and the positions the KD term scores (Table 4), and every other arm inherited the winner. At 3×10^{-6} the baseline is still rising at step 100 (0.84 in-distribution); at 3×10^{-5} it reaches the training family but takes every held-out family to or below the untrained student; 10^{-5} is the usable rate. With BOS-free training, in which the training sequence carries no BOS while evaluation and attribution do, the baseline reaches 0.512 ± 0.117 on multiplication against 0.390 with the standard convention, at the same in-distribution accuracy, which is why BOS-free training was selected for every arm; Section 5 is the graph term under the standard one. Scoring the KD term on every position whose next token is real beats scoring only the answer positions under BOS-free training (0.989 against 0.976 in-distribution at the same multiplication accuracy), while under the standard convention one of the scored positions is BOS itself and the answer-only variant is the better baseline; Table 8 therefore carries both.

Table 4: The standard-KD protocol sweep on two-digit addition, one seed each except the shaded row. Each seed is read at its own best in-distribution step; multiplication and the mean over the four held-out families are read at that step. Italics are below the untrained student. The shaded row is the protocol of Section 4.1.

lr	BOS in training	KD positions	seeds	moved	2-digit add peak (step)	2×1-digit mult	held-out mean	2-digit add last
3×10^{-6}	no (BOS-free)	all	1	0.01	0.842 (100)	0.208	0.309	0.842
3×10^{-6}	no (BOS-free)	resp	1	0.01	0.810 (70)	0.158	0.288	0.808
3×10^{-6}	yes (standard)	all	1	0.01	0.907 (50)	0.142	0.258	0.840
3×10^{-6}	yes (standard)	resp	1	0.01	0.919 (50)	0.156	0.261	0.879
10^{-5}	no (BOS-free)	all	3	0.02	0.989 ± 0.003 (50/35/40)	0.512 ± 0.117	0.420 ± 0.044	0.985 ± 0.002
10^{-5}	no (BOS-free)	all	1	0.02	0.986 (30)	0.572	0.437	0.972
10^{-5}	no (BOS-free)	all	1	0.02	0.977 (100)	0.343	0.333	0.955
10^{-5}	no (BOS-free)	resp	1	0.02	0.976 (30)	0.378	0.375	0.950
10^{-5}	no, and none at evaluation	all	1	0.02	0.895 (45)	0.379	*0.192*	0.894
10^{-5}	yes (standard)	all	1	0.02	0.984 (50)	0.294	0.283	0.958
10^{-5}	yes (standard)	resp	1	0.02	0.982 (50)	0.390	0.302	0.963
3×10^{-5}	no (BOS-free)	all	1	0.04	0.978 (80)	0.184	*0.108*	0.974
3×10^{-5}	no (BOS-free)	resp	1	0.04	0.976 (90)	*0.045*	*0.065*	0.975
3×10^{-5}	yes (standard)	all	1	0.04	0.969 (50)	*0.003*	*0.064*	0.954
3×10^{-5}	yes (standard)	resp	1	0.04	0.968 (70)	*0.043*	*0.058*	0.956

Representation baselines. Table 5 sweeps the weight of the TinyBERT and CKA terms at each learning rate. Neither moves the baseline much at 10^{-5}: TinyBERT is best at 0.3 and degrades as the weight rises, and CKA is flat from 0.3 to 3. The three-seed arms of Table 1 use 0.3 for both. The one point that stands out is CKA at weight 10, which reaches 0.743 on multiplication in its single seed with lower three-operand addition. That point is a single seed, inside the ± 0.10 spread these arms show on multiplication, and the worst CKA setting on the selection split, so the rule did not choose it; even at face value it is below GRAD's 0.955 ± 0.007.

