Deterministic "Neurons" Inside an Existing Transformer MLP

A Three-Seed Causal Study of Natural-Language Addition, Routing, Preservation, and Registered State in Qwen2.5-0.5B

Josiah Wilson
Independent Researcher

July 27, 2026

Abstract

We test whether a pretrained language model can learn to use a deterministic calculator that occupies a small activation subspace inside an existing transformer MLP, rather than invoking an external tool or replacing an answer after generation. We modified frozen Qwen2.5-0.5B-Instruct at decoder layer 16. Twenty-eight of its existing 4,864 MLP coordinates form a typed interface: sixteen learned route/operand/digit coordinates feed a frozen decimal ripple-carry adder, and twelve deterministic result coordinates return through learned columns of the ordinary MLP down projection. The residual and MLP widths do not grow. The interface has 25,088 learned weights (0.0051% of the 494-million-parameter base); the calculator has zero.

Development was explicitly adaptive and all failed audits were retained. An initial shared three-seed holdout produced 58/60 exact additions per seed and a 55/60 paired result-ablation loss per seed, but falsely routed 17/180 adversarial negatives. Retraining only the 1,792 route-row weights yielded 0/360 false routes and 360/360 token-exact preservation across two subsequent exact-string-disjoint audits. A deterministic response-local operand register then eliminated a sequence-time state-loss failure.

In the final frozen register audit, three independently learned interfaces answered 173/180 additions exactly (96.11%), including 90/90 word problems. They recovered 174/180 exact operand registers; all 174 remained stable and produced exact calculator digit-plus-end trajectories. Calculator-result ablation left 9/180 correct, a paired causal loss on 165/180 examples. All 180/180 adversarial negatives remained route-off and token-identical to untouched Qwen. In a retrospective same-prompt comparison, untouched Qwen returned the exact requested numeral on 1/60 additions under the same eight-token budget, versus 57/60, 58/60, and 58/60 for the three implants. A 64-token base-only sensitivity recovered the correct sum as its final integer on 27/60 prompts but still produced only 1/60 exact numeral-only responses. The compound protocol nevertheless failed: mean exact accuracy was 57.67/60 against a frozen 58/60 threshold, and one seed decoded exact calculator symbols for $0 + 0$ as the token "I." The result strongly supports the narrow causal implant hypothesis while locating remaining failures in learned input/output interfaces. Route, operand, and position state remain generation-runtime microcircuit state; arbitrary recurrent multi-call reasoning is not established.

Technical summary

Final frozen audit outcome	Result
Exact additions	173/180 (96.11%)
Exact first-step operand registers	174/180 (96.67%)
Exact calculator trajectories	174/180 (96.67%)
Exact natural-language word problems	90/90
Exact after calculator-result ablation	9/180 (5.00%)
Paired correct-to-incorrect ablations	165/180
Stable registers given exact first-step operands	174/174
False routes on adversarial negatives	0/180
Token-exact negative preservation	180/180
Untouched Qwen exact response (retrospective, 8 tokens)	1/60 (1.67%)
Untouched Qwen last-integer recovery (64 tokens)	27/60 (45.00%)

Narrow mechanistic verdict. The in-place path works. Learned Qwen activations supplied typed operands to a fixed calculator; exact internal symbols were causally necessary for most answers; and ordinary frozen downstream layers usually decoded those symbols correctly.

Compound protocol verdict. **FAIL**. Three of five final engineering gates passed. Mean addition accuracy missed its frozen threshold by one aggregate answer, and one exact internal trajectory was misdecoded downstream.

Principal boundary. This is a supervised single-addition activation implant with runtime-managed state. It is not evidence for unlimited calculator calls, spontaneous discovery during pretraining, four operations, residual-native recurrent registers, or arbitrary-prompt safety.

1 Motivation and prior work

Language models are flexible probabilistic sequence predictors, while arithmetic algorithms are rigid state-transition systems. Learned arithmetic architectures can improve extrapolation, but neural models still often fail outside trained lengths or prompt distributions [1, 2, 3]. Compiled transformers and neural algorithmic reasoning instead ask how explicit algorithmic structure can coexist with learned representations [4, 5].

The closest direct prior work is Dietz and Klakow's Integrated Gated Calculator (IGC) [7]. IGC freezes Llama 3.1 8B and learns input, routing, and output mappings around a non-differentiable four-operation calculator. It establishes prior art for a calculator integrated into an LLM forward process. The present project does not claim the broad integrated-calculator concept as novel.

