REPRESENTATION WITHOUT CAPABILITY, THEN CAPABILITY WITHOUT ABSTRACTION: PARITY IN GPT-2

Anonymous authors
Paper under double-blind review

ABSTRACT

Probing has become a standard instrument for characterizing what a language model represents, and it underlies a family of intervention methods, among them directional ablation and activation steering, that manipulate behavior aling probe-derived directions. These methods operationalize a rarely tested assumption: that the direction along which a concept is linearly decodable (representation) is the direction the model reads and uses (capability). Evaluating this presupposition is non-trivial for semantically rich concepts because a robust test requires an independent ground truth about what the model actually computes. We therefore construct a setting in which that ground truth is available and the rule the model implements can be identified from behaviour. We use the pretrained GPT-2 small model, whose representations were formed by ordinary language modeling, a *natural* setting. We observed that pretrained GPT-2 cannot perform parity detection above chance in any prompt format we administered, while a hyperplane in its embedding space separates numerals by parity at 0.994 on held-out tokens: *Representation Without Capability*. This representation the model cannot act upon is selective, as divisibility by three performs at 0.687. We then fine-tune it on a three-operand arithmetic verification task adapted from an intracranial-EEG protocol for human subjects (Kalinova et al., 2026): $N_1 \pm N_2 \pm N_3 = N_4$ with false candidates at $\mathrm{correct} \pm 1$. We fine-tune on both the *easy* and *difficult* variants, which differ in operand and result digit-width: single-digit operands with results in $[0, 9]$, versus a two-digit N_1 with results in $[11, 98]$. Arithmetic is not an intrinsic capability at this model scale and remains heuristic at far larger ones. We observed GPT-2 constructing a chained-parity solution: the fine-tuned model attains 97.9–100% accuracy by comparing the candidate result's parity against $\mathrm{parity}(N_1) \oplus \mathrm{parity}(N_2) \oplus \mathrm{parity}(N_3)$. Neither constituent of this solution originates in fine-tuning: parity is already separable in the embeddings, and the pretrained model already computes XOR at 0.91–1.00 over other feature-disjoint pairs. Fine-tuning enables the connection between parity identity and XOR, turning on the chained-parity detection capacity that solves the task. But *the Capability it creates is not Abstract*: behavioral tests suggest that the model stores a parity bit per token-position pair but does not compute parity from the numeral. A token never seen in N_1 contributes always *even*, and an odd three-digit and the word "cat" become indistinguishable, both at 0.985 and mean $|\Delta_{\mathrm{logit}}|$ 8.25. XOR composition was inherited from pretraining. Causal ablation confirms parity is used upon fine-tuning in the arabic easy task, as removing it costs 6.5 accuracy points at rank 40 against 0.2 for an equally decodable property of the same token and 0.0 for random directions. In the difficult condition, however, parity and the matched control damage the model equally at rank 80 (0.9554 vs 0.9582, against a 0.9826 baseline) while random directions do not (0.9813), suggesting that parity over a ninety-token operand occupies far higher rank than over a ten-token one. Interestingly, learnings from the easy condition show that removing parity leaves the model confident and wrong while random ablation leaves it hesitant and right. The same signature appears when we break a token slot's surface format, an intervention applied at the input that touches no activations. Making parity usable also dispersed it: an identical 40-dimensional cut reduces its decodability to 0.4916 in the pretrained model—chance—but only to 0.8183 after fine-tuning. Our findings have practical implications for questioning how the representation-capability rela-

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.

tion is tested. Standard procedures of directional ablation, computing the direction on a forward pass and projecting it out finds nothing: accuracy stays at 1.000, failing because every block writes parity back into the residual stream along fresh directions. Altogether, this paper suggests that representation does not entail capability, and capability does not guarantee abstraction. These observations have consequences for the literatures of probing and benchmarking, and for current views on concept erasure and approaches for unlearning. Beyond, we connect our findings with safety discussions, since a procedure that activates dormant knowledge is powerful when intended and hazardous when it arises as an unanticipated side effect of postraining.

1 Introduction and related work

Language transformer models acquire an enormous body of declarative knowledge about the world over the course of pretraining on text corpora. This knowledge is stored as geometric structure in their internal high-dimensional space (Mikolov et al., 2013; Elhage et al., 2022). Within this structure, it has been suggested that many features are linearly decodable, this is to say, individual concepts could correspond to linear directions along which activations can be projected to recover the concept's value (Park et al., 2024; Nanda et al., 2023; Marks & Tegmark, 2024). This linear representation hypothesis set the basis for the probing methodology that has become a standard instrument for investigating what a model knows (Alain & Bengio, 2017; Belinkov, 2022). Moreover, it has recently been extended into a set of intervention techniques that manipulate model behavior by directly editing the geometry of encoded concepts, among which activation steering (Turner et al., 2023; Rimsky et al., 2024), representation engineering (Zou et al., 2023), and directional ablation (Arditi et al., 2024).

A varied set of linear probes have been constructed, each recovering a different property from model activations (Alain & Bengio, 2017; Nanda et al., 2023; Marks & Tegmark, 2024) but whether the linear structure used for recovery is the form in which the model's own computation accesses and uses specific features, remains a controversial hypothesis (Park et al., 2024; Elhage et al., 2022). Elhage et al. (2022) formulates the premise that linear accessibility is what allows the next layer of a model to use a feature, but whether linear probes measure information presence or ease of extraction remains an open question in the field (Kahana et al., 2025). Decodability of a concept is a joint property of the model and the probe, risking that a sufficiently expressive probe could recover structure the model does not encode as a feature. Mechanistic interpretability has responded by distinguishing what a model represents from what it computes by introducing control tasks (Hewitt & Liang, 2019; Belinkov, 2022): amnesic probing and activation patching, and through the development of the field's causal toolkit, reflecting the recognition that decodability alone is insufficient evidence of use (Hewitt & Liang, 2019; Belinkov, 2022). Despite this recognition, several widely used intervention methods nonetheless embed the link operationally: directional ablation and activation steering attribute a behavioral change to the concept a direction encodes, which requires the identified direction to be the one the model reads and uses (Arditi et al., 2024).