Graph-loss weight. Table 6 and Figure 6 sweep λ under the shared protocol. Because gradient clipping is always binding (Appendix C), λ acts only through the share of the update the graph term owns, the *ratio* column of Table 11: 0.015 at $\lambda = 0.3$, 0.05 at 1, 0.12 at 3 and 0.22 at 6. The held-out results are flat from 1 to 6 and fall only at 0.3, and $\lambda = 3$ was chosen as the middle of the flat range. The $\lambda = 10$ run stopped at step 40 rather than 100; its peak, at step 20 like the other weights, falls in the same flat range (0.955 on multiplication, 0.506 on three-operand addition, 0.676 on three-digit addition, gradient ratio 0.24), so the range is flat from 1 to 10 and only 0.3 is too small.

Supernode size and pre-selection. Table 7 varies the neurons per supernode, $n_{\text{per}} \in \{3, 10, 30\}$, against the fraction of teacher neurons kept per layer before attribution, $\{0.003, 0.03, 0.1\}$, at $\lambda = 3$ for 50 steps. Three neurons per supernode is too few: multiplication falls to 0.46–0.74, the baseline's range, and in-distribution accuracy drops. At ten and thirty the result is stable and the pre-selection fraction does not matter, so the teacher's graph can be built from a thirtieth of its neurons at no cost.

The standard BOS convention. Table 8 is the ablation of Section 5: with BOS in the training sequence, the KD baseline with either position set against GRAD at three weights with either position set, one seed each. The negative rests on the six graph configurations agreeing rather than on seeds, and on the baseline's own six; no arm here is replicated.

Table 5: TinyBERT and CKA weight sweeps, one seed each, laid out as Table 4. Shaded rows are the weights used in Table 1.

recipe	lr	weight	seeds	2-digit add peak (step)	2×1-digit mult	3-operand add	held-out mean	2-digit add last
TinyBERT	3×10^{-6}	0.3	1	0.842 (60)	0.204	0.374	0.305	0.839
TinyBERT	3×10^{-6}	1	1	0.841 (60)	0.200	0.376	0.305	0.838
TinyBERT	3×10^{-6}	3	1	0.831 (60)	0.194	0.375	0.301	0.828
TinyBERT	3×10^{-6}	10	1	0.810 (60)	0.163	0.372	0.288	0.798
TinyBERT	10^{-5}	0.3	1	0.983 (30)	0.521	0.444	0.422	0.973
TinyBERT	10^{-5}	0.3	1	0.980 (15)	0.288	0.312	0.326	0.945
TinyBERT	10^{-5}	0.3	1	0.973 (15)	0.399	0.282	0.353	0.958
TinyBERT	10^{-5}	1	1	0.983 (30)	0.509	0.447	0.420	0.965
TinyBERT	10^{-5}	3	1	0.978 (30)	0.438	0.423	0.393	0.951
TinyBERT	10^{-5}	10	1	0.965 (60)	0.389	0.376	0.349	0.964
TinyBERT	3×10^{-5}	0.3	1	0.973 (70)	*0.114*	*0.031*	*0.085*	0.969
TinyBERT	3×10^{-5}	1	1	0.977 (80)	0.142	*0.034*	*0.090*	0.970
TinyBERT	3×10^{-5}	3	1	0.986 (50)	0.134	*0.030*	*0.102*	0.982
TinyBERT	3×10^{-5}	10	1	0.984 (70)	0.293	*0.036*	*0.140*	0.980
CKA	3×10^{-6}	0.3	1	0.843 (60)	0.204	0.373	0.305	0.842
CKA	3×10^{-6}	1	1	0.840 (60)	0.201	0.375	0.302	0.834
CKA	3×10^{-6}	3	1	0.833 (60)	0.189	0.367	0.294	0.828
CKA	3×10^{-6}	10	1	0.823 (60)	0.186	0.369	0.289	0.819
CKA	10^{-5}	0.3	1	0.985 (30)	0.542	0.456	0.432	0.972
CKA	10^{-5}	0.3	1	0.974 (15)	0.280	0.298	0.317	0.949
CKA	10^{-5}	0.3	1	0.965 (50)	0.371	*0.195*	0.267	0.956
CKA	10^{-5}	1	1	0.983 (30)	0.567	0.452	0.434	0.969
CKA	10^{-5}	3	1	0.985 (30)	0.578	0.446	0.429	0.974
CKA	10^{-5}	10	1	0.986 (30)	0.743	0.433	0.465	0.979
CKA	3×10^{-5}	0.3	1	0.970 (70)	0.229	*0.036*	*0.113*	0.968
CKA	3×10^{-5}	1	1	0.969 (20)	0.412	*0.098*	0.226	0.943
CKA	3×10^{-5}	3	1	0.970 (10)	0.742	0.259	0.369	0.964
CKA	3×10^{-5}	10	1	0.971 (100)	0.396	*0.049*	*0.176*	0.971

