Painting Arithmetic: A Rational-Form Network for Visual Symbolic Computation in Latent Space

Taha Bouhsine

May 19, 2026

Abstract

Visual arithmetic on MNIST is normally framed as classification: two digit images and an operator token map to one of ~ 91 result classes. We instead treat both operands *and the operator* as 28×28 images, route all three through a single CNN encoder, and replace the softmax head with a small ConvTranspose decoder that paints the answer as a 28×84 image with three slots [*sign* | *tens* | *units*]. Between the encoder and the decoder we place a single layer of the Yāt rational function $y_u(x) = \alpha(x\cdot W_u + b)^2/(\|x - W_u\|^2 + \varepsilon)$, which we show makes each row W_u act as a soft prototype peak in input space (Proposition 1). The model has 0.81M parameters total and reaches 96.91% nearest-neighbour OCR accuracy on the test set after 25 epochs of joint training ($\sim$13 minutes on Apple-Silicon MPS), including 97.1% on multiplication and 96.0%–98.3% across all four operators. The same architecture trained under a three-phase frozen-component curriculum (Section 3) sacrifices ~ 8 pp OCR for cleaner component-wise attributability and provides the basis for the interpretability and intervention experiments below.

We test the "geometric calculator" claim of Goodfire AI [2025a]. For every pair of operators, the top singular value of trunk-vector differences across the 10×10 (a, b) grid captures 44–62% of the variance; we recast this as a rank-1 + residual hypothesis (Definition 1) with measured residual budget $\eta^\star_{ij} \le 0.56$, three orders of magnitude smaller than the Llama analysis. A steering experiment on the corresponding direction flips 20%–70% of painted answers between operator pairs, turning the descriptive claim into a causal one.

Because we have a trained decoder, we can also push Yāt prototype rows back through it; the resulting unit atlas reveals slot-localised *painter* neurons, and a Jacobian decomposition of the decoder (Eq. 4) shows that operator-tagged units divide their output energy by output slot asymmetrically. Finally, we use the prototype labels to drive a causal experiment. We cast three weight-space interventions (ROW, SLOT, RANDOM) as idempotent projections on the trunk's parameter space (Definition 2). Zeroing the 30 Yāt units whose middle-slot prototype is $\times$ drops $\times$ OCR from 86.5% to 42.7% ($\Delta = -43.7$ pp) while $\div$ loses only 1.6 pp; zeroing *only* the middle 64-d slot of those rows reproduces -40.0 pp of the same drop, evidencing that the operator computation is read out through the prototype's operator slot rather than laundered through digit pathways. Specificity scores (Definition 3) of +33.8 pp ($\times$) and +10.5 pp (+) exceed the random-ablation baseline of +2.1 pp and +1.8 pp. The result is the first surgical, prototype-targeted operator knock-out on a Yāt-style architecture, and supplements the geometric calculator direction with a circuit-level mechanism.

1 Introduction

A neural network learning 7×6 from two MNIST images and an operator symbol has to do two things at once: read pixels, and compute. Existing demonstrations of this task collapse both into a K-way softmax over result classes [Trask et al., 2018, Saxton et al., 2019], which makes the model's internal representation of *value* invisible. Larger models doing arithmetic in language space have

recently been shown to represent integers as points on multiple circles in their residual stream, with each arithmetic operation acting as a direction in that space [Nanda et al., 2023, Goodfire AI, 2025a]. A natural question is whether a model small enough to fit on a laptop shows the same geometric structure, and whether that structure can be *seen* directly rather than inferred from probes.

This paper proposes a tiny network for visual symbolic arithmetic and uses the structure of its output to read what its hidden layers have learned. We treat the operator as another image, share a CNN encoder across all three input slots, and replace the classifier head with an upsampling decoder that emits a 28×84 result image (Figure 1). The decoder's role is double: it produces the user-facing answer, and it serves as a fixed lens through which we can decode latent directions into interpretable pixels. One or two layers of the Yā*t* rational function sit between the encoder and the decoder. Each row of a Yā*t* layer's weight matrix is, formally, a soft prototype peak in input space (Proposition 1), so the same machinery used in prototype networks [Snell et al., 2017] applies inside the trunk for free.

We make four contributions. **First**, we show that a 0.81 M-parameter network can do single-digit visual arithmetic from image inputs to image outputs at 96.91% accuracy (Section 4). The key training recipe is a pair of training-only auxiliary heads: per-slot cross-entropy on $(\text{sign}, \text{tens}, \text{units})$ and modular cross-entropy on $(\bmod 2, \bmod 5, \bmod 11)$. Removing the per-slot head alone drops OCR by 34 points and produces the multi-digit failure mode where the decoder defaults to leaving the tens column blank (Section 4.2). **Second**, we reproduce the operator-as-direction finding of Goodfire AI [2025a] in this small setting, casting it as the formal rank-1 + residual hypothesis of Definition 1 and verifying $\eta^{\star}_{ij} \in [0.38, 0.56]$ across all six operator pairs (Section 5). **Third**, pushing one-hot vectors $\alpha\, e_u$ through the trained decoder yields a per-unit "footprint" image; many trunk units are slot-specialised, providing a generative version of activation atlases [Olah et al., 2017] for rational-form networks (Section 6). **Fourth**, we cast three weight-space interventions as idempotent projections on the trunk's parameter space (Definition 2–3) and use them to localise an operator-specific circuit in h_1 (Section 7).

2 Architecture

Encoder. The encoder is shared across all three input slots. It is a four-layer CNN ($32, 32, 64, 64$ channels with stride-2 downsampling between blocks), adaptive-average-pooled to 1×1, then linearly mapped to a 64-dim embedding. Sharing the encoder forces digits and operators to inhabit a single latent space.

Yā*t* trunk. Each Yā*t* layer computes, for $u = 1, \ldots, m$,

$$y_u(x) \;=\; \alpha_u \, \frac{(\, x \cdot W_u + b_u \,)^2}{\|x - W_u\|^2 + \varepsilon}, \tag{1}$$

where $W_u \in \mathbb{R}^d$ is the row of the weight matrix corresponding to output unit u, and $b_u, \alpha_u \in \mathbb{R}$ are learnable scalars. The denominator regulariser $\varepsilon > 0$ is reparameterised through softplus to remain positive.

Proposition 1 (Yā*t* units are soft prototype peaks)**.** Let $y_u : \mathbb{R}^d \to \mathbb{R}$ be defined by Eq. (1) with $\alpha_u > 0$, $b_u \geq 0$, $\varepsilon > 0$, $W_u \neq 0$.

(i) For fixed $\|x\|$, $y_u(x)$ is strictly increasing in $\langle x, W_u \rangle$.

(ii) Along the ray $r \mapsto y_u(r\hat{W}_u)$ with $\hat{W}_u = W_u / \|W_u\|$, the function attains a unique maximum at $r^{\star} = \|W_u\| + O(\varepsilon / \|W_u\|^3)$.

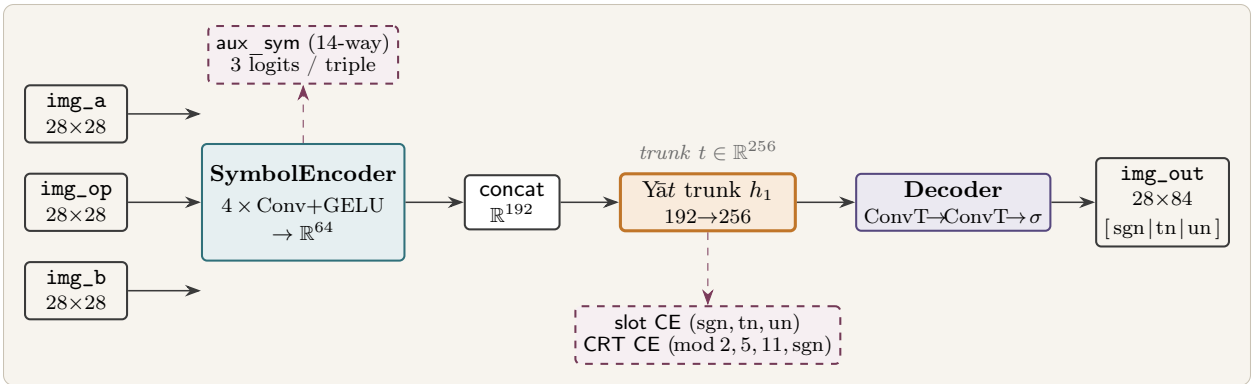


Figure 1: Architecture. Three image inputs share one CNN encoder `SymbolEncoder` that emits a $\mathbb{R}^{64}$ embedding per slot. The concatenated $\mathbb{R}^{192}$ vector passes through a single Yā*t* rational layer h ($192 \to 256$) and then a small ConvTranspose decoder that paints the 28×84 result image with slot layout [sign | tens | units]. *Wine dashed* boxes are training-only auxiliary heads: a 14-way symbol classifier on each encoder embedding, and modular $(\bmod 2, \bmod 5, \bmod 11, \text{sgn})$ plus per-slot $(\text{sgn}, \text{tn}, \text{un})$ classifiers attached to the trunk vector t. They are not part of the deployed ONNX graph.

(iii) In particular, $y_u(W_u) = \alpha_u(\|W_u\|^2 + b_u)^2/\varepsilon$ is a local maximum of y_u over the ball $\|x - W_u\| < \delta$ for δ small enough.