Moving beyond the use of linear probes, Engels et al. (2025) provided a clear existence proof for non-linear multi-dimensional representations that cannot be reduced to linear directions, and Garg et al. (2026) have shown that a linear representation does not entail linear accessibility, establishing a gap between the two properties. Empirical evidence for a dissociation between representation and use has recently been reported as well in agentic settings. Wang et al. (2026) introduced the term *representation–behavior gap* for tool-calling agents, reporting failure to act despite a probe AUC of 99%. Related work describes the same dissociation in tool use as a *knowing–doing gap*, and locates its mechanism in later layers, where cognitive representations become orthogonally misaligned with execution intent (Cheng et al., 2026). These are cases in which a model internally registers the relevant information but fails to translate it into reliable execution. The representations involved in this cases, however, are semantically rich and difficult to specific precisely, so claims about whether the model holds them inherit the vagueness of the concept itself. The dissociation between representation and use has been therefore demonstrated for execution intent, but not for a property simple enough whose presence or absence admits exact experimental testing.

Concept representation and use, moreover, are not guaranteed to be stable across different types of tasks or upon fine tuning. Recent mechanistic work indicates that fine-tuning frequently repurposes pretrained structure rather than constructing new machinery, even by rewiring a direction with a given purpose into a new circuit and different purpose Prakash et al. (2024); Venhoff et al. (2025). Olah & Team (2024) propose model diffing as a dedicated methodology for this class of question and state that conclusive evidence on the repurposing hypothesis remains absent. Establishing repurposing requires three measurements on the same property: what the pretrained model encodes, what the fine-tuned model computes, and the causal relation between them. Semantically rich concepts do not support all three measurements with equal precision. A property with a formal definition, an unambiguous label, a direct behavioral test, and a task whose complete solution space can be derived analytically supports all three.

In this paper, we fine-tune GPT2 small pretrained on an arithmetic verification task adapted from an intracranial EEG study of human arithmetic processing (Kalinova et al., 2026). The task presents a three-operand equation with a candidate result and requires a binary verdict of whether the result is true or false. The task is small enough to be characterized analytically, so the complete set of feature sets sufficient to solve it can be enumerated in closed form. The task is a direct behavioral test with an unambiguous label requiring a solution using a property with a formal definition and an analytically enumerable solution space. Given the stimulus construction of Kalinova's design, with the rule that incorrect examples differ from the correct one by $\mathrm{correct} \pm 1$, upon fine tuning the property the model chose to use is parity (as $\mathrm{correct} \pm 1$ inverts parity), one that satisfies all requirements to disentangle representation and use, and for investigating the dynamics of repurposing. We establish what the pretrained model encodes by probing, what the fine tuned model computes behaviorally using counterfactual sets, and we test the causal relation between them by directional ablation at increasing rank. We show that GPT2 small pretrained prior to fine tuning, represents parity as a linearly decodable direction from the token embeddings at 0.994, and this direction is carried to the decision point by layer 1-2, distributed redundantly enough that more than 80 of the 768 directions must be resolved before behavior changes. However, the pretrained model cannot use this represented feature at all, answering at chance in every prompt format we tested. Fine-tuning in Kalinova's task does not build the missing capability to solve it, but it connects two structures that were already present, parity identity and XOR logic, to develop a strategy to solve every instance of the task including held out examples. Evidence provided suggests that representation is not necessarily linked with knowledge accessibility, awareness or usage capability using a formally defined and simple property, showing that abundance in linear representation and transmission of the representation along the residual stream guarantees nothing about usability. Linear probes, current most used method for LLM knowledge representation, measure information presence, but not information usability or accessibility. Moreover, in our exercise highly accurate capability, even generalizing to held-out equations and unseen magnitudes, did not turn into abstraction to unseen tokens or surface formats, opening questions about what behavioral generalization certifies about underlying capabilities. We here highlight our empirical and conceptual findings about three key levels of LLM intelligence that require independent measurements: representation, capability and abstraction. Our findings also bear implications for safety, since a represented concept can remain dormant until fine-tuning connects it to a downstream computation, a procedure that is powerful when intended and hazardous when it arises as an unanticipated side effect of post-training.

2 SETUP AND DEFINITIONS

Task and data. Each item is an equation $N_1\, O_1\, N_2\, O_2\, N_3 = N_4$ with $O_i \in \{+, -\}$ and the label TRUE when N_4 equals the computed result. Following Kalinova et al. (2026), a FALSE item presents $N_4 = \mathrm{correct} \pm 1$. The *easy* condition uses single-digit operands and keeps intermediate and final results in $[0, 9]$; the ***difficult*** condition uses a two-digit N_1, single-digit N_2 and N_3, a decade crossing, and results in $[11, 98]$. Each condition contains 2,142 training and 238 validation items, balanced between labels and split at the equation level, so the TRUE and FALSE versions of an equation always share a split. Every equation in both conditions tokenizes to exactly seven GPT-2 tokens with a single-token answer, and the verdict is read at the final token in both conditions. The balancing of training data has a coverage consequence on difficult condition: it only covers a 4.9% of its equation space and include 89 of the 90 possible values of N_1.

Models and training. We fine-tune pretrained GPT-2 small (124M parameters) on each condition with weight decay 1.0 and learning rate 10^{-3}, the regime of the grokking literature. Both runs alternate between grokking and collapse for the whole of training, with 30 cycles for the easy run and 13 for the difficult run (Appendix A.1). We analyze the checkpoint at the peak of the first grokking cycle, step 1,500 for the easy model (validation accuracy 1.000) and step 8,500 for the difficult model (0.979). A third model, trained on the easy condition at weight decay 0.01 (the *low-decay model*), serves as a comparison.

Readout and metrics. The verdict is the sign of Δ_{logit}, the difference between the TRUE and FALSE verdict logits at the readout position, and $|\Delta_{\text{logit}}|$ measures confidence. Intervals are 95% Wilson intervals. The ± 1 perturbations distribute asymmetrically over answer values, so an oracle that sees only the presented answer scores 0.618 on easy and 0.681 on difficult validation; these values define the floor for evidence that a model uses the equation.

Probing. Probes are L2-regularized logistic regressions scored on held-out folds (five folds for embeddings, three for residual activations), with features standardized within the probed set. A shuffled-label probe measures empirical chance. A probe for $A \oplus B$ detects the combination only when the model has computed it, since XOR is linearly inseparable in A and B. We restrict such tests to *feature-disjoint* pairs, in which A and B depend on distinct variables, since a rich encoding of one shared variable decodes any function of that variable.