Table 6: Graph-loss weight sweep, one seed each. *Steps run* is the last evaluated step; each seed is read at its own best in-distribution step up to it.

λ	seeds	steps run	2-digit add peak (step)	3-operand add	4-operand add	3-digit add	2×1-digit mult	held-out mean	peak OOD (step)
untrained student	—	—	0.670	0.199	0.045	0.487	0.124	0.214	—
teacher	—	—	0.960	0.884	0.769	0.922	0.894	0.868	—
0 (standard KD)	3	100	0.989 ± 0.003 (50/35/40)	0.440 ± 0.027	0.120 ± 0.010	0.608 ± 0.026	0.512 ± 0.117	0.420 ± 0.044	0.431 ± 0.025 (30)
0.3	1	90	0.999 (70)	0.472	0.128	0.650	0.854	0.526	0.544 (30)
1	1	100	1.000 (60)	0.482	0.128	0.653	0.950	0.553	0.565 (20)
3	1	100	1.000 (20)	0.541	0.152	0.686	0.951	0.583	0.583 (20)
6	1	100	1.000 (20)	0.532	0.153	0.681	0.956	0.581	0.581 (20)
10	1	10	0.991 (10)	0.532	0.161	0.668	0.944	0.576	0.576 (10)

The second student. Table 10 is the standard-KD learning-rate sweep on Llama-3.2-3B-Instruct that Section 4.3 inherits, with BOS-free training and every real position scored as in the 1B protocol: at 3×10^{-6} the baseline is still rising at step 100 (0.80 in-distribution), at 3×10^{-5} it reaches the training family by step 15 and then takes the operand-count families below the untrained student, and 10^{-5} is again the usable rate, so the 3B arms run under the shared protocol unchanged. Figure 7 draws the two 3B arms against training step.

C Gradient Dynamics

Every graph run records, per step, the norms of the KD and graph gradients (the latter already scaled by λ), their ratio, their cosine, and the pre-clip norm of the total gradient; Table 11 gives the run means for every arm.

The graph gradient is near-orthogonal to the KD gradient. In every configuration, under both tokenization conventions and at every λ, the mean cosine lies between -0.001 and $+0.027$ and is negative on 40–53% of steps. The gain is therefore not an alignment effect in which the auxiliary term points the same way as KD and speeds it up. At $\lambda = 3$ the graph term owns about 0.12 of the update on average, rising from 0.07 at step 1 to 0.2 as the KD gradient shrinks tenfold over the run, and under AdamW that is enough to own the update in the coordinates the KD gradient leaves alone.

Only the ratio of the weights matters. The pre-clip gradient norm stays between 4.8 and 80 in every arm, against a clip threshold of 1.0, so every step of every run is rescaled to a unit-norm update. Hence only the ratio of λ to the KD weight acts, the logged gradient ratio is directly the share of the