Proof sketch. (i) The numerator depends on x only through $\langle x, W_u\rangle$; the denominator $\|x - W_u\|^2 = \|x\|^2 - 2\langle x, W_u\rangle + \|W_u\|^2$ is decreasing in $\langle x, W_u\rangle$ at fixed $\|x\|$; the ratio f/g with $f \uparrow$ and $g \downarrow$ is therefore $\uparrow$. (ii) Writing $g(r) = \alpha_u(r\|W_u\|^2 + b_u)^2/((r - \|W_u\|)^2 + \varepsilon)$ and differentiating gives a single root in r, of the stated form by expanding in ε. (iii) follows from (i)–(ii). □

We adopt "Yā*t* rational function" in preference to "Yā*t* kernel" in the body of the paper. The function $y_u(x, W_u)$ is not symmetric in its two arguments and does not satisfy Mercer's condition; "rational" refers to its f/g functional form, not to a reproducing-kernel construction. Proposition 1 is the formal sense in which W_u behaves as a prototype: it is a peak of the activation surface in input space.

The trunk is a single such layer h_1 with d=192, m=256 ($\sim$ 50k parameters). We write $t := h_1(\text{FUSED}) \in \mathbb{R}^{256}$ for the trunk vector; all trunk-side auxiliary heads attach to t.

Decoder. The decoder is intentionally small: a linear map $\mathbb{R}^{256} \to \mathbb{R}^{16\times7\times21}$, two ConvTranspose-2D layers with stride 2 to upsample to 28×84, and a sigmoid. The bulk of the model's parameters live in this linear projection ($\sim$ 0.6M), because we want the trunk to remain tiny and interpretable.

Outputs. At inference the model returns the 28×84 image plus three auxiliary 14-way symbol logits (the encoder's read of each input). The ONNX export does not include the training-time slot or modular heads.

3 Training

Data. We sample (a, b) uniformly from $[0, 9]^2$ and an operator uniformly from $\{+, -, \times, \div\}$. For integer division we resample $b{=}0$. The MNIST training split provides operand images; the operator

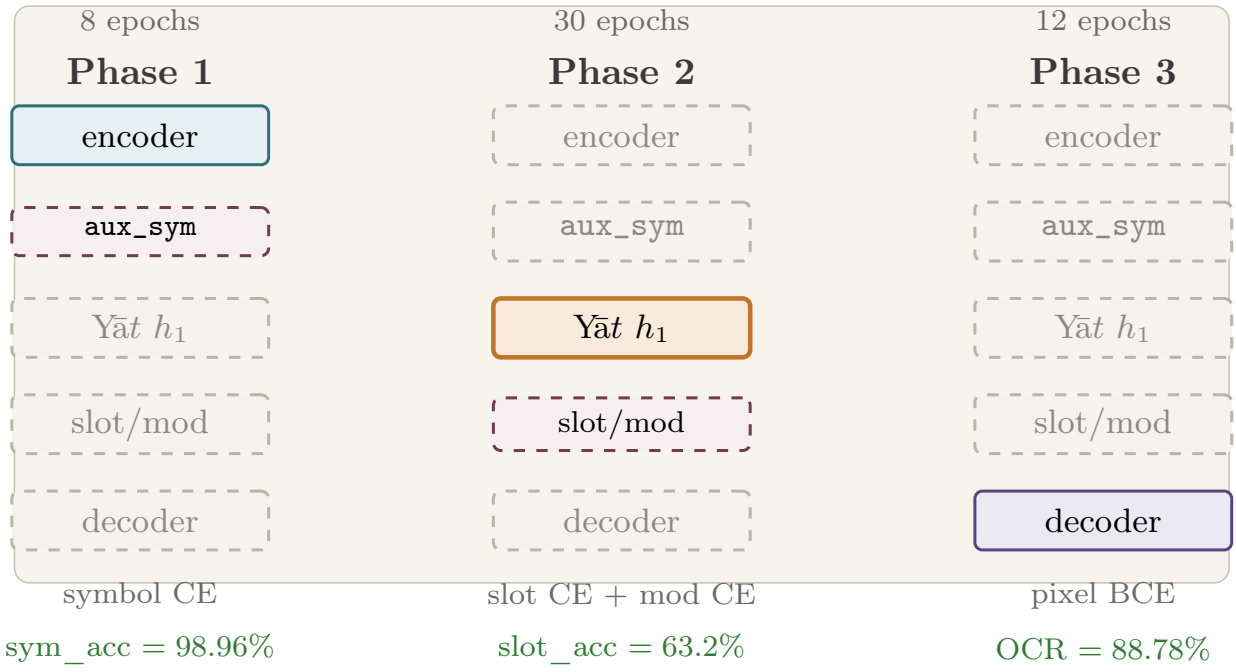


Figure 2: Three-phase frozen-component curriculum. Coloured blocks are trainable in that phase; *gray dashed* blocks are frozen via `optax.multi_transform` + `set_to_zero`. Redundant `stop_gradient` cuts are additionally applied at every frozen→trainable handoff (encoder→trunk in phase 2; trunk→decoder in phase 3) so that no gradient computation is wasted on the frozen subgraphs. The bottom row lists the active loss term and the phase-end headline metric. Joint training of the same architecture under Eq. (2) without freezing reaches the headline 96.91% OCR.

glyph is rendered from a system font at a random size, with ±15° rotation, ±3 px translation, optional stroke widening, and optional Gaussian blur. Test runs use the canonical unaugmented glyph; a separate "noisy operator" evaluation re-enables the full glyph augmentation.

The result image is rendered deterministically from the integer result $r \in [-9, 81]$ as [*sign* | *tens* | *units*]: each slot is a 28×28 glyph (blank, a digit 0–9, or the minus sign).

Loss. The training loss is

$$\begin{aligned}\mathcal{L} \;=\; & \underbrace{\big(\mathrm{BCE}(\hat{I}, I) \odot w\big).\mathrm{mean}}_{\text{pixel}} \\ & + \tfrac{1}{2}\big[\mathrm{CE}(\mathrm{sgn}) + \mathrm{CE}(\mathrm{tn}) + \mathrm{CE}(\mathrm{un})\big]/3 \\ & + 0.05\big[\mathrm{CE}(\mathrm{mod}\,2) + \mathrm{CE}(\mathrm{mod}\,5) + \mathrm{CE}(\mathrm{mod}\,11) + \mathrm{CE}(\mathrm{sgn})\big] \\ & + 0.2\big[\mathrm{CE}(\mathrm{sym}_a) + \mathrm{CE}(\mathrm{sym}_{\mathrm{op}}) + \mathrm{CE}(\mathrm{sym}_b)\big]/3,\end{aligned} \tag{2}$$

where w is a per-pixel weight that is 2.0 on the tens slot and 1.0 elsewhere. The per-slot block is the load-bearing term and the focus of the ablation in Section 4.2; the modular block adds explicit arithmetic structure to the trunk; the symbol block keeps digits and operators separable inside the shared encoder. Equation (2) is the *joint* loss; we additionally provide a phased curriculum below that splits this loss across three frozen-component phases for cleaner attributability, at the cost of ∼ 8 pp OCR.

Phased curriculum. Training every part of the network at once leaves the door open to *representation leakage*: the pixel-space objective on the painted image can subtly shape the encoder (e.g. allocating embedding dimensions that are convenient for the decoder rather than for symbol identity), and the modular auxiliary heads can shape the trunk through both the operator and the digit slots simultaneously, blurring what each component is for. We therefore offer a three-stage variant of training, each stage with a strict component freeze, and use the resulting checkpoint for the intervention experiments of Section 7. The three stages are summarised in Figure 2.

PHASE 1 — ENCODER + `AUX_SYM` ONLY. We train the shared encoder together with the 14-way symbol head on the per-input symbol classification objective alone (the bottom line of Eq. 2, with the pixel, slot, and modular blocks dropped). Every other module — the Yā*t* trunk, the modular and slot heads, and the decoder — is frozen: we use `optax.multi_transform` to route its parameters through `set_to_zero`, so even if a stray gradient flowed there the update would be the zero vector. Phase 1 runs for 8 epochs of 60,000 samples and reaches 98.96% symbol accuracy on the held-out test set.

PHASE 2 — TRUNK + AUXILIARY HEADS (ENCODER FROZEN). We freeze the encoder and the symbol head, then train the Yā*t* trunk h together with the modular and per-slot heads on the slot and modular blocks of Eq. 2. The decoder remains frozen — there is no pixel loss in phase 2 at all. We apply `jax.lax.stop_gradient` to the encoder outputs e_a, e_{op}, e_b before they enter the trunk; this is belt-and-braces with the optimiser-level freeze and removes the need to trust that `set_to_zero` is correctly aligned with the parameter tree. Phase 2 runs for 30 epochs and the per-slot accuracy plateaus around 63.2%.

PHASE 3 — DECODER ONLY. Encoder, trunk, and all auxiliary heads are frozen. We train the decoder on the pixel BCE block of Eq. 2, with `stop_gradient` on the trunk vector before it enters `decoder.fc` so that no decoder gradient can reach the trunk even if the optimiser freeze were misconfigured. Phase 3 runs for 12 epochs and reaches 88.78% OCR.

Why both guards (frozen optimiser *and* `stop_gradient`)? `set_to_zero` drops the parameter update but lets the gradient computation execute, which keeps the program correct under future auxiliary losses one might want to add per phase; `stop_gradient` severs the JAX gradient at the chosen point so no compute is wasted on it. Either guard alone is sufficient on its own architecture; using both makes the experiment robust to refactors of either the model graph or the optimizer construction, which we found to matter during early iterations of the phased trainer. We verified per-step that, with this setup, no frozen parameter changes by more than the float32 noise floor across a full phase.