The chained-parity rule. Let $p(\cdot)$ denote parity. Addition and subtraction act identically on parity, so $p(N_1 \pm N_2 \pm N_3) = p(N_1) \oplus p(N_2) \oplus p(N_3)$, and a shift by ± 1 always inverts parity. The rule

$$\hat{y} = \text{TRUE} \iff p(N_4) = p(N_1) \oplus p(N_2) \oplus p(N_3) \tag{1}$$

therefore labels every item of the stimulus set correctly, and it scores 1.000 on the training, validation and held-out splits of both conditions. Arithmetic verification and Eq. 1 reach identical training loss, and the stimulus set admits both as perfect solutions. We call each operand position a *slot*, and the digit-width a slot receives during training its *format*.

3 Representation without capability

Testing whether a given learned representation (property) entails decision-making or any other behavioral capability, requires two independent measurements on the same model: whether information about the property is recoverable from the model (representation), and whether the model can act on it (capability). Widely used intervention methods, such as directional ablation and activation steering, operationalize the assumption that a decodable direction is a usable one. Here, we test this presupposition for parity in pretrained GPT-2, a property with an exact definition and a direct behavioral test, so that both measurements can be made without ambiguity.

To our knowledge, the representation of numeral token parity has not been probed directly in GPT-2. To discover whether parity is a property represented in GPT-2, we trained a logistic probe on the 768-dimensional token embeddings of the 505 integers between 0 and 999 that the model encodes as single tokens, and evaluated it on 5-fold cross-validation, so that every numeral was classified by a probe fitted on the remaining 80% of numerals and therefore a high score signifies generalization across tokens. The probe managed to separate odd from even numerals at an accuracy of 0.994, against an empirical chance level of 0.523 measured with shuffled labels (first column Fig. 1a), suggesting that pretraining has placed odd and even numerals on opposite sides of a single hyperplane in GPT-2's embedding space. To discover whether a linear probe captures all the parity information in the embeddings and test whether non-linear separations further represent parity, we trained a non-linear probe with one hidden layer. This probe reached a held-out accuracy of 0.990, indicating that the parity information in GPT-2 embeddings is linearly encoded.

A parity hyperplane admits two interpretations: GPT-2 could encode parity as an arithmetic property of each number, or it could encode the identity of each numeral's final digit. To distinguish these interpretations, we trained four further probes with the same protocol, each targeting a property of the numeral in Fig. 1a. The final digit probe (second column Fig. 1a) is trained to predict which of the ten digits 0–9 a numeral ends in (10-way) and therefore tests the digit-identity interpretation. The probe separates all ten classes at an accuracy of 0.994 against a baseline of 0.166, indicating that

GPT-2 representations also tell apart digits with the same parity and so establishing that its embeddings encode final-digit identity. The first digit probe (third column, Fig. 1a) is trained to predict the leading digit of each numeral (10-way), informative because this property is independent of the final digit. The probe achieves an accuracy of 0.964 against a baseline of 0.220, demonstrating that GPT-2 encodes digit identity beyond the final position. The divisibility-by-five probe (fourth column, Fig. 1a) tests a property fully determined by the final digit being 0 or 5 and therefore asks whether a second property derived from final-digit identity is linearly recoverable, as the digit-identity interpretation predicts for any such property. The probe reaches an accuracy of 0.998 against a baseline of 0.729. The results obtained with these four probes however still consistent with both interpretations because all are properties dependent of single digits. Therefore, we tested a divisibility-by-three probe (fifth column, Fig. 1a), which depends on the sum of all digits modulo three and therefore cleanly distinguish the two alternative interpretations: a linear probe would recover it only if the embeddings encode modulo 3 arithmetic property directly. This last probe reaches an accuracy of 0.687 against a baseline of 0.661, so divisibility by three stays at baseline for a linear decoder while every single-digit property reaches 0.96–1.00. Together, the results from these five probes indicate that GPT-2 embeddings encode the identity of the digits of each numeral, a lexical property of each token, and parity is linearly decodable as a direct consequence of final-digit identity. This account specifies that acting on parity, for example to make a parity judgment, requires GPT-2 only to use a readout of the final digit, a representation linearly encoded in its embeddings. Therefore, parity in pretrained GPT-2 is a favorable case for testing whether a linearly decodable property is a usable one.

To understand whether GPT-2 can make parity judgements using the digit-identity readout, we ran a battery of behavioral tests. In principle, and as often assumed in mechanistic interpretability, a property that is linearly decodable from the embeddings should at least be reportable by the model. We tested this expectation with four prompt formats, each ending at the position where the model's next token should express a parity judgement (Fig. 1b). P1: "The number n is"; P2: "Is n even or odd? Answer: n is"; P3: "n is an"; and P4, which query is preceded by six labeled examples such as "The number 3 is odd. The number 8 is even. . . . ". For each prompt we presented the one-digit numbers 1–9, the two-digit numbers 10–99 and the three-digit numbers 100–199, and scored a response as correct when the logit of the correct word (" odd" or " even") exceeded the logit of the other. Resulting model accuracy lies between 0.444 and 0.556 in all twelve combinations of prompt and digit count, and between 0.500 and 0.533 for the two- and three-digit numbers, where each cell contains 90 to 100 items. All together, these results show that pretrained GPT-2 performs at chance on parity judgements, while the model encodes parity along a single direction of its embedding space at 0.994: *representation without capability*.

Parity is a property that makes the representation-capability dissociation testable with a precision that semantically rich concepts lack. For parity both the representation and the computation that converts it into a judgment can each be specified exactly. For semantically richer concepts, for example species-specific characteristics, a probe that separates felines from canines can establish a lexical category boundary without the model being able of acting on the concept, for instance for predicting differential behavior between the two species.

4 Fine-tuning turns on capability without abstraction

We fine-tuned GPT-2 in a task offering a direct way to investigate whether training can turn the parity dormant representation into a capability (usable representation). We used the arithmetic verification task we presented in Section 2, both its easy and difficult variants, which derive from neuroscience literature (Kalinova et al., 2026). The training data from both variants reward arithmetic verification and parity checking equally, because each false candidate differs from the correct result by one. In one hand, if through fine-tuning GPT-2 learns and uses arithmetic verification, it would have acquired a new capability in the process. In the other hand, if GPT-2 resolves the task by parity checking, it would have turned on the dormant representation of Section 3 into a capability.