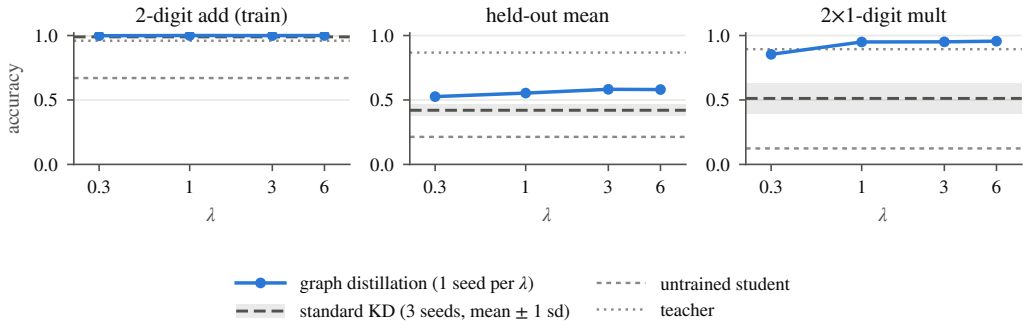

Figure 6: Accuracy against λ on a log axis, each run at its own best in-distribution step: in-distribution, mean over the held-out families, and multiplication. The dashed rule and band are the three-seed standard-KD baseline; the dotted rules are the untrained student and the teacher.

Table 7: Neurons per supernode against the teacher's pre-selection fraction, $\lambda = 3$, 50 steps, one seed each; the KD row is read up to step 50.

n_{per}	teacher top-k	2-digit add peak (step)	3-operand add	4-operand add	3-digit add	2×1-digit mult	held-out mean	peak OOD (step)
untrained student	—	0.670	0.199	0.045	0.487	0.124	0.214	—
teacher	—	0.960	0.884	0.769	0.922	0.894	0.868	—
— (standard KD, 100 steps)	—	0.989 ± 0.003 (50/35/40)	0.440 ± 0.027	0.120 ± 0.010	0.608 ± 0.026	0.512 ± 0.117	0.420 ± 0.044	0.431 ± 0.025 (30)
3	0.003	0.998 (20)	0.500	0.137	0.643	0.737	0.504	0.507 (30)
3	0.03	0.982 (10)	0.461	0.117	0.603	0.462	0.411	0.544 (20)
3	0.1	0.981 (10)	0.458	0.114	0.602	0.457	0.408	0.544 (20)
10	0.003	1.000 (30)	0.484	0.129	0.667	0.953	0.558	0.569 (20)
10	0.03	1.000 (20)	0.546	0.159	0.690	0.948	0.586	0.586 (20)
10	0.1	1.000 (30)	0.512	0.138	0.675	0.960	0.572	0.585 (20)
30	0.1	1.000 (30)	0.534	0.157	0.684	0.965	0.585	0.587 (20)

update the graph term owns and is the quantity to tune against, and $\lambda_{\text{KD}} = 0$ is the one point scaling λ cannot reach.

D Compute Methods

Four implementation choices set the cost of a graph-distillation step in Table 12.

Teacher cache. The teacher's supernode adjacency for a prompt depends only on the prompt and the graph parameters, so it is built once per prompt and cached to disk under a key of those; every run in this paper (all seeds, every λ, the scrambled control) shares one cache and pays the warm cost. Filling the cache for the 3200 prompts of a 100-step run costs 1.1×10^{17} FLOPs once, about 17 warm graph runs or 24,000 KD steps; on the fly the teacher's graph takes 84 s per step against 0.1 s cached.

Analytic pre-selection. Write-vector norms factor as $\|v_i^{\text{out}}\|_2 = |a_i| \, \|W_{\text{down}}[:, n_i]\|_2$, so the top-$k$ of Section 3.2 runs without materializing the $[n_{\text{pos}}, d_{\text{mlp}}, d]$ tensor of write vectors.

Batching. Edges are computed with one batch-expanded forward pass per batch of target nodes, and the ANOVA scores of (3) with one matmul of every mask against every activation grid.

Bounded memory. Each prompt's graph loss is backpropagated and freed before the next prompt's graph is built, so peak memory is that of one graph rather than a batch of them.

Untried levers. Two choices would reduce the graph arm's cost further: attributing a subset of each batch rather than all 32 prompts, which scales the student-side cost proportionally, and the teacher's pre-selection fraction, which Table 7 shows can drop from 0.1 to 0.003 at no cost in accuracy.

E Training and Evaluation Details

Evaluation. An answer is scored by decoding only the generated continuation and parsing its leading integer. This matters: Llama 3 tokenizes digits in groups of one to three, and the untrained student often emits an answer digit by digit, so a one-token budget truncates it, while parsing the prompt echo lets a second, echoed equation override a correct answer. Both effects move every family's accuracy, the untrained student's included, so every number in the paper uses this one convention.

