CAN WE TRAIN A COMPUTER? TWO MECHANISMS FOR CONTENT ADDRESSING IN TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformers can be programmed by hand: with the right weights, one executes the instructions of a computer. Does training find such a solution? We ask this where the answer is checkable. For SUBLEQ, a one-instruction language that is Turing-complete given unbounded memory, we write a correct 4-layer transformer circuit by hand, then train a second transformer of the same architecture from scratch. Training succeeds: it executes an instruction exactly, and runs 16 steps on its own output on almost every program that keeps running. It does not rediscover our circuit. Both networks need the same primitive, attending to the memory cell whose address is stored elsewhere, and both build it, but they encode it differently. The hand-built circuit gives every position the same key direction and separates positions by distance along it, which forces attention scores spanning about 10^5 logits (Gaussian addressing); the trained network gives positions their own directions and spikes at the target with scores spanning tens (orthogonal-key addressing). Across 12 trained networks, over seeds, one-instruction variants, depths, and widths, none reaches the hand-built regime.

The difference is invisible to the tool one reaches for first. Weight-space mode connectivity reports that the two networks are unrelated, but it reports the same for two seeds of one recipe, so it cannot separate a different algorithm from a different seed; a probe read against the known circuit separates the two mechanisms cleanly. A construction is therefore an existence proof, not a prediction of what training builds, and a claim that two networks compute alike is worth no more than what the instrument behind it can tell apart.

1 INTRODUCTION

We increasingly obtain capability by *training* networks rather than by *programming* them. This makes a basic question urgent and hard: given two networks that behave the same, do they compute the same way inside? We ask it whenever we compare checkpoints, merge models, or audit whether a capability is present, and our default answer comes from weight-space tools: distance and mode connectivity. The difficulty is that we rarely know the correct answer to check the tools against. We remove it by studying a task we can also solve by hand.

When a transformer is trained from random initialization to execute an algorithm that admits a clean hand-designed solution, does gradient descent (GD) find that solution? Constructions show that transformers with chosen weights can simulate Turing machines (Pérez et al., 2021; Giannou et al., 2023), languages such as RASP let us program transformers by hand (Weiss et al., 2021; Lindner et al., 2023), and recent systems compile whole interpreters into transformer weights (Shaw et al., 2025; Tzamos & Percepta, 2026). Whether training reaches such solutions is a separate question, and constructions need not resemble trained networks (Wen et al., 2023).

What "same" means must be fixed first. Work on mechanistic mode connectivity reads a low-loss path in weight space, modulo architecture symmetries, as a shared mechanism (Lubana et al., 2023; Juneja et al., 2023); circuit analyses compare the functional roles of components (Elhage et al., 2021; Olsson et al., 2022). We use a concrete, checkable criterion: two networks share an algorithmic primitive if a read-only probe recovers the same intermediate signal at the same computational stage

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.

in both, with a positive control (the signal is present where the reference computes it) and a negative control (nothing is recovered where it is not computed). This is weaker than weight-space identity and than a full causal abstraction (Geiger et al., 2025) and stronger than input-output equivalence; we demonstrate *shared primitives* in this sense, not full algorithmic identity.

We use SUBLEQ, a one-instruction language that is Turing-complete given unbounded memory (Giannou et al., 2023, App. C) and a variant of the URISC computer (Mavaddat & Parhami, 1988). On a 32-cell, 8-bit machine a 4-layer transformer with d_{model}=256 can execute one step exactly. We build a hand-coded reference circuit ($\boldsymbol{\theta}_C$) and train a network of identical architecture from random initialization ($\boldsymbol{\theta}_T$); both reach 100% single-step accuracy, and, fed its own predictions, $\boldsymbol{\theta}_T$ matches the interpreter for 16 steps on 99.6% of the programs that have not halted. We then ask whether the two are the same algorithm under each lens, and trace the disagreement.

Two facts the findings rely on. (i) Executing a step requires *content addressing*: a head must attend to the memory cell whose index is stored elsewhere, a pointer dereference. (ii) Two of the ways a head can address matter here. In *Gaussian addressing* the keys of all positions are near-parallel and the attention score traces a smooth parabola in position; in *orthogonal-key addressing* the keys are much less aligned and the score is a sharp spike at one position. Both give the same argmax on the training distribution; they differ in key geometry, score shape, and score scale, which we measure on the input each head actually receives (Section 3).

Our findings.

1. **Weight-space mode connectivity fails, and the failure does not separate algorithms from seeds.** Five alignments leave the distance between $\boldsymbol{\theta}_C$ and $\boldsymbol{\theta}_T$ essentially unchanged, and every path we measured, including a trained bent line, collapses mid-path. Interpolation also collapses after an accuracy-preserving rescaling shrinks $\boldsymbol{\theta}_C$'s norm about 19-fold, and between two independently trained networks under four of the same alignment methods, so a barrier appears even when the addressing mechanism is the same (Section 5).
2. **The probe finds shared pointer-dereferencing primitives.** $\boldsymbol{\theta}_T$ exactly recovers two of $\boldsymbol{\theta}_C$'s canonical roles (fetch $\mathrm{mem}[a]$ and fetch $\mathrm{mem}[b]$), and two further heads attend at the operand-c position. Ablations show that these heads carry part of the computation, not all of it (Section 6).
3. **Two mechanisms, same primitive.** $\boldsymbol{\theta}_C$ uses Gaussian addressing: keys aligned after centering, scores exactly quadratic and spanning about 1.2×10^5 logits. $\boldsymbol{\theta}_T$ uses orthogonal-key addressing: keys aligned at 0.36 to 0.54, spike-shaped scores spanning 18 to 29 logits.
4. **The distinction holds in every network we trained in the SUBLEQ family.** Across 12 trained networks (4 seeds, 3 instruction sets, 2 architectures; 50 canonical heads) the trained heads sit between random initialization and the construction on the alignment axis and never reach the construction's regime; further architectures, a weight-decay-free twin, and two more machine variants keep the spike-shaped scores, and a sequence-reversal control also has non-quadratic scores (Section 8).
5. **Sparsity preserves and uncovers primitives.** Post-convergence L_1 pressure cuts the number of nonzero body weights about 84-fold while three of the four canonical heads stay sharp, and adds a primitive (broadcast from position 0) that the probe did not find before (Section 7).