The more specific hypothesis studied here was proposed by Josiah Wilson: selected neuron-like coordinates in an otherwise ordinary network can be assigned deterministic semantics, allowing surrounding learned computation to discover when and how to use exact machinery. In the limiting version, a model trained from scratch would experience the calculator coordinates like any other neurons, except that their transfer function implements a known process. This experiment asks a practical pretrained-model question:

> Can a small frozen Qwen model learn to route ordinary language and distributed number representations through a deterministic subspace inside one existing MLP, then use the result through its ordinary downstream network while preserving unrelated behavior?

Earlier phases of this project tested output-adjacent residual bridges, an internal block wrapper, natural-language routing with a fixed parser, a direct comparison with an IGC-style learned parser,

and a multi-stage neural interface. Phase 7 is the first phase that replaces coordinates of an existing Qwen MLP activation tensor itself.

2 Architecture

2.1 Frozen pretrained model

Every experiment uses `Qwen/Qwen2.5-0.5B-Instruct`, revision

`7ae557604adf67be50417f59c2c2f167def9a775`

[6]. The model has 494,032,768 pretrained parameters, 24 decoder blocks, residual width 896, and MLP intermediate width 4,864. All pretrained weights remain frozen. Generation is greedy.

2.2 In-place activation ABI

For hidden state h at decoder layer 16 (zero-indexed), the original Qwen MLP computes

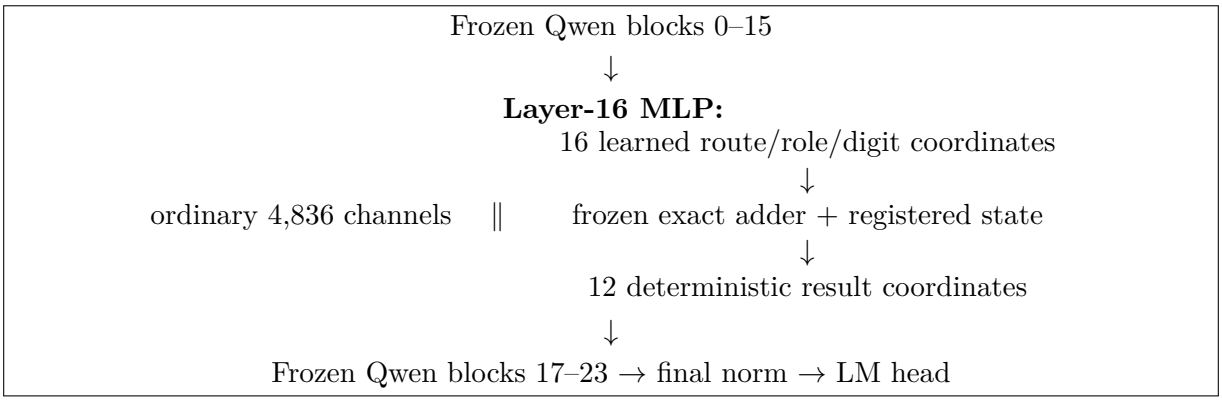
$$y_{\text{MLP}} = W_{\text{down}}\left(\text{SiLU}(W_{\text{gate}}h) \odot W_{\text{up}}h\right).$$

A census selected 28 low-impact coordinates in the 4,864-wide intermediate activation. Their original contribution is separated from the unselected base computation. The implant assigns:

Activation group	Coordinates	Learned weights
Route: `OFF`, `ADD`	2	$2 \times 896 = 1,792$
Token role: none, operand A, operand B	3	$3 \times 896 = 2,688$
Digit type: 0–9, non-digit	11	$11 \times 896 = 9,856$
Result: 0–9, end, pad	12	$12 \times 896 = 10,752$
Total	28	25,088

The first sixteen rows classify route, role, and digit from Qwen's native layer-16 residuals. Role and digit predictions must agree, and digit confidence must exceed 0.90, before a token enters an operand. Hard typed operands feed a zero-parameter decimal ripple-carry cell. Its one-hot result symbol is mapped through twelve learned down-projection columns into Qwen's residual stream. The remaining seven decoder blocks, final normalization, and tied language head are unchanged.

The calculator is not a post-generation correction. Its activation is injected while the model computes the next token, and its result must survive the normal downstream network.



2.3 Exact preservation when inactive

If the latched route is `OFF`, the wrapper restores the selected coordinates' original pretrained MLP contribution exactly. Thus the wrapper computes the original Qwen MLP, not a permanently ablated approximation. A unit test also verifies exact base recovery when no runtime context is installed.