Both fine-tuned models solve the task reaching validation accuracies of 1.000 and 0.979 on the easy and the difficult condition respectively. However, the difficult condition requires substantially longer training to reach the peak of its first grokking cycle (Appendix Fig. 5): the difficult condition takes 8,500 steps till peak, while the easy condition does it in 1,500 steps (difficult condition reaches

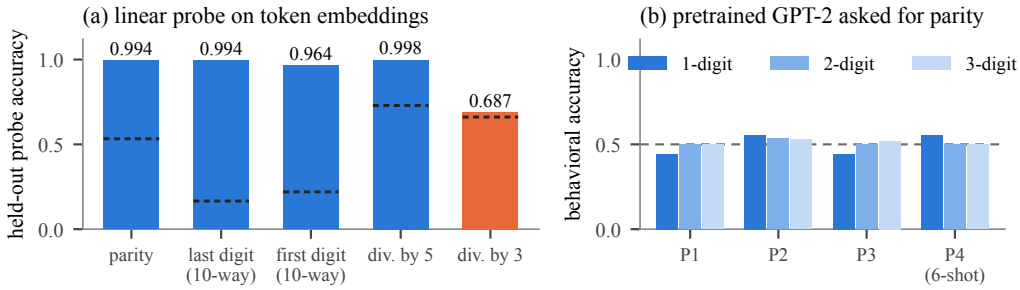

Figure 1: Representation without capability in pretrained GPT-2. (a) Held-out accuracy of five linear probes on the embeddings of the 505 single-token numerals; dashed ticks mark the higher performing of the baselines (majority-class or shuffled-label baselines). Properties determined by single digit identity reach 0.96–1.00 accuracy, but divisibility by three, which depends on the sum of all digits, stays near its baseline. (b) Accuracy of the model's odd/even judgment under the prompts P1 "The number n is", P2 "Is n even or odd? Answer: n is", P3 "n is an", and a six-shot prompt P4 with labeled examples; the dashed line marks chance.

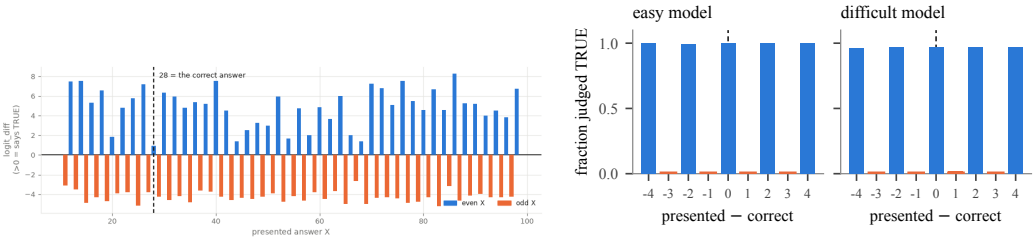

Figure 2: The fine-tuned models judge parity. (a) Δ_{logit} of the difficult model for "19 + 7 + 2 = X" over $X \in [11, 98]$; every even X (blue) is accepted, every odd X (orange) rejected, and the correct answer 28 carries one of the weakest acceptances. (b) Fraction of validation equations judged TRUE when the presented answer equals correct + offset. Even offsets (blue), which preserve parity, are accepted at the rate of the correct answer; odd offsets (orange) are rejected.

1.000 at 32,500 steps because the stored bit for zero is weak - Appendix A.4). This success, as said, admits two alternative explanations about which rules produces it: because a difference of one always inverts parity, the rule of Eq. 1 works, but also arithmetic verification can label every item correctly. The marked difference in training steps between conditions reflects the effort: the model is wiring up a nine-fold larger inventory of bits, a much bigger representation space to consult. We discovered that the model applies the rule of Eq. 1 by probing its verdicts with candidate results for which arithmetic and parity make opposite predictions. Specifically, we fixed a single equation, 19 + 7 + 2, and presented the difficult model with every candidate result X between 11 and 98. An arithmetic verifier would accept $X = 28$ alone, whereas a parity checker would accept every X that shares the parity of 28, since the operands are odd, odd and even, and their XOR is even. The difficult model accepts all 44 even values and rejects all 44 odd values (Fig. 2a). The correct arithmetic answer receives one of the weakest acceptances in the sweep, and the most confident acceptance goes to $X = 86$, with $\Delta_{\text{logit}} = +8.31$. To establish that this behavior characterizes both models across the task (GPT-2 fine-tuned for easy and difficult conditions), we presented every validation equation with the candidate result shifted from the correct one by offsets between -4 and $+4$. Arithmetic verification accepts offset 0 alone, and Eq. 1 accepts every even offset, since an even shift preserves parity. Both models accept offsets of ± 2 and ± 4 at rates of 0.965–1.000, the same rates at which each accepts the correct answer, with mean Δ_{logit} between $+9.1$ and $+9.3$ in every even-offset condition, and they judge odd offsets TRUE at rates of 0.000–0.017 (Fig. 2b). The acceptance extends to errors far larger than any in the training data: the difficult model accepts answers wrong by 6 to 30 at rates of 0.955–0.989, and an answer wrong by 30 receives the same mean confidence, $+9.1$, as the correct answer.