What this adds up to. The two networks agree on every single-step input we tested, and they agree on the primitive they use to get there, yet they write that primitive in different codes. One of our two standard instruments, weight-space connectivity, cannot see the difference; it cannot even tell a different algorithm from a different random seed. The other, a probe read against a circuit we wrote ourselves, separates the two mechanisms cleanly. Thus, a verdict about similarity is worth only as much as the instrument's demonstrated ability to fail on a case whose answer we already know.

2 Related work

Mode connectivity. Low-loss curves connect independently trained networks (Garipov et al., 2018; Draxler et al., 2018), and networks trained from a shared early checkpoint are linearly connected (Frankle et al., 2020). Entezari et al. (2022) conjecture that most SGD solutions of sufficiently wide networks are linearly connected modulo permutations, with supporting evidence from Ainsworth et al. (2023); Zhang et al. (2025) extend alignment to rotations of attention heads, and Theus et al.

(2025) add orthogonal and semi-permutation symmetries and canonicalize each head to its query-key and output-value circuits, which brings the barrier between independently trained vision transformers and small GPT-2 models to zero or near zero, though one language corpus keeps a barrier. von Oswald et al. (2023) show that trained linear-attention weights match a construction after a scalar correction and are linearly connected to it in single-layer and recurrent linear-attention models. Lubana et al. (2023) argue that a barrier implies dissimilar mechanisms, where a mechanism is the set of input attributes a network is invariant to; Juneja et al. (2023) report that linearly disconnected clusters of fine-tuned classifiers correspond to different generalization strategies; we test that reading against a known circuit.

Mechanistic interpretability and hand-built references. Circuits have been reverse-engineered in trained transformers (Elhage et al., 2021; Olsson et al., 2022; Wang et al., 2023; Conmy et al., 2023). On modular arithmetic and group composition, networks sometimes implement the predicted algorithm (Nanda et al., 2023; Chughtai et al., 2023) and sometimes one of several, depending on hyperparameters (Zhong et al., 2023). Compiled transformers provide ground truth for interpretability (Lindner et al., 2023); ALTA compiles a SUBLEQ interpreter but does not train one (Shaw et al., 2025). The quadratic score behind Gaussian addressing is the addressing step of the hard-attention construction of Barceló et al. (2024), used for memory lookup in a recent in-transformer computer with hard attention (Tzamos & Percepta, 2026). Near-orthogonal position vectors let one head look up any of many positions in low dimension (Sanford et al., 2023), and associative memories built on near-orthogonal random embeddings are learned by gradient steps (Bietti et al., 2023); the trained keys in this paper are more aligned than random ones (Section 8). We compare a trained network with a same-architecture hand-built reference on a one-instruction computer (32 cells) and characterize which addressing geometry training reaches. Lottery tickets and sparse circuits (Frankle & Carbin, 2019; Goldowsky-Dill et al., 2023) contextualize our L_1 result.

Models that execute code. LSTMs evaluate short programs (Zaremba & Sutskever, 2014), language models execute with scratchpads (Nye et al., 2021), transformers pre-train on execution traces (Liu et al., 2023), benchmarks score output prediction (Gu et al., 2024), and ensembles of infinitely wide two-layer networks provably learn, with high probability, to execute a one-instruction computer's instruction exactly (Back de Luca et al., 2025). We ask how the execution is realized inside a trained transformer, against a circuit we can write by hand.

3 Background and Measurements

The SUBLEQ machine. Each instruction has operands (a, b, c) and executes $\mathrm{mem}[b] \leftarrow \mathrm{clamp}(\mathrm{mem}[b] - \mathrm{mem}[a])$, then $\mathrm{PC} \leftarrow c$ if $\mathrm{mem}[b] \leq 0$ and $\mathrm{PC} \leftarrow \mathrm{PC}{+}3$ otherwise. We use 32 cells of 8-bit signed values with $\mathrm{clamp}(x) = \max(-128, \min(127, x))$; a state is a sequence of $S = 33$ byte tokens, the PC followed by the 32 cells.

Mode connectivity and alignment. Networks $\boldsymbol{\theta}_A$ and $\boldsymbol{\theta}_B$ are mode-connected if accuracy stays high along $\boldsymbol{\theta}(\alpha) = (1-\alpha)\boldsymbol{\theta}_A + \alpha\boldsymbol{\theta}_B$, $\alpha \in [0, 1]$. Alignment first applies a function-preserving map to $\boldsymbol{\theta}_B$: permutations of heads, FFN units, and residual dimensions; per-head orthogonal maps $\mathbf{W}_q \to \mathbf{O}\mathbf{W}_q$, $\mathbf{W}_k \to \mathbf{O}\mathbf{W}_k$ with $\mathbf{O} \in \mathrm{O}(d_{\mathrm{head}})$ (Zhang et al., 2025); and per-head positive rescalings $\mathbf{W}_q \to \mathbf{D}\mathbf{W}_q$, $\mathbf{W}_k \to \mathbf{D}^{-1}\mathbf{W}_k$. Global rescaling $\boldsymbol{\theta} \mapsto c\boldsymbol{\theta}$ is not a symmetry, since it changes the function unless the readout compensates; ReLU layers also admit node-wise rescalings (Neyshabur et al., 2015).