2.4 Runtime microcircuit state

Three fixed state elements were introduced during development:

1. The first response route is latched, preventing later generated text from turning the calculator on.
2. The first valid typed operands and lengths are captured in a response-local register, preventing repeated classification from drifting as the generated sequence grows.
3. A deterministic counter selects successive result symbols and then the end symbol.

These are calculator-runtime tensors, not learned parameters. They make the current single-call circuit coherent, but they are not yet written into or recovered from recurrent Qwen residual state. Generation reruns the full sequence without a key/value cache to expose all token activations to the sequence scanner; this is a prototype implementation, not an optimized inference kernel.

3 Training and architecture selection

3.1 Why layer 16

An early sequence implant at layer 23 recovered direct operands more reliably than word-problem operands. A post-audit depth probe trained equal linear typed interfaces at layers 4, 8, 12, 16, 20, and 23. Joint word-problem role/digit accuracy was 100% at layers 4–16, 96.88% at layer 20, and 79.17% at layer 23. Earlier layers preserved digits but routed negatives poorly. Layer 16 combined 100% probe operand classification with 99.4% route recall and zero probe false routes on consumed data, so the compact implant moved there.

3.2 Supervised interface ladder

The hard calculator blocks gradients across its input decisions. Development therefore used explicit typed supervision:

1. Collect frozen layer-16 residuals from positive and negative prompts.
2. Train route, token-role, and digit rows with categorical losses.
3. Teacher-force typed calculator state, then train result columns through Qwen's normal language-model loss.
4. Remove teacher forcing and evaluate the complete hard interface.

The base model, calculator, and all unselected weights remain frozen. Three independent interface/output seeds (13,201–13,203) were trained. This establishes learnability with scaffolding; it does not show that language-model pretraining alone would discover the ABI.

3.3 Targeted router hardening

The first three-seed audit exposed systematic false routes. After retaining that failure, only the two route rows (1,792 weights) were retrained on a family-balanced union of every then-consumed family: 126 positive and 128 negative families, 4,064 training prompts, and 1,016 exact-string-disjoint development prompts. The remaining 23,296 learned interface weights were byte-identical. All seeds reached 504/504 positive and 512/512 negative development decisions at threshold 0.5.

This intervention was adaptive and is not presented as part of the earlier confirmation. It received separate frozen audits.

4 Experimental discipline

Development and evaluation proceeded in a sequence of frozen engineering audits. Each protocol fixed checkpoint hashes, prompt-family strings, data seeds, decoding, outcomes, and gates before the corresponding generation. Later audits are post-audit confirmations of changed architectures, not retroactive replacements and not independent external preregistrations.

Stage	Frozen evaluation	Disposition
Audit 1	40 additions, 40 negatives	25/40 exact; failed positive gate
Audit 2	60 additions, 60 negatives	54/60 exact; operand and ablation gates failed
Audit 3	Three seeds, shared 60+60	58/60 each; routing/preservation failed
Audit 4	Hardened router, three seeds	59/60 mean; routing passed, state drift failed
Audit 5	Operand register, three seeds	57.67/60 mean; 3/5 final gates passed

No unsuccessful seed was discarded. Raw per-token diagnostics include route probabilities, decoded registers, calculator symbols, generated tokens, ablated outputs, and untouched-base outputs.

5 Final frozen audit

5.1 Data and gates

Audit 5 used 30 direct additions, 30 addition word problems, and 60 adversarial non-addition prompts per seed. Operands had one to four digits. Twenty positive and thirty negative family strings were exact-string disjoint from all earlier project families. All seeds received identical prompts in identical order.

The architecture passed only if all seeds met a 57/60 exact floor, their mean reached 58/60, every seed recovered at least 57/60 operands, paired ablation loss was at least 50/60, false routes were at most 2/60, preservation was at least 58/60, and every active exact-input execution had exact internal and external output.

5.2 Per-seed results

Metric	Seed 13,201	Seed 13,202	Seed 13,203
Exact additions	57/60	58/60	58/60
Direct additions	27/30	28/30	28/30
Word problems	30/30	30/30	30/30
Exact first-step operands	58/60	58/60	58/60
Exact calculator trajectories	58/60	58/60	58/60
Exact after result ablation	3/60	3/60	3/60
Paired causal loss	55/60	55/60	55/60
False routes	0/60	0/60	0/60
Token-exact preservation	60/60	60/60	60/60

5.3 Frozen gate disposition

Gate	Observed	Result
Each seed $\geq 57/60$, mean $\geq 58/60$	Mean 57.67/60	**FAIL**
Each seed exact operands $\geq 57/60$	58/60 each	**PASS**
Each paired ablation loss $\geq 50/60$	55/60 each	**PASS**
False routes $\leq 2/60$, preservation $\geq 58/60$	0/60 and 60/60 each	**PASS**
Every exact registered execution exact externally	173/174	**FAIL**

The mean-accuracy gate missed by one aggregate correct answer. This is retained as a failure rather than rounded to 58.