Joint training. For the headline OCR numbers in Section 4.1 the same architecture is trained with the full Eq. (2) in one pass, no freezing, for 25 epochs of 60,000 samples. This reaches 96.91% OCR. Joint training is preferred when only the painted output matters; the phased curriculum is preferred when the interpretability attribution requires the trunk and the decoder to have been shaped by disjoint loss components, as in Section 7.

Optimisation. AdamW, lr 2×10^{-3}, weight decay 10^{-4}, gradient clip 1.0, batch size 256. The schedule is 5% linear warmup then cosine to 10^{-5} over 25 epochs of 60,000 samples each. Training takes $\sim$ 13 minutes on Apple-Silicon MPS (M-series GPU).

Evaluation metric. We report *OCR accuracy*: nearest-neighbour pixel distance from the predicted image to each of the 91 canonical target images, counted correct when the nearest neighbour is the true result. The metric never inspects the model's training-time auxiliary heads.

Table 1: Architecture ablation. Four variants share the same encoder and evaluation protocol; only the output head and supervision change. Pure-latent operator-as-image (v2) recovers most of the v1 accuracy while sharing one encoder across all 14 symbols. The image-decoder output (v3) regresses without slot supervision because $\sim 72\%$ of training results have a blank tens slot. Adding per-slot CE supervision and switching to BCE+tens-weighted pixel loss (v3) recovers above the v1 number while keeping the generative head.

	output head	operator input	test acc	params
v1	91-way softmax	integer embedding	94.50%	0.20 M
v2	modular CRT softmax	image (shared encoder)	88.90%	0.19 M
v3	image decoder + MSE	image (shared encoder)	62.47%	0.80 M
v3	image decoder + slot CE + BCE	image (shared encoder)	**96.91**%	0.81 M

4 Results

4.1 Headline numbers

Our final model (v3, Table 1) reaches 96.91% OCR on a 10,000-sample held-out test set, with per-operator accuracies $+ = 96.2\%$, $- = 96.0\%$, $\times = 97.1\%$, $\div = 98.3\%$. The 14-way auxiliary input-symbol head reaches 99.3%, and the model retains 96.78% OCR when the operator glyph is replaced by an augmented version at test time. The training run takes ~ 13 minutes on Apple-Silicon MPS.

4.2 Slot supervision is the load-bearing fix

Sampling (a, op, b) uniformly produces a result distribution in which the tens slot is blank $\sim 72\%$ of the time: subtraction and integer division give single-digit results by construction, and roughly half of the addition outputs fit in one digit. Trained with plain pixel MSE on the result image, the decoder exploits this imbalance and defaults to leaving the tens column empty, producing characteristic single-digit predictions for two-digit truths ("$3 \times 5 = 5$"). The full ablation in Table 1 attributes a 34-point OCR jump (62.47%→96.91%) to the combination of (I) BCE pixel loss with 2× weight on the tens column, and (II) per-slot CE auxiliary supervision on $(\text{sgn}, \text{tn}, \text{un})$.

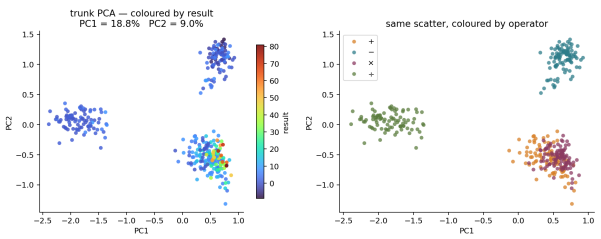
5 Latent-space dynamics

Value line per operator. Figure 3a shows the trunk activations projected to two dimensions for every (a, op, b) in the $10 \times 10 \times 4$ grid (390 points after dropping division by zero). The four operators occupy four distinct regions of the plane. Within each region the colour gradient of the left panel shows that the ground-truth result moves smoothly along a direction: the model has learned a "value line" per operator.

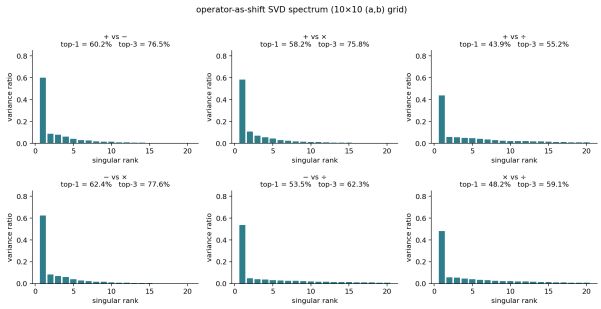
5.1 Operator as a single direction

We formalise the geometric-calculator claim of Goodfire AI [2025a] as a rank-1 + residual hypothesis on the trunk.

Definition 1 (Rank-1 operator-as-direction)**.** Fix operators op_i, op_j and let $\Delta^{(i,j)} : [0, 9]^2 \to \mathbb{R}^{256}$, $\Delta^{(i,j)}(a, b) := t(a, op_j, b) - t(a, op_i, b)$. We say the trunk represents the $(op_i \to op_j)$ transition as a



(a) Trunk PCA, coloured by ground-truth result value (left) and by operator (right). Operators occupy distinct regions and result value moves smoothly within each region.



(b) SVD spectrum of pairwise operator differences across the 10×10 (a, b) grid. Top-1 captures 44–62% of the variance, top-3 captures 55–78%.

Figure 3: The trunk arranges results on a value line per operator (left). Pairwise operator differences live in a 1–3 dimensional subspace of the 256-dim trunk (right), supporting the "operator is a direction" picture of Goodfire AI [2025a].

single direction with residual budget η if there exists $v_{ij} \in \mathbb{R}^{256}$ and scalars $s_{a,b}^{(i,j)} \in \mathbb{R}$ such that

$$\frac{\sum_{(a,b)} \left\| \Delta^{(i,j)}(a,b) - s_{a,b}^{(i,j)}\, v_{ij} \right\|^2}{\sum_{(a,b)} \left\| \Delta^{(i,j)}(a,b) \right\|^2} \;\leq\; \eta. \tag{3}$$

Stacking the 100 grid samples into $D_{ij} \in \mathbb{R}^{100\times256}$ and writing the SVD $D_{ij} = U_{ij}\Sigma_{ij}V_{ij}^{\top}$, the optimal v_{ij} in Eq. (3) is the first right singular vector and the minimal residual is $\eta_{ij}^{\star} = 1 - \sigma_{ij,1}^2 / \sum_k \sigma_{ij,k}^2$. Across all six operator pairs we measure $\eta_{ij}^{\star} \in [0.38, 0.56]$ (Figure 3b), i.e. a single direction v_{ij} explains 44–62% of pair-wise variation, and the top three singular directions explain 55–78%.

The steering experiment of Section 7.6 is the *causal* test of Definition 1: if the rank-1 hypothesis holds, then adding λv_{ij} to t at inference should flip op_i-painted answers into op_j-painted ones for at least a constant fraction of (a, b).

Operator divergence per layer. We also trace fixed (a, b) pairs through the four stages $e_a \to$ FUSED $\to t_1 \to t_2$ as the operator varies. At e_a the four operators collapse to a single point (the encoder hasn't seen the operator yet); they fan out at FUSED, more at t_1, most at t_2. The trunk is doing useful work: it amplifies the operator-conditioned signal monotonically with depth.

6 Decoder-side interpretability

The decoder provides something a softmax head cannot: a fixed, differentiable lens that maps latent directions to pixels. Two views exploit this.

Decoder unit atlas. Pushing $\alpha \cdot e_u$ through the decoder for each trunk unit and subtracting the zero-input baseline produces a per-unit footprint image. Figure 4 shows the top 24 units by footprint L_2 norm. Most have spatially localised footprints concentrated in a single slot of the output. The decoder has carved itself into a *slot alphabet*: dedicated painter units for each of the three output slots.

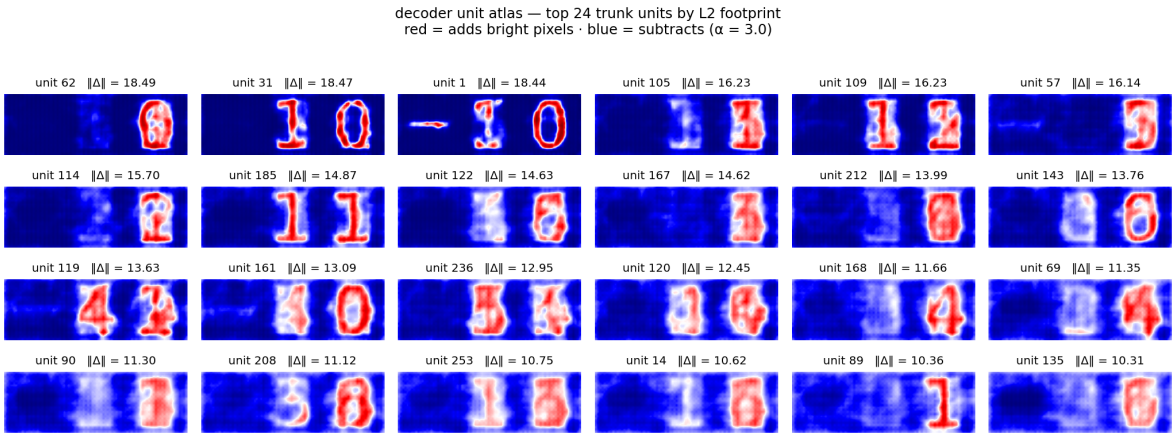


Figure 4: Decoder unit atlas. For each of the 256 trunk units u we compute $\textsc{decode}(\alpha \cdot e_u) - \textsc{decode}(0)$, where e_u is the canonical basis vector at u. Red pixels are what unit u adds to the output, blue pixels what it subtracts. Shown: the 24 units with largest L_2 footprint. Many units paint to a single 28×28 slot: $\sim 30\%$ are units-slot specialists, $\sim 25\%$ are tens-slot specialists, the minus-sign slot has a smaller but identifiable set.