Argmax-key probe. For head (ℓ, h) and state x, let $\hat{k}(x)$ be the key position that receives the largest attention weight from the query at position 0, the program-counter token, following the practice of reading a head's most-attended position as its prediction (Clark et al., 2019). For a canonical role r with state-dependent target $t_r(x)$, for example $\mathrm{mem}[\mathrm{pc}]{+}1$ for "fetch the cell addressed by operand a", a head is canonical if $\hat{k}$ tracks t_r with Pearson correlation $\rho \geq 0.99$ over 200 random states and its mean peak attention weight exceeds 0.5. Because Pearson correlation cannot distinguish targets that differ by a constant offset (Theorem 2), we assign the role by exact match, $\hat{k}(x) = t_r(x)$ on at least 99% of states. The probe is invariant to positive rescaling of queries and keys, so the magnitude gap between solutions does not affect it.

Two content-addressing mechanisms. A head content-addresses when, for each state, the score $\mathbf{q} \cdot \mathbf{k}_i$ is much larger at the target position $i = t(x)$ than at any other; Figure 1 sketches the two ways to achieve this. In *Gaussian addressing* the per-position keys $\{\mathbf{k}_i\}$ are near-parallel and the score is a parabola, $-s(i-t)^2 + c$ (Barceló et al., 2024), whose single maximum sharpens as the curvature s grows. In *orthogonal-key addressing* the keys are much less aligned and the score at the target stands out from a flat floor, so the score curve is a spike rather than a parabola. Both produce a sharp argmax on the training distribution; they differ in the geometry of the keys, in the shape of the score, and in its scale.

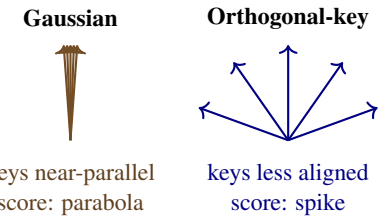

Figure 1: Two key geometries that produce sharp, content-addressed attention, and the score shapes they give.

Measuring the mechanism. We compute each head's keys $\mathbf{k}_i = \mathbf{W}_k \mathbf{x}_i$ and query $\mathbf{q} = \mathbf{W}_q \mathbf{x}_0$ from the input $\mathbf{x}$ the head actually receives, the residual stream at its layer, on 200 random states. Softmax ignores any key component shared by all positions, because such a component shifts every score by the same amount, so we remove the mean key $\bar{\mathbf{k}}$ over positions before measuring angles. The *key alignment* of a head is:

$$\mathrm{K}_{\mathrm{off}} = \mathbb{E}_x \Big[\frac{1}{S(S-1)} \sum_{i \neq j} \big| \cos(\mathbf{k}_i - \bar{\mathbf{k}}, \mathbf{k}_j - \bar{\mathbf{k}}) \big| \Big], \tag{1}$$

near 1 for near-parallel keys and near 0 for orthogonal ones; since $S = 33 > d_{\mathrm{head}} = 32$, the largest pairwise $|\cos|$ is at least $1/32$ (Welch, 1974; Strohmer & Heath, 2003). Without centering, the shared component inflates the alignment of trained keys; Sections K, L, N and O report both versions for every network. We summarize the score curve $i \mapsto \mathbf{q} \cdot \mathbf{k}_i / \sqrt{d_{\mathrm{head}}}$ by the R^2 of a quadratic fit in i and by its range $\max_i - \min_i$ in logits, both averaged over states. R^2 and the range do not depend on centering.

In plain terms, alignment near 1 means a head separates positions by degree along one shared direction, so the winner is the largest of many similar scores; alignment near 0 means it separates them by identity, so the winner is the one direction that matches the query. The first way of addressing has to spread its scores widely to make the peak stand out, and the second does not. That is why the two mechanisms can behave identically and still differ by orders of magnitude in scale.

4 Setup: two solutions on one architecture

Architecture. An encoder-only transformer with $L = 4$ layers, $d_{\mathrm{model}} = 256$, $H = 8$ heads ($d_{\mathrm{head}} = 32$), $d_{\mathrm{ff}} = 1024$, ReLU, no LayerNorm, and bidirectional attention: 3.29M parameters (Section B). Layers and heads are indexed from 0; L3H0 is head 0 of the last layer.

Hand-coded $\boldsymbol{\theta}_C$. We set every weight analytically from four ReLU-arithmetic identities (Section C): the quadratic score of Gaussian addressing, the integer step $\mathbb{1}[x > 0] = \mathrm{ReLU}(x) - \mathrm{ReLU}(x-1)$, a ReLU multiplexer for gated multiplication, and a hat indicator for cell-addressed writes. The first attention layer fetches operands a, b, c at pc$+1, +2, +3$; the second dereferences $\mathrm{mem}[a]$ and $\mathrm{mem}[b]$; the FFNs compute the difference and the branch predicate and route the results to cell b and to the PC. The circuit has 1,043 nonzero weights (180 in the body) and reaches full-state accuracy on 100,000 random states. It follows the looped-transformer framework of Giannou et al. (2023), whose own construction addresses memory with binary position codes rather than the quadratic score. At an output-head scale of $\alpha = 100$ its loss and gradient vanish to float32 precision, and Hessian-vector products along 31 sampled directions are non-negative.

Trained $\boldsymbol{\theta}_T$. Same architecture, random initialization, AdamW with dropout 0.1 (learning rate 3×10^{-4}, weight decay 0.01, batch 256) for 80,000 steps on (`state-before`, `state-after`) pairs drawn from random states and short program traces, with a curriculum on the number of instructions and $100\times$ loss weight on the positions a step changes (Section E). Training contains no multi-step rollouts. We analyze each seed's best evaluation checkpoint (seed 42: step 68K, full-state accuracy 1.000); seeds 42 to 45 are retained.