These results established that parity drives model verdicts on both models but an open question remains: does arithmetic contribute to such verdicts if parity carries no information? To test this,

Table 1: Parity-neutral re-audit. Replacing the FALSE candidates by $\mathrm{correct} \pm 2$ preserves parity, so Eq. 1 judges every item TRUE and scores exactly 0.5. Accuracy on the original ± 1 split and on the ± 2 split, with the rate of TRUE verdicts on the ± 2 split ($n = 238$ per cell).

model	weight decay	accuracy, ± 1 split	accuracy, ± 2 split	TRUE rate, ± 2 split
easy	1.0	1.0000	0.5042	0.996
difficult	1.0	0.9790	0.4916	0.975
low-decay (easy condition)	0.01	1.0000	0.6555	0.845

we removed parity as a cue by replacing every false candidate in the validation sets with $\mathrm{correct} \pm 2$. A shift of two preserves parity, so every false candidate now shares the parity of the correct result: a model that relies only on Eq. 1 would judge every item TRUE and score exactly 0.5, whereas any arithmetic contribution would lifts accuracy above 0.5. Both models land at chance and answer TRUE to nearly every item: the easy model scores 0.5042 with a TRUE rate of 0.996, and the difficult model scores 0.4916 with a TRUE rate of 0.975 (Table 1). The verdicts of both models therefore rest on parity alone. Gradient descent during fine-tuning selected parity among two solutions with identical training loss: chained-parity rule versus arithmetic. Selection was made towards the cheaper of two solutions (Varma et al., 2023): 1) verifying arithmethic requires a running total across three operands, carry and borrow across the decade boundary in the difficult condition, and a comparison of magnitudes, while 2) checking parity requires only the final-digit readout that the pretrained embeddings already support, combined across the three operands by an XOR. This happened largely because under strong weight decay, training favors the solution with the smaller parameter norm, the mechanism through which grokking selects efficient circuits. To explore the role of high vs low weight decay in this setting, we fine-tuned GPT-2 in the easy condition with a hundredfold weaker weight decay (0.01). The model scored with an accuracy of 0.6555 on the $\mathrm{correct} \pm 2$ parity neutral test, suggesting that the model combined parity with a magnitude cue: the model rejects answers wrong by 6 at rates of 0.894–0.936, and accepts parity-preserving errors less often the further they lie from the correct answer (Appendix A.2). This finding indicates that the strength of the weight decay shapes which solution is selected by fine-tuning. The same pretrained representation can thus be turned into different capabilities that reach identical task performance: which capability emerges depends on the construction of the fine-tuning task, which determines the set of solutions with zero training loss, and on the hyperparameter regime, which determines the solution that training selects. Our results have broader implications, shaping our understanding of how knowledge representation can be differentially turned into differential capabilities. Fine-tuning has thus turned the dormant parity representation into a capability, and we saw it doing so in two forms. Under the grokking regime, both models became pure parity checkers, judging the parity of a three-operand result with near-perfect accuracy (a judgment the pretrained model performs at chance for a single number) and the generalization they display on held-out equations follows from parity. Under weak weight decay, the same representation became one component of a hybrid rule, in which parity decides between the correct answer and its immediate neighbors, and magnitude rejects distant candidates. The observed difference in training length until the first grokking peak is consistent with the parity solution: the difficult model stored about nine times as many parity bits as the easy condition, one per operand token per slot.

Two open questions remain about the origin and the reach of the parity capability: does it draw on the representation present before fine-tuning, and does it extend beyond the tokens seen during training? These questions are relevant because together they indicate the extent to which the capability is abstract. The first question concerns repurposing: if the parity solution is assembled from structure that pretraining already built, fine-tuning connects existing components, and the representation of Section 3 becomes usable once connected. The second question concerns generality. An abstract parity capability computes parity from the numeral itself, and it therefore applies to any numeral, including numerals absent from a given operand position during training. A capability bound to its training inventory instead retrieves a stored value for each familiar token and assigns an arbitrary value to every other token. The two questions also constrain each other. The pretrained representation encodes digit identity for all 505 single-token numerals and generalizes to held-out tokens at 0.994, so a capability that reads this representation should inherit its generality. Testing the capability on unfamiliar tokens therefore reveals both whether fine-tuning built an abstract rule and whether the rule reads the representation that the probes detect.

Table 2: Tokens absent from the N_1 slot during training (difficult model). Each row substitutes one token for N_1 in 1,400 equations "t + N_2 + N_3 = N_4" and reports the fraction of verdicts that match Eq. 1 with $p(N_1)$ set to even or to odd.

token t	kind	behaves as even	behaves as odd	mean $\lvert\Delta_{\text{logit}}\rvert$
104	three-digit, even	0.985	0.015	8.25
105, 107	three-digit, odd	0.985	0.015	8.25
cat, dog	word	0.985	0.015	8.25
62	two-digit, even, trained	0.994	0.006	8.77
63	two-digit, odd, trained	0.150	0.850	5.51

Residual probes at the readout position locate each operand's parity at the decision point of the pretrained model: operand parities are decodable at 0.85–0.99 after block 2 and at 0.97–1.00 after block 6 (Fig. 3a). The parity match, the XOR of the four token parities that equals the label, stays between 0.48 and 0.55 in every layer of pretrained GPT-2. The fine-tuned easy model decodes the parity match at 0.968 after block 1 and at 0.997 after block 2, and holds it to the output. On the readout residual of pretrained GPT-2, feature-disjoint XOR probes reach 1.000 for operator agreement $O_1 \oplus O_2$ and 0.912–0.979 for the three magnitude pairs $(N_i{>}4) \oplus (N_j{>}4)$, with margins of $+0.38$ to $+0.48$ over their shuffled baselines (Fig. 3b). The four parity pairs, whose inputs are individually decodable at 0.995–1.000, reach 0.505–0.696. Pretraining thus builds XOR for combinations that support next-token prediction, such as operator agreement and relative magnitude, and leaves the parity combination near chance. Fine-tuning connects parity identity to the XOR machinery: all four parity XORs rise by $+0.29$ to $+0.47$, reaching 0.966–0.999. The pair that combines magnitude with parity changes by $+0.04$, and the five remaining pairs fall by 0.18 to 0.37; operator agreement drops from 1.000 to 0.626, at its shuffled baseline of 0.634. The loss of $O_1 \oplus O_2$ exceeds the erosion of its inputs under weight decay (O_1: 1.000 to 0.737); showing that the parity XOR reuses the pretrained XOR circuit requires activation patching (Appendix A.5). he probes establish that the parity combination appears and the others decline. Whether fine-tuning redirects the pretrained XOR circuit toward parity features remains a hypothesis to be tested through activation patching (Appendix A.5). The second question towards abstraction concerns the reach of the parity capability. We tested ross-condition transfer and saw that the easy model scores 0.605 on difficult validation (95% CI $[0.542, 0.665]$) and the difficult model scores 0.559 on easy validation ($[0.495, 0.620]$). Probe sets that vary one property at a time separate format from arithmetic (Fig. 4a). With every slot in its trained format, the difficult model generalizes across arithmetic content: it scores 0.933 on two-digit equations that stay within one decade, a type absent from its training set, 0.975 after permuting the single-digit operands, and 1.000 on equations whose intermediate result exceeds 99. A digit-width change in a single slot inverts the verdict: a two-digit N_2 gives 0.143, a two-digit N_3 gives 0.210, and a one-digit answer gives 0.227, while mean $\lvert\Delta_{\text{logit}}\rvert$ under the N_2 and N_3 changes remains at 7.7 and 7.0 against 8.6 in the native format. Eq. 1 predicts exactly this signature, since a single wrong bit inverts an XOR deterministically and leaves confidence intact. Moreover, a one-digit N_1 isolates the contribution of an unfamiliar token. In set D, where N_1 has one digit and the answer keeps its trained two-digit format, accuracy is 0.960–1.000 for every even N_1 and 0.000–0.033 for every odd N_1 (Fig. 4b), so the model supplies the constant value *even*. Tokens absent from the N_1 slot during training behave identically whatever their identity (Table 2). With N_1 replaced by the even numeral 104, the odd numerals 105 and 107, or the words "cat" and "dog", the verdicts follow Eq. 1 with $p(N_1)$ set to even on 0.985 of the items, at mean $\lvert\Delta_{\text{logit}}\rvert$ 8.25 in all five cases. Trained two-digit tokens in the same slot register their own parity, with 62 behaving as even at 0.994 and 63 as odd at 0.850. The fine-tuned model therefore stores one parity bit per token and slot, acquired from the tokens seen in that slot, and assigns the default *even* to every other token. The composition generalizes to new combinations of familiar tokens, as the difficult model reaches 0.979 on held-out equations after training on 4.9% of its equation space, whereas the stored bits remain tied to the token inventory of each slot. Residual probes separate an absent slot parity (N_1) from a misread one (N_2, N_3) (Appendix A.3), and the residual 2.1% error of the difficult model concentrates on the token " 0", whose weak stored bit produces the single site of genuine uncertainty (Appendix A.4).