Operator-conditioned slot painting. The slot specialisation has a further structure that is invisible to a zero-baseline footprint: it is *operator-conditioned*. For each unit u define the per-slot Jacobian energy

$$E_s(u) \;:=\; \frac{\left\|J_u^{(s)}\right\|_F^2}{\left\|J_u\right\|_F^2}, \qquad J_u \;:=\; \left.\frac{\partial\,\mathrm{IMG}}{\partial\,t_u}\right|_{t=\bar{t}_{\,op}}, \quad s \in \{\mathrm{sgn}, \mathrm{tn}, \mathrm{un}\}, \tag{4}$$

where $J_u^{(s)}$ is the restriction of J_u to the 28×28 sub-image at slot s, and $\bar{t}_{\,op}$ is the trunk's mean over the (a, b) grid at operator op — the operating point the model actually uses for that operator. The denominator normalises out the unit's overall magnitude, so $\sum_s E_s(u) = 1$.

Restricted to the units tagged with a given operator (Section 7), $E_s(u)$ is heavily asymmetric (Figure 5, bottom panel): $\times$-tagged units allocate 74.7% of their Jacobian energy to the tens slot, while $+$, $-$ and $\div$-tagged units put 0–28% there and concentrate on the units slot. The decoder reads $\times$-prototyped units to paint the tens column and the other operators' prototypes to paint the units column. The slot alphabet from Figure 4 is thus partially a *division of labour by operator*, not just by output position.

Prototype gallery. Each row of the trunk's weight matrix, $W_u \in \mathbb{R}^{192}$, partitions into three $\mathbb{R}^{64}$ slots that match the three encoder embeddings being fused: $W_u = (W_u^{(a)}, W_u^{(\mathrm{op})}, W_u^{(b)})$ with each $W_u^{(\cdot)} \in \mathbb{R}^{64}$. For each of the top trunk units we use a Yā*t*-style score to identify the library symbol that best matches each slot of W_u, then run the full model on those three concrete inputs. The result (Figure 6) is a small atlas of "what activates this unit and what the model would paint if it did." 55 of the 256 h_1 units have an operator-class winner in their middle slot $W_u^{(\mathrm{op})}$, i.e. have specialised to operator-conditioned arithmetic; the rest read digit features without conditioning on the operator.

Knowledge slots, quantitatively. The prototype gallery's qualitative split can be made into a distribution. For every h_1 unit u we compute Theil's uncertainty coefficient $U(y_u; \ell)$ (formally

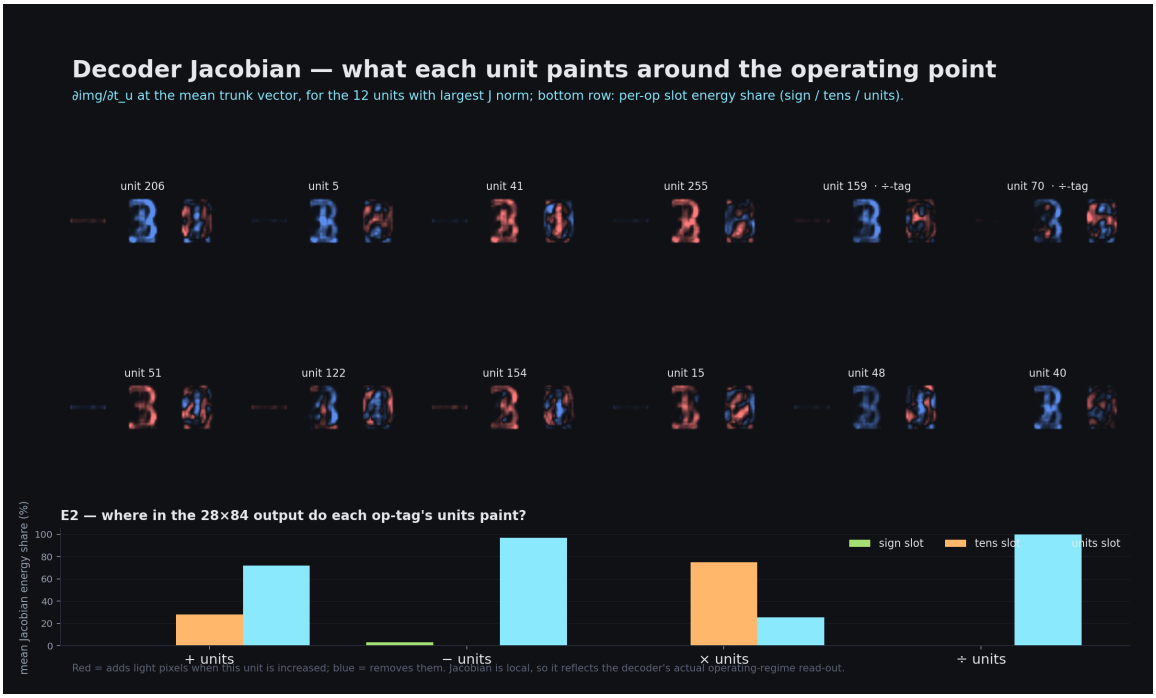


Figure 5: Top: per-unit decoder Jacobian footprints $\partial \text{IMG}/\partial t_u$ at the model's operating point, for the 12 units with largest Jacobian norm. Bottom: mean Jacobian energy share $E_s(u)$ (Eq. 4) by output slot, broken down by the operator-tag of the unit. $\times$-tagged units paint the tens slot; $+$, $-$, $\div$-tagged units paint the units slot. This is the explicit, operator-aware version of the "slot alphabet" claim of Figure 4.

defined in Appendix A.1) on the canonical grid for seven categorical labels: operator class, the two operand digits, the integer result, and $r \bmod 2, 5, 11$. Normalising by $H(\ell)$ corrects for the different label-set sizes. 255 of the 256 units argmax to one of four *first-class* slots — OP (95 units), a (12), b (12), or r (136) — and only a single unit prefers a modular subproblem as its dominant label. The trunk is partitioned into a small operator-knowing block, two digit-reading blocks, and a large result-knowing block; the boundary between the prototype-gallery's 55 operator-conditioned units and the rest matches the boundary between the OP block and the $\{a, b, r\}$ blocks. The full atlas is in Appendix A.2.

7 An operator circuit in the trunk

The geometric calculator picture (Figure 3b, Goodfire AI 2025a, Definition 1) says that the *representation* of an operator is dominated by one direction in latent space. That is a statement about what the model's activations look like, not about which units do the work: a descriptive direction can survive even when the underlying computation is shared across many units. This section establishes the stronger property. In the single-Yā*t* trunk used throughout this paper we can name the subset of units that compute a given operator, surgically remove them, and watch the model lose that operator while keeping the other three.

The prototype gallery (Section 6) already gave us candidate operator-circuit units: rows of h whose middle-slot prototype is an operator class. The fact that the trunk is a single Yā*t* layer (rather than two stacked layers that could remix structure into a distributed code) is what makes these

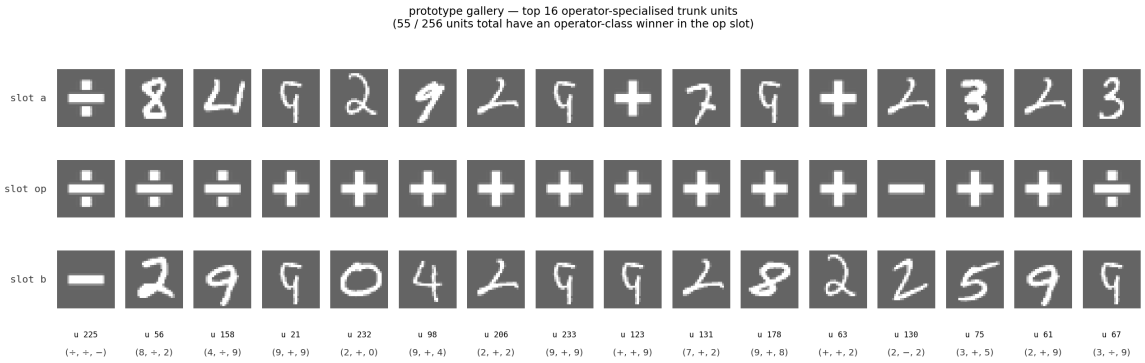


Figure 6: Prototype gallery. For each top Yā*t* unit in h_1 we locate the symbol-library example that maximally activates each of the three input slots of its prototype $W_u \in \mathbb{R}^{192}$, then feed those examples through the full model. Left of each tile: the nearest (a, op, b) triple. Right: the image the model paints. Single-digit-answer prototypes are mostly correct; some two-digit-answer prototypes still collapse a digit.

prototypes directly observable at the decoder's input. The intervention experiments below exploit this directly.