Table 1: Mode connectivity. Δ: relative change of $\|\boldsymbol{\theta}_C - \boldsymbol{\theta}_T^{\text{aligned}}\|_F$. Mid-path: minimum over interior α of the fraction of states whose changed positions are all correct; for the seed pair, the minimum full-state accuracy follows the slash (means over three evaluation sets of 512 states).

Method	Δ	$\boldsymbol{\theta}_C \leftrightarrow \boldsymbol{\theta}_T$	seed 42 $\leftrightarrow$ 43
Naive	–	0.000	0.074 / 0.000
Head/FFN perm.	$< 10^{-6}$	0.000	0.069 / 0.000
Residual perm.	$< 10^{-6}$	0.000	0.052 / 0.000
Iterative perm.	$< 10^{-6}$	0.000	0.030 / 0.000
$\mathrm{O}(d_{\text{head}})$ Procrustes	$\leq 10^{-4}$	–	0.081 / 0.000
Positive rescaling	$\leq 2 \cdot 10^{-3}$	0.000	–
Bent line	–	0.012	–

Multi-step execution. Feeding $\boldsymbol{\theta}_T$'s own prediction back as the next input for 16 steps (4 seeds; 5 draws of 400 starting states from the training distribution for each), the full 33-token state matches the interpreter on 99.6% of the trajectories still executing at step 16. Pooled over all trajectories the figure is 71%; almost all of the gap comes from programs that have halted, a frozen state that never appears in training and that the network reproduces in only 8% of cases (Section F).

The magnitude gap. $\|\boldsymbol{\theta}_C\|_F \approx 1.17{\times}10^5$ at $\alpha = 1$, against $\|\boldsymbol{\theta}_T\|_F \approx 93$. The gap is uneven: the output head carries $1.17{\times}10^5$ of $\boldsymbol{\theta}_C$'s norm (against 13 for $\boldsymbol{\theta}_T$), the embeddings 2,942 (against 17.2), and the four blocks 5,550 (against 90.5); see Table 3.

The two solutions differ less in what they do than in how loudly they do it: $\boldsymbol{\theta}_C$ carries about 1,300 times $\boldsymbol{\theta}_T$'s norm, nearly all of it in the output head. Any tool that compares parameters directly reports that difference first, whatever the two networks compute.

5 Weight-space mode connectivity fails

We align $\boldsymbol{\theta}_T$ to $\boldsymbol{\theta}_C$ with five methods and interpolate linearly, and we also train a bent-line path (Garipov et al., 2018) (Section G). In Table 1, the isometric alignments change the distance by at most 10^{-4}, positive rescaling by 0.2%, and every path we measured collapses mid-path, the bent line reaching only 1.2%. The isometric ceiling is structural. Permutations and orthogonal maps preserve norms, so $\|\boldsymbol{\theta}_C - g\boldsymbol{\theta}_T\|_F \geq \big|\|\boldsymbol{\theta}_C\|_F - \|\boldsymbol{\theta}_T\|_F\big| \geq 0.999\|\boldsymbol{\theta}_C\|_F$ for every such g (Theorem 3, Section A). Positive rescaling closes little because the gap sits in the output head and the embeddings, not in the per-head query and key weights where the rescaling acts.

The barrier does not separate algorithms from seeds. Two controls show why Table 1 is not evidence of different algorithms. (i) Scaling $\boldsymbol{\theta}_C$'s output-head weights and biases by a common $c > 0$ preserves every argmax, and hence 100% accuracy, while shrinking $\|\boldsymbol{\theta}_C\|_F$ about 19-fold; interpolation from this rescaled $\boldsymbol{\theta}_C$ to $\boldsymbol{\theta}_T$ still collapses to 0%. (ii) Two independently trained networks whose addressing heads fall in the same regime (Section 8), seeds 42 and 43, are not connected either. Under every alignment we applied to this pair (four methods and one combination), full-state accuracy falls to 0 mid-path and changed-position accuracy to at most 8.1%, and a weight-decay-free twin of seed 42 also collapses (about 1% full-state accuracy mid-path; Section H).

A collapse therefore appears between two networks with the same addressing mechanism, so under these alignments a collapse does not indicate a different algorithm. Barrier heights do differ: without alignment, the loss barrier is about 26,000 between $\boldsymbol{\theta}_C$ and $\boldsymbol{\theta}_T$, about 3,800 after rescaling $\boldsymbol{\theta}_C$'s head, and 4.7 between the seeds. The rescaled pair still differs 61-fold in body norm, so these heights do not separate magnitude from mechanism. All of this concerns the tested alignments; richer symmetry classes bring the barrier between independently trained transformers to zero or near zero in most settings tested by Theus et al. (2025). Under the tested alignments, then, a barrier is not evidence of dissimilar mechanisms, the reading of Lubana et al. (2023). We read the mechanism from the probe instead.

Read plainly, a barrier says that the straight line between two parameter vectors passes through bad networks. It does not say that the two networks compute differently, because the same barrier appears between two networks that compute the same way. An instrument that cannot separate a different algorithm from a different random seed cannot answer the question we are asking with it.