Table 8: The standard BOS convention: standard KD against GRAD, one seed each, laid out as Table 6.

arm	KD positions	λ	2-digit add peak (step)	3-operand add	4-operand add	3-digit add	2×1-digit mult	held-out mean	peak OOD (step)
untrained student	—	—	0.670	0.199	0.045	0.487	0.124	0.214	—
teacher	—	—	0.960	0.884	0.769	0.922	0.894	0.868	—
standard KD	all	—	0.984 (50)	0.245	0.073	0.520	0.294	0.283	0.289 (60)
standard KD	resp	—	0.982 (50)	0.283	0.071	*0.466*	0.390	0.302	0.302 (50)
GRAD	resp	1	0.981 (50)	0.272	0.070	*0.463*	0.386	0.297	0.297 (50)
GRAD	resp	3	0.982 (50)	0.264	0.069	*0.453*	0.361	0.286	0.286 (50)
GRAD	resp	10	0.975 (60)	0.235	0.054	*0.432*	0.287	0.252	0.300 (10)
GRAD	all-no-bos	1	0.969 (60)	0.236	0.065	*0.479*	0.294	0.269	0.298 (10)
GRAD	all-no-bos	3	0.966 (60)	0.238	0.062	0.490	0.284	0.269	0.303 (10)
GRAD	all-no-bos	10	0.960 (60)	0.204	0.049	0.496	0.224	0.243	0.310 (10)

Table 9: Llama-3.2-3B-Instruct as the student under the protocol, λ and supernodes of Table 1, nothing retuned; three seeds per arm, laid out as Table 1.

	untrained student	teacher	standard KD (3 seeds)	graph distillation ($\lambda = 3$, 3 seeds)
2-digit add (train)	0.618	0.960	0.991 ± 0.002	$\mathbf{1.000 \pm 0.000}$
selected step, per seed	—	—	45/55/40	25/20/15
3-operand add	0.747	0.884	0.852 ± 0.031	$\mathbf{0.912 \pm 0.023}$
4-operand add	0.559	0.769	$\mathit{0.503 \pm 0.056}$	$\mathbf{0.618 \pm 0.037}$
3-digit add	0.799	0.922	0.959 ± 0.006	$\mathbf{0.975 \pm 0.007}$
2×1-digit mult	0.812	0.894	0.991 ± 0.001	$\mathbf{0.996 \pm 0.002}$
held-out mean	0.729	0.868	0.826 ± 0.021	$\mathbf{0.875 \pm 0.015}$
peak held-out mean (step)	—	—	0.853 ± 0.011 (35)	$\mathbf{0.895 \pm 0.006}$ (35)
2-digit add (train), last step	—	—	0.987 ± 0.002	$\mathbf{0.997 \pm 0.002}$

Held-out families. Three- and four-operand addition are sums of two-digit numbers, three-digit addition is the only family whose answers need two tokens, and multiplication is a two-digit by one-digit product. The current suite has two- and four-operand sums only, so the operand-count boundary cannot be tested from the three-operand side; a three-operand three-digit family is an open item. In the released code and histories the families are `22_add`, `222_add`, `2222_add`, `33_add` and `21_mult`.

F Analysis Details

Checkpoints. The standard-KD, SFT and GRAD ($\lambda = 1$) students of Section 5, seeds 1–3, at step 100, under the protocol of Table 13. Seed s shares initialization and data order across the three arms, and the step-1 KD loss agrees across them to four decimals.

Commitment and correctness. Greedy decoding with the evaluation settings of Table 13. A continuation is *committed* if it begins with an integer (a leading space allowed, as the evaluator allows it). *Forced* accuracy is whether the most likely number token at the answer position is the gold answer's first token; every answer in multiplication, three-operand and four-operand addition is one Llama-3 token. $P(\texttt{?})$ is the next-token probability of tokens beginning with `?`. The evaluation without BOS uses the same prompts and decoding with no special tokens.

Layer transplant. All parameters of decoder layers in a span (attention, MLP and both norms) are copied from the GRAD student into its seed-matched KD student, which is then scored as above.