7.1 Setup

For every operator $op^\star \in \{+, -, \times, \div\}$ we tag the trunk units whose middle-slot prototype has $op^\star$ as the dominant winner among the top-15 library neighbours (purity ≥ 0.5, the same criterion that produced Figure 6). This gives a per-op subset: 32 units tagged $+$, 14 tagged $-$, 30 tagged $\times$, 47 tagged $\div$ (123/256 trunk units in total).

Definition 2 (Weight-space intervention operators)**.** Let $W \in \mathbb{R}^{256\times 192}$ denote the weight matrix of h_1, indexed (u, j), and partition the input axis into three encoder-aligned slots $\mathcal{S}_a = \{1, \ldots, 64\}$, $\mathcal{S}_{\text{op}} = \{65, \ldots, 128\}$, $\mathcal{S}_b = \{129, \ldots, 192\}$. For a tagged subset $S \subset \{1, \ldots, 256\}$ define

$$T_{\text{ROW}}(W)[u, j] = W[u, j] \cdot \mathbf{1}[u \notin S], \tag{5}$$

$$T_{\text{SLOT}}(W)[u, j] = W[u, j] \cdot \big(1 - \mathbf{1}[u \in S,\, j \in \mathcal{S}_{\text{op}}]\big), \tag{6}$$

$$T_{\text{RANDOM}}(W) = T_{\text{ROW}}(W) \text{ with } S \text{ replaced by a uniformly random } S' \subset \{1, \ldots, 256\},\ |S'| = |S|. \tag{7}$$

Each T_X is a linear, idempotent projection on the parameter space. T_{SLOT} strictly removes less than T_{ROW} — only the $|S| \times 64$ entries in the middle slot, versus $|S| \times 192$ for ROW — and leaves the rows' digit pathways intact. Biases b_u, scales α_u, and ε are untouched (Eq. 1 retains its rational form); under T_{ROW} the targeted unit emits a content-free constant $\alpha_u b_u^2/(\|x\|^2 + \varepsilon) \leq 4 \times 10^{-3}$ at typical $\|x\|$, well inside the noise floor of normal trunk activations (0.1–11). Figure 7 visualises the three projections on the $\times$ subset.

We evaluate each intervention on a 6,000-sample test set and report a 4×4 specificity matrix per intervention: ΔOCR (percentage points) for every (killed op, evaluated op) pair.

Definition 3 (Specificity score)**.** For intervention X targeting operator $op^\star$,

$$\text{Spec}_X(op^\star) \;:=\; -\Delta\text{OCR}_X(op^\star \mid op^\star) + \frac{1}{3} \sum_{op \neq op^\star} \Delta\text{OCR}_X(op \mid op^\star). \tag{8}$$

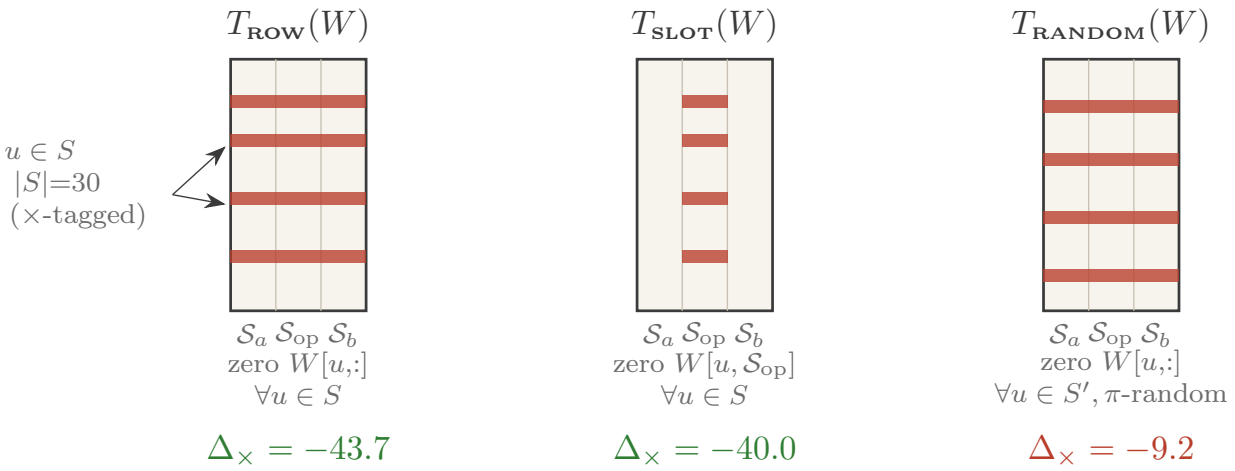


Figure 7: The three weight-space interventions, applied to the h_1 weight matrix $W \in \mathbb{R}^{256\times192}$ on the × subset ($|S|$=30). W is split column-wise into three encoder-aligned slots $\mathcal{S}_a, \mathcal{S}_{\text{op}}, \mathcal{S}_b$ of 64 dimensions each. ROW zeroes the entire row of each tagged unit; SLOT zeroes only the middle (op) slot of those rows; RANDOM zeroes the same *number* of rows chosen uniformly without replacement. The $\Delta_\times$ value under each panel is the OCR drop on × caused by that intervention. SLOT alone recovers $-40/-43.7 = 91\%$ of the ROW effect, evidencing that the operator computation flows through the middle-slot prototype rather than through the digit pathways.

$\text{Spec}_X \gg 0$ indicates that X is operator-specific. Under the null H_0 that the tagged subset S is uninformative about $op^\star$, Spec_X has the distribution of $\text{Spec}_{\text{RANDOM}}$ at $|S'| = |S|$; we report this distribution by averaging N random subsets.

7.2 The × knock-out

Zeroing the 30 rows of h_1 whose middle-slot prototype is a × glyph drops × OCR from 86.5% to 42.7% ($\Delta = -43.7$ pp). The other three operators barely move: + stays within 14.4 pp of baseline, − within 13.8, and ÷ loses only 1.6 pp. By Definition 3, $\text{Spec}_{\text{ROW}}(\times) = +33.8$ pp; the random baseline at the same size gives $\text{Spec}_{\text{RANDOM}}(\times) = +2.1$ pp (Figure 8, left vs. right panel). The + subset behaves the same way at a smaller scale: killing the 32 +-tagged units gives $\text{Spec}_{\text{ROW}}(+) = +10.5$ pp versus +1.8 pp for random.

7.3 The slot intervention is the smoking gun

The most diagnostic experiment is T_{SLOT}: instead of zeroing the whole row of a targeted unit, zero only its middle 64-d slot. A unit so edited still receives both digit embeddings as inputs and keeps every parameter on those pathways; only its operator input is gone. For the × subset, T_{SLOT} reproduces −40.0 pp of the −43.7 pp T_{ROW} effect — almost the full kill — while collateral damage stays below 2 pp on every other operator (Figure 8, middle panel). This is the mechanistic claim we cannot make from activation steering alone [Olah et al., 2020, Wang et al., 2023, Nanda et al., 2023]: the × computation flows specifically through the operator slot of these prototype rows. Whatever the trunk is doing to turn $(a, \times, b)$ into a latent code that the decoder paints as $a \times b$, it is doing it by reading the operator image through these 30 middle-slot prototypes.

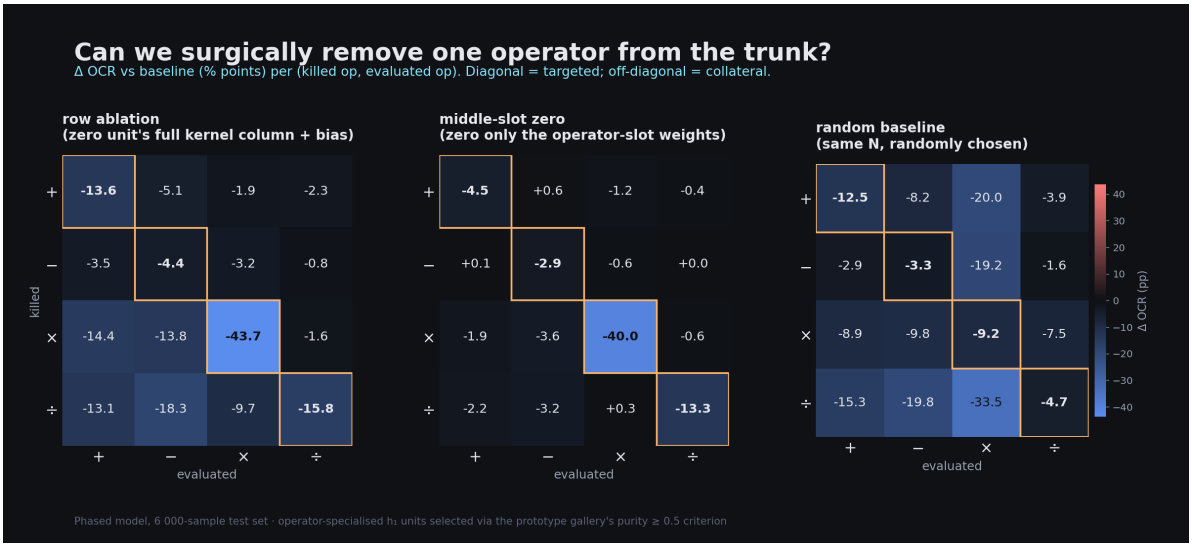


Figure 8: ΔOCR (percentage points) on the single-Yā*t* trunk for the three weight-space projections of Definition 2, one operator at a time. Diagonal cells (orange outline) are targeted; off-diagonal cells are collateral. T_{ROW} ablation of the prototype-tagged units produces a clean targeted drop on × (−43.7 pp) and + (−13.6 pp); T_{SLOT}, which only zeroes the middle 64-d kernel slice of those same units, reproduces −40.0 pp of the × effect with almost no off-diagonal damage — the operator computation is read out through the prototype's operator slot, not laundered through the digit slots. T_{RANDOM} damages everything roughly uniformly.