6 Retrospective comparison with untouched Qwen

6.1 Design and metric boundary

After audit 5 was complete, untouched Qwen was run on the identical 60 addition prompts and 60 adversarial negatives. The prompt set, implant checkpoints, and implant outputs were already frozen. For each base generation, the layer-16 wrapper was temporarily uninstalled, restoring the original Qwen MLP. Model revision, chat formatting, greedy decoding, and the primary eight-token response budget matched audit 5.

This comparison was not specified in the frozen audit-5 protocol and is therefore retrospective, not a preregistered endpoint. Two metrics are kept separate:

- **Exact response** requires the complete stripped output to equal the requested numeral.
- **Last-integer recovery** requires only the final integer appearing anywhere in the response to equal the expected sum.

Exact response is the primary descriptive comparison because every positive prompt explicitly requested the numeral without prose. Last-integer recovery is a deliberately generous sensitivity measure for verbose base responses.

6.2 Same-budget result and extended-generation sensitivity

Condition	Exact response	Last-integer recovery
Untouched Qwen, 8 tokens	1/60	2/60
Untouched Qwen, 64 tokens	1/60	27/60
Implant seed 13,201, 8 tokens	57/60	57/60
Implant seed 13,202, 8 tokens	58/60	58/60
Implant seed 13,203, 8 tokens	58/60	58/60

Under identical decoding, the pooled implant exact-response rate was 173/180 (96.11%), compared with 1/60 (1.67%) for untouched Qwen, a 94.44-percentage-point difference. The base model often began an explanation or restated the expression instead of emitting a numeral. Its 2/60 last-integer score at eight tokens includes one truncated response that ended in the operand `0`; thus 1/60 is the more faithful strict score.

The 64-token sensitivity addresses the possibility that the shared eight-token budget merely cut off explanations before their answers. Base last-integer recovery rose to 27/60, while exact numeral-only response remained 1/60. The larger budget therefore revealed genuine arithmetic ability in base Qwen but did not erase the implant advantage.

Prompt subset	Base 8-token exact	Base 64-token last integer	Implant exact
Direct additions	1/30	13/30	83/90
Word problems	0/30	14/30	90/90

For strict exact response, seed 13,201 had 56 implant-only correct prompts and no base-only prompt; seeds 13,202 and 13,203 each had 57 implant-only prompts and no base-only prompt. Against the generous 64-token last-integer base metric, seeds 13,202 and 13,203 solved all 27 prompts solved by base

plus 31 additional prompts, with no base-only success. Both systems missed the same two systematic operand-framing prompts. Seed 13,201 had 31 implant-only and one base-only success, the latter being its known downstream `0+0` decoding failure.

On the 60 adversarial negatives, every implant seed remained token-identical to untouched Qwen (180/180 seed-prompt comparisons). These rows measure preservation rather than task correctness because they do not have a single reference answer. Full prompt-level outputs are retained in the supplementary HTML report and CSV.

7 Causal evidence

The principal causal intervention zeros only the twelve calculator-result activations when the route is active. It does not disable Qwen, alter the prompt, or change route/operand predictions. Full accuracy was 173/180; ablated accuracy was 9/180. On 165 paired examples, the full implant was correct and its ablation was wrong. The residual nine reflect arithmetic that base Qwen could solve independently under the altered activation path.

Internal traces further separate calculation from interfaces:

- 174/180 prompts supplied the intended typed operands.
- The fixed cell produced the exact digit-plus-end trajectory on all 174/174 of those prompts.
- The operand register remained unchanged through every generation step on all 174/174.
- Downstream generation matched the exact trajectory on 173/174.

Thus no observed final error arose from wrong ripple-carry arithmetic on exact registered inputs. The one conditional error occurred after the correct internal result.

8 Routing and preservation

Before hardening, audit 3 produced 17 false routes in 180 adversarial seed-prompt evaluations. The failures were semantically coherent: subtraction, multiplication, squaring, and other number-bearing instructions sometimes resembled addition locally. Route confidence overlapped true requests, so a threshold-only repair would have suppressed legitimate additions.