Linear CKA fails for a related reason. Linear CKA (Kornblith et al., 2019) between $\boldsymbol{\theta}_C$ and the interpolants stays at or above 0.99 in layers 1 to 3, suggesting a shared body that does not exist. CKA is invariant to isotropic rescaling, so the larger network's activations dominate the interpolants over

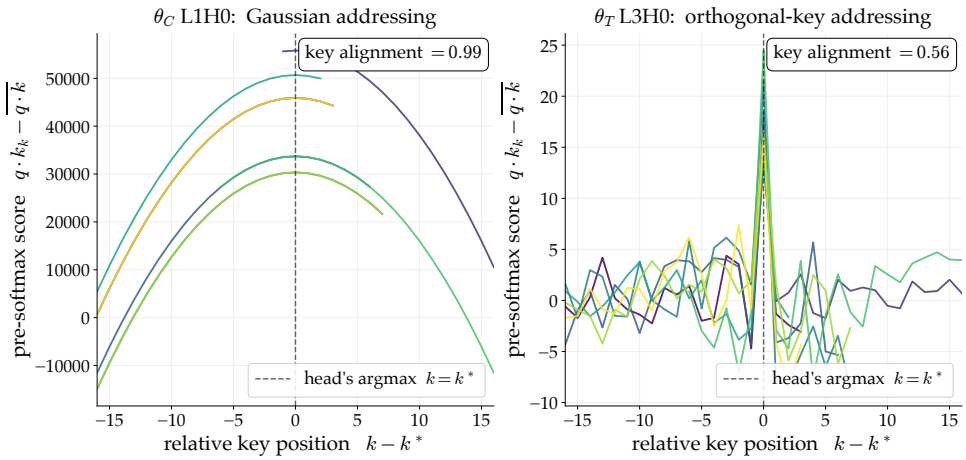

Figure 2: Pre-softmax score curves of matched fetch $\mathrm{mem}[a]$ heads, $\boldsymbol{\theta}_C$ L1H0 (left) and $\boldsymbol{\theta}_T$ L3H0 (right), for 8 states with different program counters, each centered on its own argmax k^*. Left: smooth parabolas, the quadratic identity of Equation (2). Right: sharp spikes over a flat floor. The vertical scales differ by more than $2{,}000\times$; key alignment on this sample of real layer inputs is 0.99 and 0.56.

most of the path (Theorem 4, Section I). The caution runs in both directions: a similarity score near 1 can report which network is louder rather than what either one computes.

6 Mechanistic probe: orthogonal-key addressing

Which roles does $\boldsymbol{\theta}_T$ allocate to attention? Probing the 32 heads of $\boldsymbol{\theta}_T$ (seed 42) on 200 random states, four heads pass (Table 2). L3H0 and L3H3 exactly match $\boldsymbol{\theta}_C$'s fetch $\mathrm{mem}[a]$ and fetch $\mathrm{mem}[b]$ heads, two-level pointer dereferences. L2H4 and L3H7 attend at $\mathrm{pc}{+}3$, the position of operand c, which doubles as the default next PC; Pearson correlation alone labels them operand-a fetches, because $\mathrm{pc}{+}1$, $\mathrm{pc}{+}2$, and $\mathrm{pc}{+}3$ move together. The remaining roles of $\boldsymbol{\theta}_C$ are diffuse or absent at convergence, and Section 7 recovers one. As a negative control, no head of eight randomly initialized networks with the same architecture passes the probe: their mean peak attention weight is at most 0.03, the uniform value.

Table 2: The four canonical heads of $\boldsymbol{\theta}_T$ (seed 42), with a matched $\boldsymbol{\theta}_C$ head. ρ: Pearson correlation of the argmax with the target; peak: mean peak attention weight; $\mathrm{K}_{\mathrm{off}}$: key alignment (1); range: score range in logits; all on the heads' real inputs.

Head	Target	ρ	peak	$\mathrm{K}_{\mathrm{off}}$	R^2	range
L3H0	$\mathrm{mem}[\mathrm{pc}]{+}1$	1.00	1.00	0.54	0.10	23.7
L3H3	$\mathrm{mem}[\mathrm{pc}{+}1]{+}1$	1.00	1.00	0.48	0.12	29.2
L2H4	$\mathrm{pc}{+}3$	1.00	1.00	0.36	0.10	18.4
L3H7	$\mathrm{pc}{+}3$	1.00	0.95	0.37	0.18	23.6
$\boldsymbol{\theta}_C$ L1H0	$\mathrm{mem}[\mathrm{pc}]{+}1$	1.00	1.00	0.99	1.00	$1.4{\times}10^5$

The mechanism. Figure 2 contrasts the score curves. $\boldsymbol{\theta}_C$ produces smooth bowls on which the quadratic identity fits exactly. $\boldsymbol{\theta}_T$ produces a sharp spike at the target over a flat floor, despite identical argmax behavior. The keys differ accordingly: after removing the shared component, $\boldsymbol{\theta}_C$'s keys are near-parallel (alignment 0.99), while $\boldsymbol{\theta}_T$'s are far less aligned (0.36 to 0.54), though more aligned than at initialization, and $\boldsymbol{\theta}_C$'s scores span about 10^5 logits against 18 to 29 for $\boldsymbol{\theta}_T$. We call the trained mechanism *orthogonal-key addressing* to contrast it with the construction's near-parallel keys; trained keys are not orthogonal, and a hand construction reaches the orthogonal extreme (Section 8).

Both networks answer the same question, which memory cell the pointer names, and on the states we probe their heads point at the same cell. They differ in how the answer is written down. The construction gives every position the same key direction and separates positions by how far along it they sit, so the target wins by a margin that grows with the square of the distance and the scores must be enormous. The trained network gives positions their own directions and lets the query match one of them, so a margin of a few logits suffices. The primitive is shared; the code for it is not.

Mechanism ablations. Zeroing the position embedding collapses the canonical heads of both networks, so both are positional. Zeroing the token embedding leaves $\boldsymbol{\theta}_C$'s heads sharp but aimed at the wrong target, since Gaussian addressing reads only positions, whereas $\boldsymbol{\theta}_T$'s heads, which sit two to three layers deeper, collapse, so their keys or queries depend on token-derived features computed upstream (Section J).