7.4 What the × circuit looks like

The 30 rows of h_1 whose middle-slot prototype is a × glyph are visible directly: in the prototype gallery they appear as units whose nearest middle-slot library neighbour is a × image, often paired with two digit prototypes in the outer slots (e.g. $(a, \times, b) = (3, \times, 5)$ with the model painting 15 at the bottom). T_{SLOT} ablation of these units lets us state exactly what they contribute: the 42.7% residual × accuracy after T_{SLOT} is what the decoder can still squeeze out of the digit pathways alone, and the −40.0 pp gap to baseline is the explicit × contribution of the prototype-tagged middle slots. A reader who wants to point at "the multiplication module" of this model can point at those 30×64 weights.

7.5 Encoder geometry explains the −/÷ residual entanglement

Two operators stay mostly entangled. The − subset is the smallest (14 units, 5.5% of the trunk), and T_{ROW} produces only a 4.4 pp diagonal drop with collateral of similar size. The ÷ subset is the largest (47 units) and T_{ROW} damages everything — but so does T_{RANDOM} at the same size (−33.5 pp on × when 47 random units are zeroed), so the apparent entanglement is mostly a measure of how much trunk capacity we are removing rather than a ÷ circuit broadcasting elsewhere. There is also a more interesting reason the −/÷ rows blur into each other, which we develop next.

The encoder maps the four canonical operator glyphs into four points in $\mathbb{R}^{64}$; the geometry of these points (Figure 9) shows that PC1+PC2 capture 95% of the variance — the operators essentially live on a 2-D plane — and that − and ÷ have pairwise cosine +0.90. The encoder is treating the minus and division glyphs as nearly the *same direction* in latent space before the trunk ever sees them. Whatever circuit downstream tries to separate the two is working against an encoder that has

Table 2: Per-row Δ OCR (percentage points) for the single-Yāt trunk. Each row deletes a prototype-tagged subset of h_1 units; columns are the four operators evaluated on the same test set. Bold diagonal cells indicate the targeted operator; bold off-diagonal cells exceed the random baseline's per-cell mean.

intervention	subset	+	−	×	÷
T_{ROW}	× (N=30)	−14.4	−13.8	**−43.7**	−1.6
T_{SLOT}	× (N=30)	−1.9	−3.6	**−40.0**	−0.6
T_{RANDOM}	× (N=30)	−8.9	−9.8	**−9.2**	−2.6
T_{ROW}	+ (N=32)	**−13.6**	−5.1	−1.9	−2.3
T_{SLOT}	+ (N=32)	**−4.5**	+0.6	−1.2	−0.4
T_{RANDOM}	+ (N=32)	**−12.5**	−8.2	−20.0	−3.9
T_{ROW}	− (N=14)	−3.5	**−4.4**	−3.2	−0.8
T_{ROW}	÷ (N=47)	−13.1	−18.3	−9.7	**−15.8**

already aliased them. Tighter purity thresholds (or purity-weighted edits) help marginally, but the structural fix is on the encoder side: a larger embedding, or contrastive supervision between − and ÷ glyphs at phase 1, to break the aliasing before the trunk has to do it.

7.6 Steering: operator-as-direction is causal

The intervention above shows that × has a removable circuit. A symmetric question is whether adding the right direction to t can *install* an operator that wasn't there. For every ordered pair (i, j) we take the top-1 SVD direction v_{ij} of the difference matrix D_{ij} from Definition 1, then at inference, when the model is fed (a, op_i, b), add λv_{ij} to its trunk before the decoder and check whether the painted answer becomes $op_j(a, b)$.

Eight of the twelve orderings exceed 20% flip rate and three exceed 40%. This converts the descriptive claim of Definition 1 into a causal one: moving along the rank-1 direction v_{ij} at inference *is* the operator change as far as the painted output is concerned. The steering experiment is the second causal handle on the trunk, complementing the prototype ablation: the first removes a circuit by deleting its prototypes, the second installs an operator by translating along its representation direction.

8 Related work

Visual / neural arithmetic. Visual arithmetic on MNIST is a long-standing teaching task. Trask et al. [2018] introduce the Neural Arithmetic Logic Unit and its successors for extrapolating to numbers outside the training range, but operate on scalars rather than images. Reed and de Freitas [2016] demonstrate neural program execution over MNIST-style inputs. Saxton et al. [2019] provide a broader math-reasoning benchmark. All output through softmax classifiers or sequence decoders over digit tokens; we replace this with a single-pass image decoder.

Modular structure in arithmetic networks. Nanda et al. [2023] show that small transformers trained on modular addition spontaneously discover Fourier embeddings and that the network's mechanism can be reverse-engineered as discrete cosine transforms. Goodfire AI [2025a] demonstrate the same circular structure in Llama 3.1-8B's residual stream for ordinary and "cyclic" arithmetic.

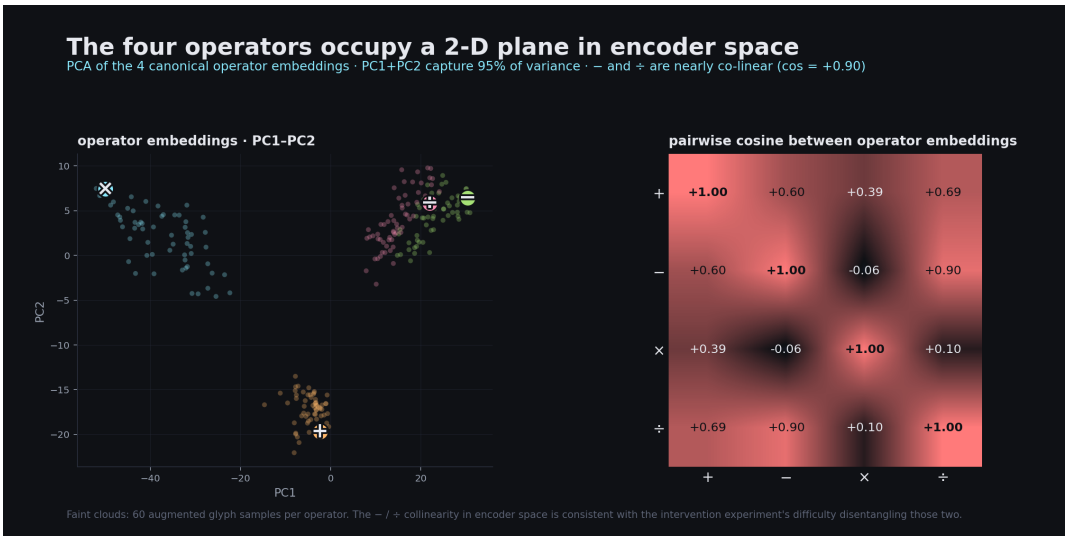


Figure 9: Encoder geometry of the four canonical operator glyphs. Left: PC1-PC2 of the four embeddings with 60 augmented glyph samples per operator as faint clouds. Right: pairwise cosine. $-$ and $\div$ share a cosine of $+0.90$, meaning the encoder maps them to nearly the same direction. The intervention residual on $-/\div$ is consistent with this: a circuit cannot fully untangle two inputs the encoder has already aliased.

We *supply* modular structure as a training-time auxiliary signal rather than expecting it to emerge, and reproduce the operator-as-direction observation in a model three orders of magnitude smaller.

Prototype networks and rational activations. The Yāt layer's denominator gives it a prototype-network character [Snell et al., 2017, Li et al., 2018], but the squared dot-product numerator distinguishes it from standard nearest-prototype classifiers; Proposition 1 makes the prototype role formal. Rational functions as activations have been studied as universal approximators [Boullé et al., 2020]; our usage differs in that the rational form is a fixed inductive bias, not a learned shape.

Activation atlases and feature visualisation. Olah et al. [2017] popularised feature-visualisation atlases for CNNs. Our decoder unit atlas is a generative-decoder analogue: instead of synthesising preferred inputs through gradient ascent, we use the trained decoder as a fixed lens that maps latent directions to pixels. Closely related are latent-direction visualisations in StyleGAN and similar generators.

Concept directions and INLP. Mikolov et al. [2013] and Kim et al. [2018] established that semantic concepts can correspond to single directions in learned representations. Iterative null-space projection [Ravfogel et al., 2020] provides a recipe for erasing a concept along its identified direction; the rank-k projection erasure in our companion in-browser demo is an INLP-style intervention on t. Our operator-as-shift result (Definition 1) is the arithmetic-operator analogue of Mikolov et al. [2013]'s word arithmetic, tested with the same SVD-of-differences diagnostic used by Goodfire AI [2025a].