After targeted route-row training, audit 4 and audit 5 jointly produced:

$$0/360 \text{ false routes}, \qquad 360/360 \text{ token-exact preserved outputs.}$$

Negatives included other operations, labels, concatenation, quoted addition, refusal, explanation, hypothetical properties, and misleading addition words. This is strong in-distribution semantic generalization across new family strings, not a guarantee over arbitrary language.

9 Failure analysis

9.1 Systematic operand framing

All six final-audit operand errors came from two instances of one direct family:

Intended operands	Seeds affected	Representative decoded registers
39, 531	all three	extra/misassigned digit between roles
72, 603	all three	7260, 3 or equivalent role spill

The cross-seed agreement indicates a representational/family weakness, not random initialization noise. Broader role/digit supervision or a structured neural span-to-register interface is the next narrow repair. A fixed decimal parser would solve it but would abandon the fully neural input objective.

9.2 Downstream result decoding

For the prompt whose operands were 0 and 0, seed 13,201 registered both zeros and emitted calculator symbols `0, EOS`, yet the next token was "I." Seeds 13,202 and 13,203 decoded the same example correctly. Result columns therefore make exact symbols highly reliable but not logically guaranteed under the frozen downstream network. A tied digit-token decoder, stronger result-margin objective, or a later redundant result code may remove this last conditional failure.

9.3 State drift and its repair

Audit 4 found one seed whose initially correct operand disappeared after three generated digits because the prototype reclassified all prompt tokens at every step. Capturing operands once fixed that exact case and yielded 174/174 stable registers on fresh audit-5 inputs. This validates registered state as an architectural requirement, while also emphasizing that the current register is runtime state rather than learned residual recurrence.

10 Interpretation

The study supports a specific, bounded claim:

> A frozen small language model can learn a compact interface to a deterministic activation subspace inside an existing MLP, route natural-language additions through it, causally depend on its exact result, and preserve unrelated outputs when inactive.

The evidence is stronger than a demonstration that a Python calculator returns correct sums. The model must produce typed neuron activations from language, the fixed cell runs inside its layer computation, and frozen downstream Qwen must interpret the result. Causal ablation and per-step traces localize success to that path.

The evidence is weaker than the motivating end-state. The learned input and output mappings still make errors, and persistent state is supplied by the generation runtime. The system has no mechanism to close one call, return to ordinary reasoning, discover another arithmetic subproblem, reset its registers, and invoke the same calculator again.

11 Limitations and next experiments

1. **Single operation and scale.** Only nonnegative one-to-four-digit addition was tested in Phase 7, on one Qwen revision.
2. **Supervised ABI.** Route, role, and digit rows received explicit labels. The system did not discover the interface from task reward alone.
3. **Runtime-managed persistence.** Route, operands, and position are not stored in ordinary recurrent residual activations.
4. **No arbitrary multi-call reasoning.** One response may execute one latched addition. There is no learned call/reset/return protocol or loop invariant.

5. **Adaptive research sequence.** Later audits followed observed failures. They are honest new holdouts for changed architectures, not independent replications.

6. **Retrospective base benchmark.** The audit-5 prompts and implant outputs were frozen before untouched-base generation, but the comparison itself was not specified in the audit-5 protocol.

7. **Prompt-family scope.** Exact family-string disjointness does not imply unrestricted linguistic robustness.

8. **Inference efficiency.** Full-sequence recomputation without a cache is slower than a production implementation.

The next architectural milestone is a residual-to-register controller with explicit `IDLE`, `CAPTURE`, `EMIT`, and `RESET` state. It should make state visible inside reserved activation coordinates, permit multiple bounded calls, and train Qwen on multi-step tasks where using the calculator improves task loss. Loop prevention should be structural: one transition per call state, a maximum call budget, mandatory reset after `EOS`, and no activation without a newly captured valid request. Only after that should the experiment add subtraction/multiplication/division, replicate on a Llama-class model, and test more original deterministic domains such as board-game transitions or molecular state calculations.

12 Reproducibility and provenance

The frozen model revision, channel indices, learned checkpoints, protocol documents, raw outputs, and machine-generated analyses are retained in the repository. Key protocols are:

- `PHASE7_MULTISEED_PROTOCOL.md` (audit 3);
- `PHASE7_ROUTER_HARDENING_PROTOCOL.md` (audit 4);
- `PHASE7_OPERAND_REGISTER_PROTOCOL.md` (audit 5).