A causal check. The probe is correlational, so we remove heads and measure the fraction of changed positions predicted correctly; at least 31 of the 33 tokens are copied, so a score over all positions would stay high even when the step is wrong. Averaged over five draws of 3,000 states, removing all 32 heads drops this accuracy from 1.00 to 0.21. Keeping only the four canonical heads gives 0.45, above four heads kept at random (0.29) but far from 1.00. Removing the non-canonical head L2H0 alone lowers accuracy to 0.50, more than removing any canonical head, and removing L3H7 leaves accuracy at 1.00. The probe identifies a real sub-circuit of the addressing, not the whole computation (Section J).

Both regimes fit in the architecture, at very different scales. Our Gaussian construction uses a position embedding of norm $\Theta(S^{5/2})$, where a unit-norm frame suffices for orthogonal-key addressing (Theorem 5, Section A). Where that norm sits depends on the parameterization, but the score range does not: a score that is exactly quadratic in position, with its neighbors a gap s below the peak, spans at least $s((S-1)/2)^2$ logits (at least 1,354 at $S = 33$ for the smallest gap that puts 99% of the attention on the target; Remark 1), while an equiangular frame spans only its gap. No trained network we examine reaches the Gaussian regime (Section 8). The architecture is therefore not what picks the mechanism. Both codes fit in the same 32-dimensional heads; they differ in what they cost, and training never pays the price of the wide score range that exact quadratic addressing demands.

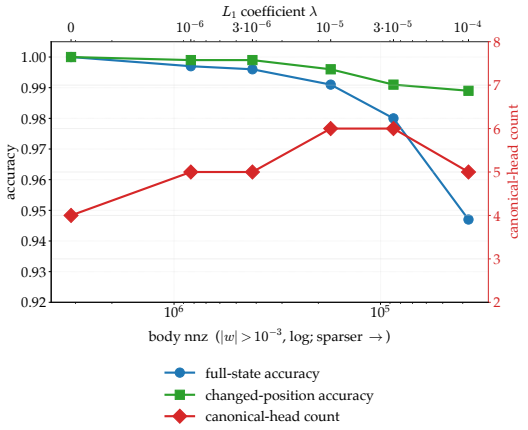

Figure 3: Post-convergence L_1 sweep on $\boldsymbol{\theta}_T$'s body: accuracy (left axis) and canonical-head count (right axis, red) against the number of nonzero body weights. At the sparsest point nonzero body weights have fallen about 84-fold, changed-position accuracy is 0.989, and full-state accuracy is 0.947.

7 Sparsity preserves and uncovers primitives

We apply L_1 pressure to $\boldsymbol{\theta}_T$'s body (output head and embeddings excluded) for 30,000 further steps, sweeping $\lambda \in \{10^{-6}, 3{\times}10^{-6}, 10^{-5}, 3{\times}10^{-5}, 10^{-4}\}$ with a linear ramp over the first 15K steps, and re-run the probe at each λ (Figure 3). At the sparsest setting nonzero body weights fall about 84-fold. Three of the four canonical heads (L2H4, L3H0, L3H3) keep their roles with peak weight at least 0.93 at every λ; L3H7, the weakest (peak weight at most 0.75), loses its role at two settings, including the sparsest. The canonical heads' score curves stay spike-shaped ($R^2 \leq 0.25$): L_1 does not push $\boldsymbol{\theta}_T$ toward Gaussian addressing. This is a cheap test of whether the trained solution is a cluttered version of the construction, and it fails: pressure that removes about 99% of the body weights leaves the same spike-shaped addressing in place.

From $\lambda = 3{\times}10^{-6}$ on, a broadcast-from-position-0 head appears at L3H1 (peak weight 0.74 to 0.97), a role for which no head of the unsparsified network passes the probe, and a second fetch $\mathrm{mem}[b]$ head appears at L3H4 (peak weight 1.00 at every λ) alongside L3H3. Three L_1 runs at $\lambda = 10^{-5}$ that start from the seed-42 network with different random seeds place the same six roles in the same six head slots, with peak weights within 0.08 of each other except for L3H7 (0.62 to 0.77; Section M). L_1 after convergence thus works as a probe of last resort, lifting out roles that are diffuse at convergence.

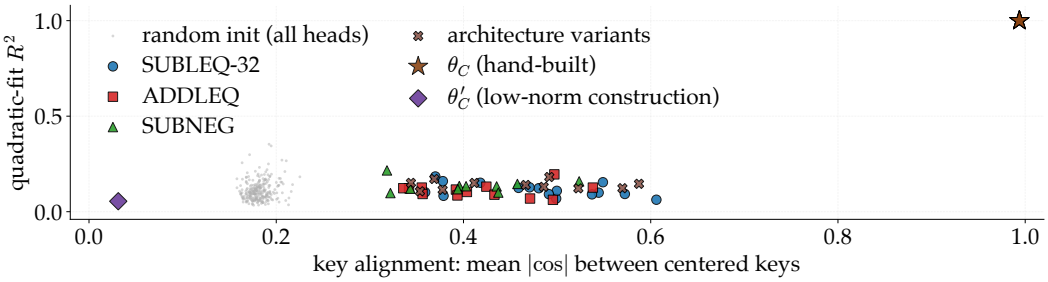

Figure 4: Mechanism map on the heads' real inputs. Horizontal: key alignment (1); vertical: quadratic-fit R^2 of the score curve. Colored markers: the 50 canonical heads of 12 trained networks. Gray: all heads of 8 randomly initialized networks with $\boldsymbol{\theta}_T$'s architecture. Star: $\boldsymbol{\theta}_C$'s five addressing heads. Diamond: $\boldsymbol{\theta}'_C$, a low-norm hand construction with an orthogonal Fourier frame (Section D).