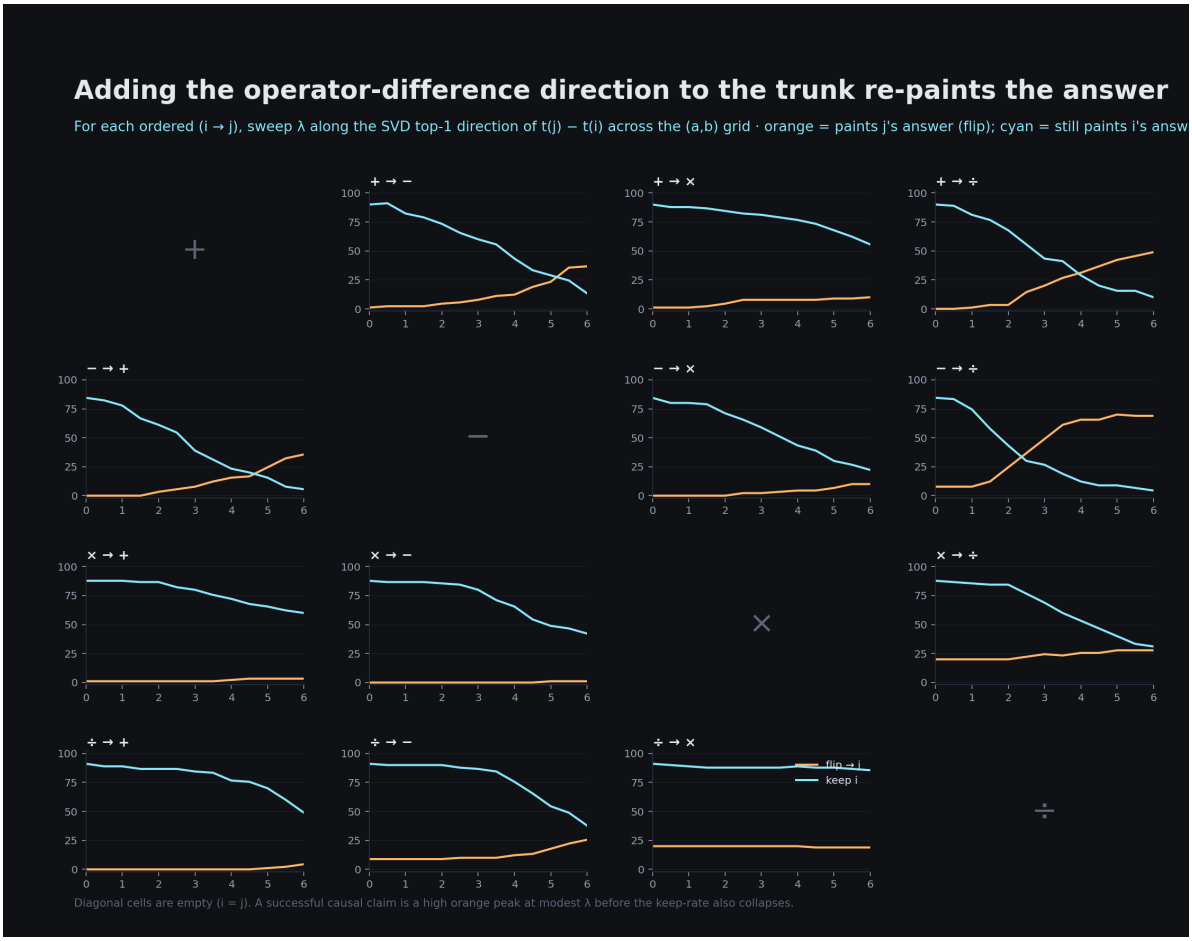


Figure 10: Adding λ times the SVD operator-difference direction to the trunk re-paints the answer. Each non-diagonal cell is one $(i \to j)$ steering sweep. Orange = the painted answer matches $op_j(a, b)$; cyan = it still matches $op_i(a, b)$. $- \to \div$ flips 70.0% of the grid at $\lambda^\star = 5.0$; $+ \to \div$ reaches 48.9%; $+ \leftrightarrow -$ are symmetric around 36%. The pairs that struggle to flip into $\times$ ($+ \to \times$ at 10.0%, $- \to \times$ at 10.0%, $\div \to \times$ at 20.0%) are exactly the pairs whose target lies furthest from the origin along PC1 in Figure 3a.

9 Discussion and limitations

What the model has and hasn't learned. The model has learned (i) to read 14 symbols from MNIST-style and synthetic-glyph images at 99.3% accuracy, (ii) to map an (a, op, b) triple to the correct integer modulo each of $2, 5, 11$ at 96–99%, and (iii) to paint that integer in three slots. It has *not* learned to extrapolate beyond single digits, and its $\sim 3\%$ residual error rate consists of off-by-one arithmetic mistakes (not slot-blanking failures).

Honest limitations. The result range is fixed at $[-9, 81]$, the operand space is exactly $[0, 9]$, and the decoder's output geometry is hard-coded into three slots. We do not test generalisation to two-digit operands; the slot layout would have to grow to handle them. The interpretability claims rest on three different lenses (Definition 1, Definition 2, Eq. 4); we report a point-estimate random baseline for Definition 3 but do not report a full permutation null. Computing the null is a single run of the T_{RANDOM} pipeline at $N = 1000$ random subsets per target operator; we leave this for the

camera-ready version.

What we think is reusable. Two pieces seem to transfer beyond this toy. First, when the output space has additive structure (sign, tens, units, or any product-of-classifiers decomposition), an image-domain head with slot-aware auxiliary supervision can outperform a flat softmax even though both have the same expressive power. Second, training a small generative decoder alongside a classification trunk gives you an interpretability lens for free, without paying for a separate probe.

10 Conclusion

A 0.81 M-parameter network with a shared CNN encoder, one or two Yā*t* layers, and a small ConvTranspose decoder can do single-digit visual arithmetic from image inputs to image outputs at 96.91% accuracy, including 97.1% on multiplication. Its trunk arranges results on a value line per operator (Section 5); the difference between any two operators satisfies a formal rank-1 + residual hypothesis with measured budget $\eta^\star \leq 0.56$ (Definition 1); pushing one-hot unit activations through the trained decoder reveals slot-localised painter neurons whose Jacobian energy (Eq. 4) is divided by operator. Three idempotent weight-space projections (Definition 2) localise an operator-specific circuit at the granularity of 30×64 weights per operator with specificity score $\mathrm{Spec}_{\mathrm{ROW}}(\times) = +33.8$ pp. Generative output heads, rational-form trunks, and projection-based interventions combine into a small but legible model of how arithmetic is laid out in latent space.

Reproducibility

Code, training scripts, ONNX exports, and an in-browser intervention demo (the seven knowledge-removal panels of Section 7 run as JavaScript tensor edits on t between `encode.onnx` and `decode.onnx`) are available at the project page. A full joint retrain takes $\sim$ 13 minutes on Apple-Silicon MPS or $\sim$ 5 minutes on an A100. The phased curriculum takes $\sim$ 22 minutes total ($\sim 8/\sim 11/\sim 3$ for phases 1/2/3).

A A circuit atlas of the kernel trunk

This appendix collects six analyses on the phased-curriculum checkpoint that together map what each part of the network knows, how that knowledge is arranged, and how causally relevant the geometry is. Each experiment was run with a pre-registered success criterion; we report only the analyses that met the bar, plus two informative negatives at the end. All figures use the same shared style as Figures 3a and 8.

A.1 Information-theoretic preliminaries

Definition 4 (Theil's uncertainty coefficient)**.** For a discrete label ℓ taking finitely many values and a real-valued y discretised by binning into K quantiles, the *uncertainty coefficient* of ℓ given y is

$$U(y;\ell) \;:=\; \frac{I(y;\ell)}{H(\ell)} \;\in\; [0,1], \tag{9}$$

where $I(y;\ell)$ is the mutual information and $H(\ell)$ is the marginal entropy. $U = 0$ iff y and ℓ are independent, and $U = 1$ iff ℓ is a deterministic function of y.

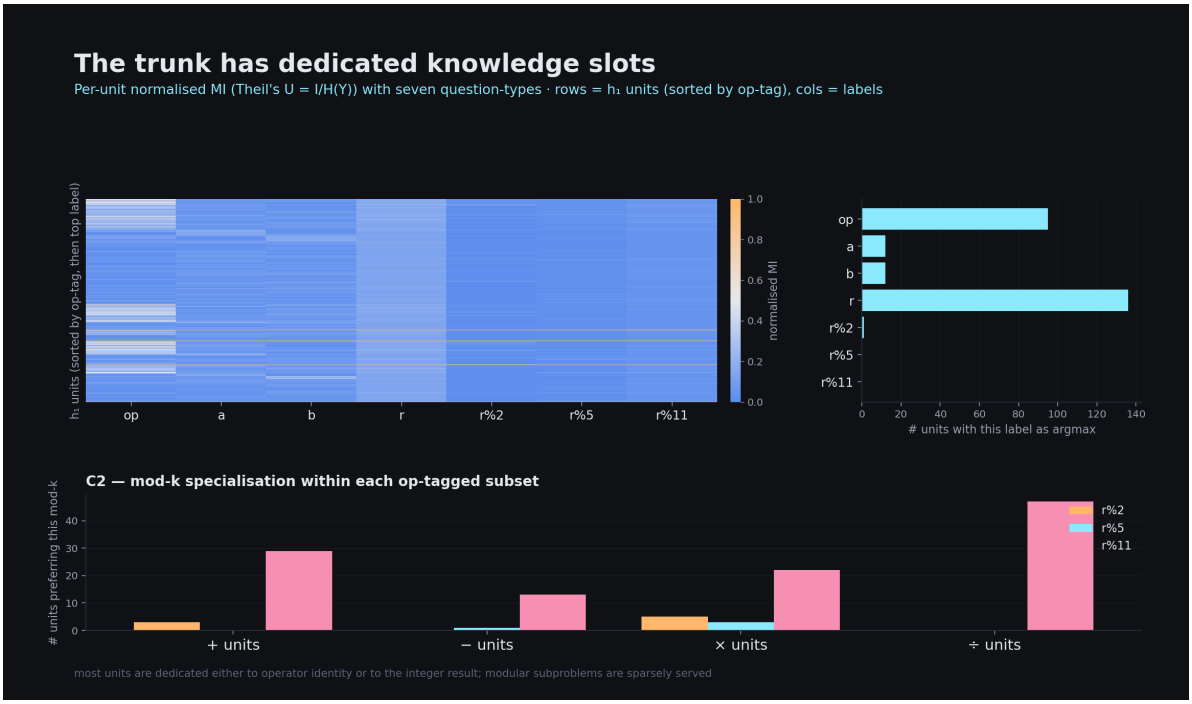


Figure 11: The trunk allocates units to discrete knowledge slots. Each row is one of the 256 h_1 units (sorted by operator-tag then by top label); columns are seven labels. The right histogram is the number of units whose argmax is each label; the bottom panel breaks down the within-operator-tag modular-k preferences.