We finally questioned: where does fine-tuned GPT-2 reads parity information from? and, how it uses it? We performed directional ablation on the residual stream and observed that by fitting a

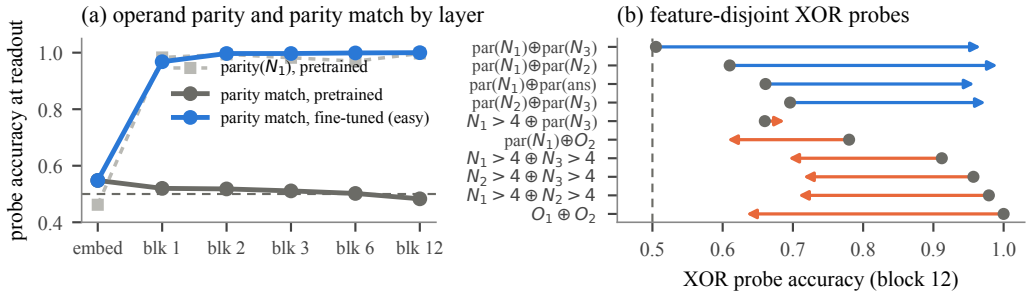

Figure 3: Fine-tuning connects parity and XOR predate fine-tuning. (a) Probe accuracy at the readout position on the full easy dataset ($n = 2{,}420$). Pretrained GPT-2 carries the parity of N_1 to the decision point by block 1 (dashed), and its parity match, the XOR, stays at chance in every layer (grey). The fine-tuned easy model computes the parity match by block 2 (blue). (b) Feature-disjoint XOR probes at block 12, from pretrained (grey dot) to fine-tuned (arrow head). Parity pairs (blue) rise from 0.505–0.696 to 0.966–0.999; among the remaining pairs (orange), the magnitude–parity pair rises by 0.04 and the other five fall by 0.18–0.37.

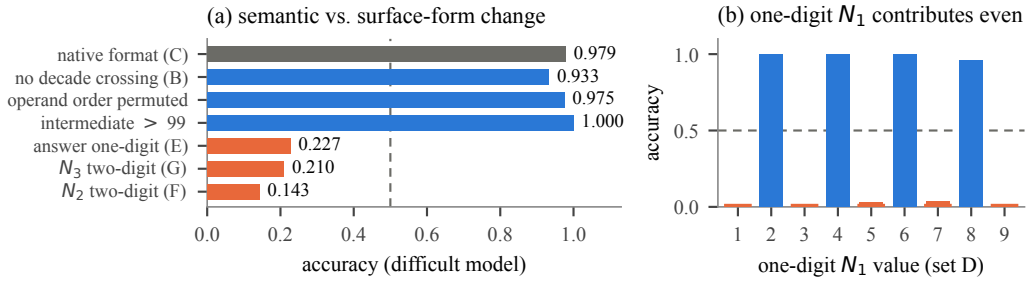

Figure 4: Semantics survive, surface form inverts the verdict (difficult model). (a) Accuracy under semantic changes with every slot in its trained format (blue) and under a digit-width change in one slot (orange); grey marks the native format. (b) Set D, a one-digit N_1 with a two-digit answer, so that N_1 is the only slot out of format. Even N_1 values (blue) are judged correctly and odd values (orange) inverted, since the model supplies the value *even* for every one-digit N_1.

parity basis on a clean forward pass and projecting it out of every block leaves accuracy at 1.000. Each model block writes parity back into the stream, restoring its decodability from 0.723 at block 1 to 1.000 by block 3. Removing parity from the embedding matrix leaves accuracy at 0.983, so the model reads parity from the residual stream and leaves the embedding direction unused. An adaptive procedure, which refits the basis at each block on the ablated stream and compares parity with an equally decodable property of the same token, reveals the use of parity. In the easy model, removing parity at rank 40 costs 6.5 accuracy points, against 0.2 for the matched control and 0.0 for random directions (Appendix 3). Parity removal leaves the model confident and wrong, whereas generic damage leaves it hesitant and right, the signature produced by breaking a slot's format at the input. In the difficult model, parity and the matched control damage accuracy equally (0.9554 and 0.9582 at rank 80), consistent with parity over ninety operand tokens occupying more than 80 directions. Fine-tuning also dispersed parity: the same 40-dimensional cut reduces its decodability to chance (0.4916) in pretrained GPT-2 and only to 0.8183 after fine-tuning.