Audit-5 raw result SHA-256 values are:

- seed 13,201: `0acbcacd51c67579878a36c7741f96fc4ce87a921701ddc00e06afd29c69082d`;
- seed 13,202: `79861af7af101139049ab7632793eb3b0e19a8f9c27861d1bfb195684b9d460e`;
- seed 13,203: `2492fee048aef0303294c5c6f1ca03cdf911a008e0c032ca78bb1a500d288372`.

The retrospective untouched-base comparison is retained as `phase7_results/base_vs_implant_audit5.json` (SHA-256 `b233e38b67f1b8165fa6b0b3131ba4254e5634ed1053cd5efdec198a179c826f`) and `phase7_results/base_vs_implant_audit5_rows.csv` (SHA-256 `236668b5a3fe0218078d82dc5768d9195950d861f663ce62a00c515877e8c2d8`).

The complete problem-by-problem reader is `phase7_base_comparison/report.html`. The comparison implementation and outputs were committed at `43a5787`.

Audit-5 was frozen at commit `4996db6`; the compact result record was committed at `e6dc5f2`. Experiments ran on macOS 26.5.2 arm64 with Python 3.12.12, PyTorch 2.13.0, Transformers 4.57.6, and Apple MPS. Large checkpoint tensors remain local under ignored artifact directories; hashes and all compact/raw evaluation records are tracked.

13 Conclusion

The experiment did not produce a flawless or fully recurrent neural calculator. It did produce a reproducible causal proof that a pretrained 0.5B-parameter transformer can organize natural-language arithmetic around a zero-parameter deterministic subspace occupying 28 existing MLP coordinates. The calculator and its registered state were exact on every correctly framed final-audit execution; route hardening preserved every evaluated negative; ablation removed 165 otherwise correct answers.

The retrospective untouched-base comparison supplies practical context for that causal result. Under identical eight-token generation, untouched Qwen returned 1/60 exact requested numerals versus 173/180 pooled implant responses. Even with 64 tokens and the looser last-integer measure, base recovered 27/60; the two best implant seeds recovered every one of those prompts plus 31 more. This is strong comparative evidence, while its post-hoc status prevents it from being treated as a frozen protocol gate.

The remaining errors are informative. Six are learned operand framing, one is downstream result decoding, and none is observed incorrect addition from exact typed state. That decomposition supports continued work on neural interfaces and recurrent state while keeping the core deterministic-neuron hypothesis alive. The scientifically honest verdict is therefore strong narrow support, not completion of the broader multi-call intelligence claim.

Acknowledgment

The deterministic-neuron hypothesis and research direction were proposed by Josiah Wilson. Experimental design, implementation, local execution, analysis, and drafting were conducted collaboratively with OpenAI Codex under his direction. All failed protocols and observed errors described here are retained.

References

[1] A. Trask, F. Hill, S. Reed, J. Rae, C. Dyer, and P. Blunsom. Neural Arithmetic Logic Units. *NeurIPS*, 2018. https://arxiv.org/abs/1808.00508.

[2] S. Jelassi, S. d'Ascoli, C. Domingo-Enrich, Y. Wu, Y. Li, and F. Charton. Length Generalization in Arithmetic Transformers. 2023. https://arxiv.org/abs/2306.15400.

[3] H. Cho, J. Cha, P. Awasthi, S. Bhojanapalli, A. Gupta, and C. Yun. Position Coupling: Improving Length Generalization of Arithmetic Transformers Using Task Structure. 2024. https://arxiv.org/abs/2405.20671.

[4] D. Lindner, J. Kramár, S. Farquhar, M. Rahtz, T. McGrath, and V. Mikulik. Tracr: Compiled Transformers as a Laboratory for Interpretability. *NeurIPS*, 2023. https://arxiv.org/abs/2301.05062.

[5] W. Bounsi, B. Ibarz, A. Dudzik, J. Hamrick, L. Markeeva, A. Vitvitskyi, R. Pascanu, and P. Veličković. Transformers Meet Neural Algorithmic Reasoners. 2024. https://arxiv.org/abs/2406.09308.

[6] Qwen Team. Qwen2.5 Technical Report. 2024. https://arxiv.org/abs/2412.15115.

[7] F. Dietz and D. Klakow. IGC: Integrating a Gated Calculator into an LLM to Solve Arithmetic Tasks Reliably and Efficiently. 2025. https://arxiv.org/abs/2501.00684.