Residual patching. Every multiplication prompt tokenizes to $[\langle\mathrm{bos}\rangle, a, \star, b, =]$. We run the KD student and overwrite the output of decoder layer L at one position with the GRAD student's on the same prompt; *commit* is then whether the most likely next token is a number or whitespace.

Controls. Identical to the standard-KD arm (seed 1 unless stated) with one added term, computed on the training batch with BOS prepended: *commit*, $-\frac{1}{B}\sum_b \log \sum_{n \in \mathcal{N}} p_\theta(n \mid \langle\mathrm{bos}\rangle, x_b)$ at the position predicting the first answer token, with $\mathcal{N}$ every vocabulary item that decodes to an optional space followed by digits; or *gold CE*, the cross-entropy of the answer tokens.

Internals. Specificity and routing JSD (Table 2) are computed on held-out two-digit addition prompts with the supernode construction of Section 3.3; membership noise is the distance between two draws of supernode membership for the same model. Patching a carrying prompt's activations into a non-

Table 10: The standard-KD learning-rate sweep on the 3B student, each seed at its own best in-distribution step; the shaded row is the shared protocol. Italics are below the untrained 3B student.

lr	seeds	2-digit add peak (step)	3-operand add	4-operand add	3-digit add	2×1-digit mult	held-out mean	2-digit add last
3×10^{-6}	1	0.803 (80)	0.811	*0.544*	0.886	0.891	0.783	0.800
10^{-5}	3	0.991 ± 0.002 (45/55/40)	0.852 ± 0.031	*0.503 ± 0.056*	0.959 ± 0.006	0.991 ± 0.001	0.826 ± 0.021	0.987 ± 0.002
3×10^{-5}	1	0.974 (15)	*0.655*	*0.262*	0.873	0.998	*0.697*	0.946

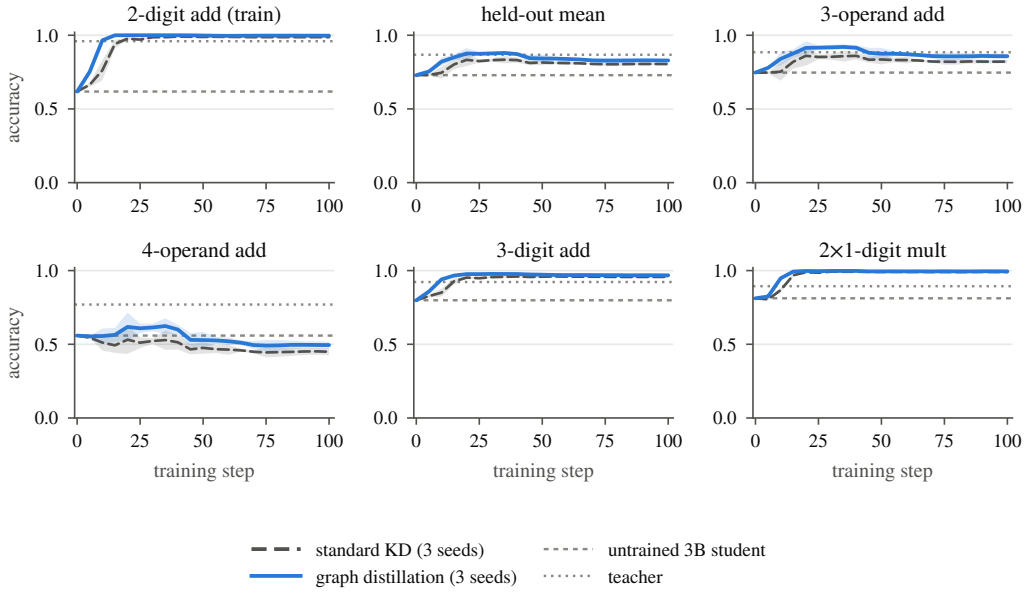

Figure 7: The 3B student: accuracy against training step for standard KD and graph distillation, in-distribution, the mean over the held-out families, and each family. Seed means with a ± 1 sd band over three seeds. The dashed rule is the untrained 3B student and the dotted rule the teacher.

carrying one recovers 0.48 of the answer shift through the teacher's operand ones-digit supernodes and 0.23–0.28 in every student, against at most 0.02 for matched random neurons.