AI USE STATEMENT

We used AI assistance, specifically Claude Sonnet to accompany coding and prepare the first manuscript version. Project ideation, critical thinking and interpretation are full responsibility of authors. AI outputs were not used as final products.

Reproducibility statement

All experiments are implemented as Jupyter notebooks and provided as supplemental material.

Ethics statement

This work studies a publicly available language model on a synthetic arithmetic task and involves no human subjects, personal data, or new data collection. We identify no direct risk of harm. Our findings concern methods for interpreting and evaluating models, and they are intended to support more reliable assessments of model capabilities.

References

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. In *ICLR Workshop Track*, 2017. arXiv:1610.01644.

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

Yize Cheng, Chenrui Fan, Mahdi JafariRaviz, Keivan Rezaei, and Soheil Feizi. Model-adaptive tool necessity reveals the knowing-doing gap in llm tool use. *arXiv preprint arXiv:2605.14038*, 2026. URL https://arxiv.org/abs/2605.14038.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/toy_model/index.html.

Joshua Engels, Eric J. Michaud, Isaac Liao, Wes Gurnee, and Max Tegmark. Not all language model features are one-dimensionally linear. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

Nikhil Garg, Jon Kleinberg, and Kenny Peng. How many features can a language model store under the linear representation hypothesis? *arXiv preprint*, 2026. URL https://arxiv.org/abs/2602.11246.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP)*, pp. 2733–2743, 2019.

Jonathan Kahana, Eliahu Horwitz, Imri Shuval, and Yedid Hoshen. Deep linear probe generators for weight space learning. *arXiv*, 2025. URL https://arxiv.org/abs/2410.10811.

M. Kalinova, B. Kerkova, A. Kalina, et al. Temporal order of activations and interactions during arithmetic calculations measured by intracranial electrophysiological recordings in the human brain. *Scientific Reports*, 16:5587, 2026.

Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. In *COLM*, 2024.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, 2013.

Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. In *BlackboxNLP Workshop, EMNLP*, 2023.

Chris Olah and Anthropic Interpretability Team. Model diffing. Transformer Circuits Thread, 2024.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. In *International Conference on Machine Learning (ICML)*, 2024.

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

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering llama 2 via contrastive activation addition. In *ACL*, 2024.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Activation addition: Steering language models without optimization. *arXiv preprint arXiv:2308.10248*, 2023.

Vikrant Varma, Rohin Shah, Zachary Kenton, János Kramár, and Ramana Kumar. Explaining grokking through circuit efficiency. *arXiv preprint arXiv:2309.02390*, 2023.

Constantin Venhoff, Iván Arcuschin, Philip Torr, Arthur Conmy, and Neel Nanda. Understanding reasoning in thinking language models via steering vectors. *arXiv preprint*, 2025.

Wang et al. Model-adaptive tool necessity reveals the knowing-doing gap in llm tool use. *arXiv preprint arXiv:2605.14038*, 2026.

Andy Zou, Long Phan, Sarah Chen, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

A Appendix

Table 3: Held-out adaptive ablation. Task accuracy after removing k directions per block, averaged over two seeds. The matched control removes an equally decodable property of the same token (N_1 above its median). Baseline accuracy is 1.0000 (easy) and 0.9826 (difficult, held-out halves).

	easy model, rank k				difficult model, rank k		
removed direction	20	40	80	120	20	40	80
parity(N_1)	0.9971	**0.9351**	**0.8636**	0.8748	0.9826	0.9765	0.9554
matched control	0.9996	0.9979	0.9901	0.9260	0.9838	0.9793	0.9582
random control	1.0000	1.0000	0.9996	0.9764	0.9838	0.9830	0.9813

A.1 Dataset and training details

Both conditions hold out validation and paper-holdout splits at the *equation* level (both the TRUE and FALSE version of an equation land in the same split), so no cell used for evaluation was trained on. Validation sets are 238 items each, exactly balanced TRUE/FALSE. An answer-only "do-nothing" baseline – an oracle that ignores the equation and always predicts the majority label for the presented answer value – scores 0.618 on *arabic_easy* validation and 0.681 on *arabic_difficult* validation; this, not 0.5, is the correct floor for judging whether a cross-condition or ablated score reflects any use of the equation at all.

A.2 The low-decay model combines parity with magnitude

The low-decay model is trained on the easy condition at weight decay 0.01. Its confidence decays with distance from the correct answer: it accepts offsets of ± 2 at 0.705–0.758 and offsets of ± 4 at 0.264–0.319, and it rejects answers wrong by 6 at 0.894–0.936 (Fig. 6). The grokking models accept all of these parity-preserving errors at the rate of the correct answer. The low-decay model therefore uses parity together with a magnitude cue, which also explains its 0.6555 accuracy on the parity-neutral split (Table 1).

Table 4: Six conclusions drafted during this investigation and later retracted by a control run against them. We keep this table because the refuted framings are exactly the ones a less adversarial analysis of the same numbers would have kept.

retracted claim	what refuted it
Fine-tuning *destroyed* a pretrained representation (3-digit parity 0.990 $\rightarrow$ 0.541).	Artifact of standardizing on pooled statistics and then scoring a subgroup whose embedding norms had shrunk $\sim 5\times$ under fine-tuning's weight decay. Within-set standardization shows parity intact at ≈ 0.99 everywhere.
The embedding space merely *rotated* under fine-tuning.	Procrustes alignment (fit on non-number tokens) achieves fit quality 0.408, far from the ≈ 1.0 a true rotation would show: the space was genuinely reorganized.
Fine-tuning *recruited* the pre-existing parity direction (cosine similarity 0.98 with the pretrained direction).	Removing that direction leaves task accuracy essentially unchanged (1.0000 $\rightarrow$ 0.9832); the high cosine reflected that a 1,500-step run had barely moved the embedding, not that the direction was in use.
The 0.979 own-condition ceiling is a training-*coverage* gap (one N_1 value never seen in training).	The unseen-N_1 validation items are all answered correctly; every one of the five errors instead contains a zero-valued operand.
A single-digit N_1 is intrinsically "read as odd."	The probe had two slots out of format at once (N_1 *and* the answer); their errors canceled and inverted the apparent direction. Holding one slot in-format at a time reveals the true default is *even*.
Pretrained GPT-2 computes *no* feature combination of any order.	True only for parity pairs, and the tested pairs were not all disjoint (sharing an underlying variable lets a rich single-feature encoding decode their XOR for free). On disjoint pairs, pretrained GPT-2 computes XOR at 0.91–1.00 for other feature combinations.