8 Universality across seeds, instruction sets, and architectures

We replicate the probe and the measurements across 12 trained networks: 4 seeds of SUBLEQ, 4 of ADDLEQ, and 2 of SUBNEG, variants that change the arithmetic or the branch condition (Section L), plus a 6-layer and a width-128 network. Figure 4 places each of their 50 canonical heads by key alignment and R^2, measured on the heads' real inputs; per-head values are in Section K.

Three regimes. $\boldsymbol{\theta}_C$'s five addressing heads sit alone in the upper right (alignment 0.99, $R^2 = 1.00$). The trained heads form one cluster, whatever the instruction set or architecture: alignment 0.32 to 0.61 (median 0.43), R^2 from 0.06 to 0.22, and score ranges of 9 to 42 logits against about 1.2×10^5 for $\boldsymbol{\theta}_C$. All 256 heads of the randomly initialized networks sit at alignment 0.16 to 0.23, below every trained head: training makes keys more aligned than at initialization, but nowhere near as aligned as in the construction. A hand-built addressing head with an orthogonal Fourier frame and $\mathcal{O}(1)$ weights, $\boldsymbol{\theta}'_C$, fetches exactly and sits at the orthogonal extreme (alignment 0.03, $R^2 = 0.05$), so orthogonal-key addressing is itself hand-buildable at low norm.

What the map shows is a band rather than a point. Every network we trained sits between random initialization and the construction, closer to the random end, and none of the things we varied, seed, instruction set, depth or width, moves a network out of that band. Within this family, training does not wander over the solutions that work: it lands in one region of them every time.

Beyond the twelve networks. Three further architectures (Pre-LayerNorm with ReLU, GELU without LayerNorm, and 6 layers with LayerNorm), a weight-decay-free twin, a two-operand accumulator variant (Subleq2), and a 16-cell SUBLEQ keep spike-shaped scores, with $R^2 \leq 0.36$ and score ranges below 30 logits on every head the probe selects or that attends sharply. LayerNorm raises the key alignment of sharply attending heads (medians 0.58 and 0.59 in the two LayerNorm networks, maximum 0.80) without approaching the construction (Section N). A network trained on sequence reversal also has non-quadratic scores ($R^2 < 0.28$ over all heads). Its median alignment over all heads is higher than $\boldsymbol{\theta}_T$'s (0.62 against 0.52), while its one sharply attending head sits at 0.29, so a single control network does not show whether the alignment band depends on the task (Section O).

9 Conclusion

We compared a hand-coded SUBLEQ-executing transformer with networks of the same architecture trained from random initialization, and asked when two solutions count as the same algorithm. Weight-space alignment fails to bridge them, but under the same tools it also fails between two trained networks, so the barrier does not answer the question.

Mechanistic probing gives a sharper answer: the trained networks share the construction's pointer-dereferencing primitives and realize them through a different mechanism, less aligned keys with spike-shaped scores at small magnitude instead of near-parallel keys with quadratic scores at large magnitude. In every network we trained in this family, the addressing heads use orthogonal-key addressing, with scores spanning tens of logits, not the construction's Gaussian addressing, with scores spanning about 10^5 logits. The methodological lesson is that weight-space connectivity and

mechanistic probing answer different questions, and that a known circuit is what lets us tell which answer to trust.

In practice this means calibrating the instrument before believing its verdict. A barrier between two networks is evidence about coordinates, not about computation, and the cheapest way to see that is to run the same test on a pair whose answer is already known, such as two seeds of one recipe. It also means that a construction is a hypothesis about how a task can be done, not a prediction of how training will do it: ours is correct, economical to describe, and not what gradient descent builds.

Limitations. *Scope.* One task family (single steps of one-instruction machines) and small transformers; whether the same mechanisms appear in other algorithmic tasks or at scale is open, and one control task does not show whether the alignment band transfers (Section O). *Probe vocabulary.* Our targets are $\boldsymbol{\theta}_C$'s roles, so primitives that $\boldsymbol{\theta}_T$ uses and $\boldsymbol{\theta}_C$ does not are invisible to the probe. The causal check shows that such computation exists (head L2H0), and the FFN is load-bearing: removing all four FFN blocks leaves 0.18 of the changed positions correct, with blocks 1 (0.26 when removed alone) and 3 (0.50) mattering most. Our claims concern the geometry of the addressing heads the probe identifies. Ablation results can change with the ablation method, zero, mean, or resampling (Miller et al., 2024). *Weight-space coverage.* We test permutations, per-head rotations, positive rescaling, and curved paths. Richer symmetry classes (Theus et al., 2025) may connect some of the pairs we study; our controls show only that the tested tools do not separate algorithm from seed. *Measurement.* Key alignment depends on removing the component shared by all keys; without it, trained and constructed keys look similarly aligned (Section K). The quadratic-fit R^2 and the score range do not depend on this choice and separate the two mechanisms on their own. *Statistics.* Four seeds for SUBLEQ and ADDLEQ, two for SUBNEG, and one for each architectural variant; the ranges in Figure 4 are empirical ranges over heads, not confidence intervals.

AI USE STATEMENT

We used large language model assistants to write, test, and independently re-implement analysis code; to run experiments and check their results; to search the literature and check references against their sources; to produce figures; and to draft and edit text. The authors chose the research question, approved the experimental design and every claim, and checked AI-assisted code, numbers, and citations against results files and sources. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The supplementary material contains the code that trains every network, builds the hand-coded circuit, and runs the alignment, probing, ablation, sparsity, and measurement experiments, with unit tests for the evaluation metrics and the alignment methods. Architecture and training hyperparameters are in Section 4 and Section E; the hand-coded construction is specified in Section C; proofs are in Section A; measurement definitions are in Section 3 and Section K.

REFERENCES

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git Re-Basin: Merging models modulo permutation symmetries. In *International Conference on Learning Representations (ICLR)*, 2023.