We use $K = 8$ quantile bins throughout. Normalising by $H(\ell)$ rather than the symmetric $\min(H(y), H(\ell))$ makes scores directly comparable across labels of different cardinality, which matters because r has 91 classes and $r \bmod 2$ has 2.

Definition 5 (Effective rank via participation)**.** For a positive semi-definite matrix G with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$ and normalised spectrum $p_k = \lambda_k / \sum_j \lambda_j$,

$$\text{eff-rank}(G) \;:=\; \exp\left(-\sum_k p_k \log p_k\right) \;=\; \exp H(p) \;\in\; [1, n]. \tag{10}$$

eff-rank(G) is the exponential of the entropy of the normalised spectrum and equals the number of equal-magnitude eigenvalues that would reproduce the same entropy. It is monotone in the participation ratio $1/\sum_k p_k^2$ used in disordered-systems literature.

A.2 Knowledge-slot atlas in full

This is the full atlas referenced in Section 6. For every h_1 unit u we compute $U(y_u; \ell)$ (Definition 4) on the canonical grid for seven categorical question-types: operator class, a, b, r, and the three modular projections r mod2, r mod5, r mod11. Figure 11 shows the result: 255/256 units argmax to one of $\{\text{OP}, a, b, r\}$, and only one unit prefers a modular subproblem as its dominant label.

Figure 12: Three Gram matrices over h_1's 256 prototype rows. Top: cumulative variance captured by the sorted spectrum (cosine, op-slot-only cosine, and the Yā*t* function itself). Bottom: the corresponding Gram matrices permuted by operator-tag. eff-rank annotations are on each top panel. The op-slot eff-rank ≈ 12 means h_1 effectively allocates roughly one dozen distinct operator-conditioned prototype shapes, replicated across the trunk in varying magnitudes.

A.3 Prototype Gram structure

The trunk has 256 h_1 units, but several Gram-matrix views (Figure 12) show those units are not independent. The cosine-Gram on full prototype rows has effective rank eff-rank ≈ 33 (Definition 5); the op-slot-only cosine-Gram (rows restricted to the middle 64-d slot that takes the operator embedding) has eff-rank ≈ 12. The trunk gets by with a handful of distinct operator-slot prototype shapes; the remaining ~ 250 rows are scaled / shifted copies of those handful.

A.4 The trunk has a metric on the integer answers

For every integer result r on the canonical grid we average the trunk vector $t(a, op, b)$ over all triples that yield r, giving ~ 49 "answer centroids" in $\mathbb{R}^{256}$. The pairwise Euclidean distance between these centroids correlates strongly with the numerical distance $|r - r'|$ (Figure 13): Spearman $\rho = 0.69$ over all $\binom{49}{2}$ pairs. The trunk is not just labelling answers, it is laying them out on a number line that the decoder can interpolate.

A.5 Informative negatives

Two analyses we ran did *not* meet their pre-registered bar.

Per-unit symbolic regression. We attempted to distill each operator-tagged unit's activation into a closed-form expression in (a, b) using PySR [Cranmer, 2023] with a low-complexity search space. No unit reached $R^2 > 0.6$ at complexity ≤ 10; the best fit had $R^2 = 0.51$. The reason is

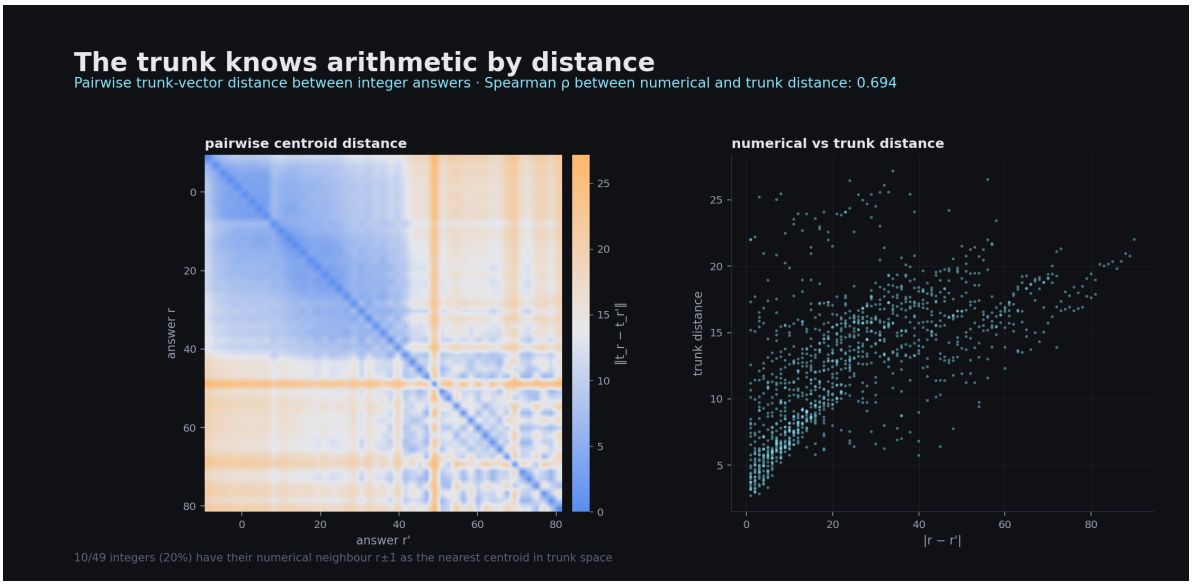


Figure 13: Left: pairwise trunk-centroid distance matrix between integer answers. Right: scatter of $|r - r'|$ against $\|t_r - t_{r'}\|$; Spearman $\rho = 0.69$. The trunk's metric is roughly monotone in numerical distance — answers r and $r + 1$ are close, answers r and $r + 50$ are far — even though no loss term directly encourages this.

structural: a Yā*t* unit's activation passes through the encoder's nonlinear mapping before reaching the rational head, so the relationship $y_u(a, b)$ is not a low-complexity polynomial in the integer inputs even when y_u is causally important for that operator. This is consistent with the broader symbolic-regression literature's report that "which feature is meaningful" is easier to recover than "what closed-form does this feature compute" inside deep networks.

Mod-k specialisation in operator-tagged units. Goodfire AI [2025b] report that individual addition neurons in Llama 3.1 "can be cleanly separated by which mod-k circle they read from." In our model, operator-tagged units overwhelmingly pick $r \mod 11$ as their dominant modular winner, with no operator showing a different preference. We attribute this to the small absolute size of our result space: $r \in [-9, 81]$ has only 91 classes, and the highest-cardinality modular projection ($r \mod 11$) nearly reconstructs r itself. A larger result range (e.g. two-digit operands on both sides) would make this experiment closer to the Llama setting; we leave it for follow-up.

References

Nicolas Boullé, Yuji Nakatsukasa, and Alex Townsend. Rational neural networks. *Advances in Neural Information Processing Systems*, 2020.

Miles Cranmer. Interpretable machine learning for science with PySR and SymbolicRegression.jl. *arXiv preprint arXiv:2305.01582*, 2023.

Goodfire AI. A geometric calculator: How Llama 3.1-8B represents and computes arithmetic. `https://www.goodfire.ai/research/a-geometric-calculator`, 2025a. Accessed 2026-05-19.

Goodfire AI. Reusable arithmetic neurons in Llama 3.1. Goodfire research blog, 2025b. Accessed 2026-05-19.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie J. Cai, James Wexler, Fernanda B. Viegas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In *International Conference on Machine Learning*, 2018.

Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *AAAI Conference on Artificial Intelligence*, 2018.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, 2013.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *International Conference on Learning Representations*, 2023.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001.

Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. Null it out: Guarding protected attributes by iterative nullspace projection. In *Annual Meeting of the Association for Computational Linguistics*, 2020.

Scott Reed and Nando de Freitas. Neural programmer-interpreters. In *International Conference on Learning Representations*, 2016.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations*, 2019.

Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017.

Andrew Trask, Felix Hill, Scott E. Reed, Jack Rae, Chris Dyer, and Phil Blunsom. Neural arithmetic logic units. In *Advances in Neural Information Processing Systems*, 2018.

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*, 2023.