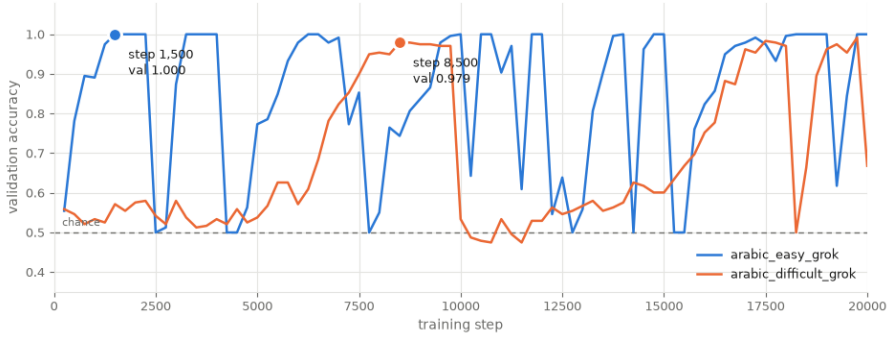

Figure 5: Validation accuracy of the two grokking runs over the first 20,000 steps. Both runs alternate between grokking and collapse; the markers indicate the analyzed checkpoints, the peak of each run's first grokking cycle (easy: step 1,500, accuracy 1.000; difficult: step 8,500, accuracy 0.979). The difficult run first reaches 1.000 at step 32,500, nine cycles later.

A.3 Absent and misread slot parity

Residual probes at block 12 of the difficult model, on 800 constructed equations per set with exactly one slot out of format, separate two routes to an inverted verdict. For a one-digit N_1, parity(N_1) is decodable at 0.501, so the slot's parity is absent at the readout and the default *even* enters the XOR. For a two-digit N_2 or N_3, the slot's parity stays decodable at 0.816 and 0.864, against 0.98 for in-format slots, while task accuracy is 0.130 and 0.174. Below-chance accuracy with decodable information indicates that the model's own reader extracts the inverted value; this reading follows from the inversion and awaits a direct measurement of the reader. Two out-of-format slots cancel: on

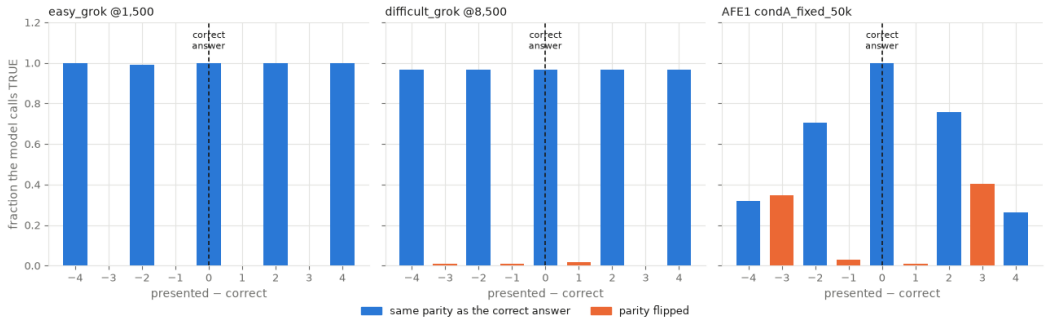

Figure 6: Fraction of validation equations judged TRUE as a function of the offset between presented and correct answer, for the two grokking models and the low-decay model. Blue offsets preserve parity; orange offsets invert it.

easy-condition equations, where N_1 and the answer both leave the difficult model's format, accuracy is 0.058 for even and 0.948 for odd N_1, the reverse of set D, which is why every probe set in Section 4 holds exactly one slot out of format.

A.4 The token "0" carries a weak stored bit

All five validation errors of the difficult model contain the operand 0, and the one N_1 value absent from training (10) is answered correctly on both of its validation items. On 800 constructed equations per digit in N_2, the token "0" yields 0.800 against 0.960–0.983 for the other nine digits. Its errors occur at mean $|\Delta_{\text{logit}}|$ 1.60 against 7.22 when correct, and every item with $|\Delta_{\text{logit}}| \geq 6$ is answered correctly. The effect accumulates across slots: accuracy is 0.981 with both N_2 and N_3 non-zero, 0.839 and 0.871 with a zero in one of them, and 0.519 with zeros in both, where 63.4% of items lie at $|\Delta_{\text{logit}}| < 2$. The easy model handles the same token at 1.000 in every slot. Training frequency accounts poorly for the effect, since digit 1 appears as often as digit 0 in the difficult training set (0.146 against 0.147) and scores 0.978; the cause remains open.

A.5 Open questions carried forward

We list, without resolving, the concrete next steps this investigation's own open-questions log identifies as load-bearing: (1) activation patching between minimal pairs that differ in one operand's parity, to establish causally (not just correlationally) that fine-tuning repoints a specific pretrained circuit; (2) ablating the parity subspace from *pretrained* GPT-2 before fine-tuning, to test whether the pre-existing representation is merely available or actually necessary for the $\sim$1,500-step grokking speed; (3) retraining on a dataset where wrong answers are drawn from $\{\pm 1, \pm 2\}$, which caps the parity shortcut's achievable accuracy near 0.75 and forces a magnitude comparison to do better, as the only fix that does not merely re-evaluate an already-shortcut-learned checkpoint; (4) measuring general language-modeling perplexity before and after fine-tuning, since parity was never a pretrained *capability* and so cannot license a claim about capability loss, while the observed $\sim 5\times$ embedding norm shrinkage under $\text{wd} = 1.0$ makes a genuine capability cost plausible and currently untested; and (5) replicating the representation/operator/binding decomposition for a second model family and a second latent property, to test whether it is specific to parity in GPT-2-small or a more general description of how fine-tuning composes pretrained structure.