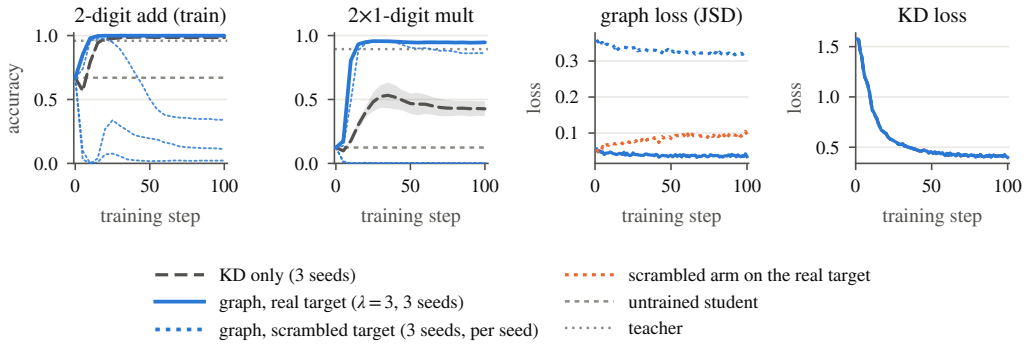

Figure 8: Real against scrambled routing target at $\lambda = 3$, with the KD-only baseline. *Left pair:* in-distribution and multiplication accuracy. *Third:* the graph loss the real arm trains on; the scrambled arm's loss on its own target and on the real target is drawn here. *Right:* the KD loss of the arms.

Table 11: Run means of the gradient metrics for every graph arm. *Ratio* is $\lambda\|g_{\text{graph}}\|/\|g_{\text{KD}}\|$; the cosine column gives the signed mean and, in parentheses, the mean absolute value; *neg* is the fraction of steps with a negative cosine; *pre-clip* is the range of the total gradient norm before clipping at 1.0. The standard-BOS rows are the answer-position variants of Table 8.

arm	$\|g_{\text{KD}}\|$	$\lambda\|g_{\text{graph}}\|$	ratio	cos (mean | cos |)	neg	pre-clip
BOS-free, $\lambda = 3$	14.8	1.34	0.118	+0.0004 (0.012)	0.48	5.3–72.8
0.02						
BOS-free, $\lambda = 3$, scrambled	14.8	2.01	0.185	-0.0036 (0.010)	0.55	5.8–73.8
0.02						
BOS-free, $\lambda = 0.3$	15.4	0.17	0.015	-0.0005 (0.010)	0.49	5.2–79.5
0.02						
BOS-free, $\lambda = 1$	14.5	0.51	0.046	+0.0011 (0.015)	0.46	4.8–79.5
0.02						
BOS-free, $\lambda = 6$	14.6	2.46	0.218	-0.0009 (0.016)	0.53	5.2–79.5
0.02						
BOS-free, $\lambda = 10$	52.0	10.20	0.188	+0.0008 (0.039)	0.40	32.2–80.5
0.02						
standard BOS, $\lambda = 1$	15.9	0.43	0.047	+0.0199 (0.056)	0.41	3.8–130.0
0.02						
standard BOS, $\lambda = 3$	15.7	1.26	0.131	+0.0262 (0.090)	0.46	3.6–127.0
0.02						
standard BOS, $\lambda = 10$	16.1	4.00	0.380	+0.0267 (0.087)	0.53	4.8–125.5
0.02						

Table 12: Cost and accuracy on one axis. FLOPs cover every matmul, attention and convolution kernel of a training step of 32 prompts, teacher forward and both attribution graphs included; *PFLOPs spent* is that per-step cost times the steps taken, and rows are paired on it – GRAD at its held-out peak (step 20) against standard KD stretched to 280 steps, at step 100 against KD at 1,400. Accuracy is each seed at its own best in-distribution step within the budget, those steps in parentheses. The cold-cache row is the row above it with the teacher's graphs rebuilt each step rather than read from cache: a cost, not another training.