Artur Back de Luca, George Giapitzakis, and Kimon Fountoulakis. Learning to add, multiply, and execute algorithmic instructions exactly with neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 87469–87513, 2025.

Pablo Barceló, Alexander Kozachinskiy, Anthony Widjaja Lin, and Vladimir Podolskii. Logical languages accepted by transformer encoders with hard attention. In *International Conference on Learning Representations (ICLR)*, 2024.

Alberto Bietti, Vivien Cabannes, Diane Bouchacourt, Hervé Jégou, and Léon Bottou. Birth of a transformer: A memory viewpoint. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 1560–1588, 2023.

Bilal Chughtai, Lawrence Chan, and Neel Nanda. A toy model of universality: Reverse engineering how networks learn group operations. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 6243–6267, 2023.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? An analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 276–286, 2019.

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 (NeurIPS)*, volume 36, pp. 16318–16352, 2023.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A. Hamprecht. Essentially no barriers in neural network energy landscape. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1309–1318, 2018.

Benjamin L. Edelman, Surbhi Goel, Sham Kakade, and Cyril Zhang. Inductive biases and variable creation in self-attention mechanisms. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5793–5831, 2022.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations (ICLR)*, 2022.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3259–3269, 2020.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of DNNs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah Goodman, Christopher Potts, and Thomas Icard. Causal abstraction: A theoretical foundation for mechanistic interpretability. *Journal of Machine Learning Research*, 26(83):1–64, 2025.

Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11398–11442, 2023.

William F. Gilreath and Phillip A. Laplante. *Computer Architecture: A Minimalist Perspective*. Springer US, 2003. ISBN 978-1-4613-4980-8.

Nicholas Goldowsky-Dill, Chris MacLeod, Lucas Sato, and Aryaman Arora. Localizing model behavior with path patching. *arXiv preprint arXiv:2304.05969*, 2023.

Alex Gu, Baptiste Rozière, Hugh Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida I. Wang. CRUXEval: A benchmark for code reasoning, understanding and execution. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 16568–16621, 2024.

Jeevesh Juneja, Rachit Bansal, Kyunghyun Cho, João Sedoc, and Naomi Saphra. Linear connectivity reveals generalization strategies. In *International Conference on Learning Representations (ICLR)*, 2023.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. 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.

David Lindner, János Kramár, Sebastian Farquhar, Matthew Rahtz, Thomas McGrath, and Vladimir Mikulik. Tracr: Compiled transformers as a laboratory for interpretability. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 37876–37899, 2023.

Chenxiao Liu, Shuai Lu, Weizhu Chen, Daxin Jiang, Alexey Svyatkovskiy, Shengyu Fu, Neel Sundaresan, and Nan Duan. Code execution with pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 4984–4999, 2023.

Ekdeep Singh Lubana, Eric J. Bigelow, Robert P. Dick, David Krueger, and Hidenori Tanaka. Mechanistic mode connectivity. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22965–23004, 2023.

Farhad Mavaddat and Behrooz Parhami. URISC: The ultimate reduced instruction set computer. *International Journal of Electrical Engineering & Education*, 25(4):327–334, 1988.

Oleg Mazonka and Alex Kolodin. A simple multi-processor computer based on Subleq. *arXiv preprint arXiv:1106.2593*, 2011.

Joseph Miller, Bilal Chughtai, and William Saunders. Transformer circuit evaluation metrics are not robust. In *First Conference on Language Modeling (COLM)*, 2024. Preprint: arXiv:2407.08734, "Transformer Circuit Faithfulness Metrics are not Robust".

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

Behnam Neyshabur, Ruslan Salakhutdinov, and Nathan Srebro. Path-SGD: Path-normalized optimization in deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, 2015.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Jorge Pérez, Pablo Barceló, and Javier Marinković. Attention is Turing-complete. *Journal of Machine Learning Research*, 22(75):1–35, 2021.

Clayton Sanford, Daniel Hsu, and Matus Telgarsky. Representational strengths and limitations of transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 36677–36707, 2023.

Peter Shaw, James Cohan, Jacob Eisenstein, Kenton Lee, Jonathan Berant, and Kristina Toutanova. ALTA: Compiler-based analysis of transformers. *Transactions on Machine Learning Research*, 2025.

Thomas Strohmer and Robert W. Heath, Jr. Grassmannian frames with applications to coding and communication. *Applied and Computational Harmonic Analysis*, 14(3):257–275, 2003.

Alexander Theus, Alessandro Cabodi, Sotiris Anagnostidis, Antonio Orvieto, Sidak Pal Singh, and Valentina Boeva. Generalized linear mode connectivity for transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 150971–150999, 2025.

Christos Tzamos and Percepta. Can LLMs be computers? Executing programs inside transformers with exponentially faster inference. Percepta blog, March 2026. URL https://www.percepta.ai/blog/can-llms-be-computers.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 35151–35174, 2023.

Kevin Ro 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.

Gail Weiss, Yoav Goldberg, and Eran Yahav. Thinking like transformers. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11080–11090, 2021.

Lloyd R. Welch. Lower bounds on the maximum cross correlation of signals (corresp.). *IEEE Transactions on Information Theory*, 20(3):397–399, 1974. doi: 10.1109/TIT.1974.1055219.

Kaiyue Wen, Yuchen Li, Bingbin Liu, and Andrej Risteski. Transformers are uninterpretable with myopic methods: a case study with bounded Dyck grammars. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 38723–38766, 2023.

Wojciech Zaremba and Ilya Sutskever. Learning to execute. *arXiv preprint arXiv:1410.4615*, 2014.

Binchi Zhang, Zaiyi Zheng, Zhengzhang Chen, and Jundong Li. Beyond the permutation symmetry of transformers: The role of rotation for model fusion. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 77090–77106, 2025.

Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 27223–27250, 2023.