arm	steps	TFLOPs / step	× KD	PFLOPs spent	seeds	2-digit add (step)	2×1-digit mult	held-out mean
standard KD	100	4.6	1.0	0.46	3	0.989 ± 0.003 (50/35/40)	0.512 ± 0.117	0.420 ± 0.044
GRAD, warm cache	20	63.6	13.9	1.27	3	0.999 ± 0.001 (20/20/20)	0.949 ± 0.003	0.586 ± 0.004
standard KD, stretched schedule	280	4.6	1.0	1.28	1	0.986 (30)	0.572	0.437
GRAD, warm cache	100	63.6	13.9	6.36	3	0.999 ± 0.001 (70/35/30)	0.955 ± 0.007	0.551 ± 0.015
standard KD, stretched schedule	1400	4.6	1.0	6.41	1	0.977 (100)	0.343	0.333
GRAD, cold cache	100	1144.6	250	114	—	—	—	—

Table 13: Configuration shared by every run in the paper unless a section says otherwise. The three trainers (SFT, standard KD, GRAD) share one implementation of the optimizer, precision regime, data order, tokenization and evaluation.

teacher / student	Meta-Llama-3-8B-Instruct / Llama-3.2-1B-Instruct, 1.24B parameters (Section 4.3: Llama-3.2-3B-Instruct, 3.21B parameters, same protocol)
training data	two-digit addition prompts "a+b=", batch 32, 100 steps
optimizer	AdamW, peak learning rate 10^{-5}, 10 warmup steps, cosine decay to 10^{-6}, gradient clipping at 1.0
precision	bf16 master weights, gradients and optimizer moments; bf16 autocast forward; teacher bf16
tokenization	BOS-free training: no BOS in the training sequence, BOS in evaluation and attribution
KD term	$\tau = 1$, forward KL over every position whose next token is real, weight 1
graph term	$\lambda = 3$, JSD, the six functional supernodes of Section 3.3, 10 neurons each
attribution	top 10% of neurons per layer by write norm in the teacher, 1% in the student; logit targets: top tokens covering 0.95 of $\mathrm{softmax}(z/\tau)$, at most ten; full-Jacobian edges
evaluation	greedy decoding, up to 5 new tokens, leading integer of the continuation; every 5 steps (ablation runs: 10) on the training family and the four held-out families
test sets	a fixed 2000-prompt subset of each family's test split, disjoint from training
baselines	TinyBERT recipe (MSE on projected hidden states and on attention patterns, uniform layer map, projection initialized by least squares) and linear CKA on residual streams; weight 0.3
seeds	1, 2, 3 for every arm; 1 for the ablations
hardware	one NVIDIA A100 80 GB

Table 14: Multiplication split into commitment and correctness, BOS at evaluation, step 100. Students are three-seed means ($\pm$ sd where above 0.01); controls are one seed. The last column evaluates the same checkpoints without BOS.

arm	acc	commit	acc|commit	forced	$P(?)$	acc, no BOS
teacher	0.894	0.907	0.986	0.973	0.187	0.767
base	0.124	0.128	0.969	0.340	0.614	0.132
KD	0.441 ± 0.051	0.466 ± 0.060	0.947 ± 0.014	0.542 ± 0.036	0.260 ± 0.023	0.353
SFT	0.715 ± 0.050	0.894 ± 0.050	0.800 ± 0.039	0.514 ± 0.040	0.079 ± 0.029	0.343 ± 0.029
GRAD	0.949	0.999	0.950	0.859	0.062	0.352 ± 0.010
GRAD, scrambled	0.018	0.018	1.000	0.192	0.742	0.367
KD + commit (0.01)	0.955	1.000	0.955	0.921	0.019	0.362
KD + commit (0.1)	0.966	1.000	0.966	0.965	0.001	0.370
KD + gold CE (0.1)	0.971	1.000	0.971	0.966	0.002	0.366