TRANSFORMERS ARE HARD FOR MACHINES, NOT ONLY FOR CIRCUITS

Anonymous authors
Paper under double-blind review

ABSTRACT

Saha, Xu, Ye, and Yu construct a transformer that no extended arithmetic circuit computes in fewer than $LH \cdot N^{\omega - o(1)} - O(LH \cdot N^2)$ gates when $\omega > 2$, and ask whether the bound survives in the Word-RAM model, whose algorithms branch on the values they compute. We prove one theorem, and a corollary answers their question for the numerically-branched algorithms defined next. Call a Word-RAM algorithm numerically-branched if its operations on real cells come from $\{+, -, \times, /, \exp, \ln\}$ and a real value reaches word memory only in two ways. One is a comparison; the other is a read of its bits by a function whose level sets are countable unions of intervals. Rectified activations, argmax, threshold masking, and top-k retrieval are all computed by numerically-branched algorithms. The theorem: if every extended arithmetic circuit agreeing with a function f on *some* nonempty open subset of a region has size at least s, then every numerically-branched algorithm computing f on the whole region takes time at least s. Every circuit lower bound whose reduction reads only local information therefore transfers. There is no hypothesis on f, because the proof works with the algorithm's own trace. On a region where every comparison and every bit-read has a fixed outcome, that trace is a fixed circuit, and the region is built by nested shrinking rather than by a genericity assumption. The theorem has four consequences, for the transformer of Saha et al., its sigmoid and gated-linear-unit variants, rectified-linear-unit variants on an open piece where every rectifier is active and the perceptron is the identity, and $N \times N$ real matrix multiplication.

1 INTRODUCTION

The cost of evaluating a transformer (Vaswani et al., 2017) is a question of practical as well as theoretical interest. A substantial line of work establishes hardness for the subproblems associated with attention (Duman Keles et al., 2023; Alman & Song, 2023; 2024a), and Saha et al. (2026) prove a bound for a transformer that reaches the matrix-multiplication exponent ω. Their transformer has L layers with H heads each and embedding dimension $m = \Theta(N)$, where N is the sequence length. The head outputs within each layer are added (summation aggregation). When $\omega > 2$, every extended arithmetic circuit (eAC) computing it has size at least $LH \cdot N^{\omega - o(1)} - O(LH \cdot N^2)$ (Saha et al., 2026, Theorem 5.1). Up to the additive term this bound matches the natural algorithm, which computes each head independently by fast matrix multiplication in time $LH \cdot N^{\omega + o(1)}$.

The eAC model is algebraic. A circuit is a straight-line program on exact real numbers over the fixed alphabet $\{+, -, \times, /, \exp, \ln\}$. Their proof uses the Extended Baur–Strassen Theorem for partial derivatives (Saha et al., 2026, Theorem 4.4), which carries the classical theorem of Baur & Strassen (1983) to circuits with $\exp$ and $\ln$ gates. Algorithms on word machines can branch on the values they compute. Whether the exponent-ω bound survives in the Word-RAM model is the question Saha et al. (2026, Section 6) leave open.

Our main theorem yields a corollary that answers the question of Saha et al. (2026) for the numerically-branched algorithms defined below. The theorem is a transfer principle: a regional lower bound against circuits becomes a time lower bound against numerically-branched machines. The transfer is possible because the extraction of matrix products from derivatives of the circuit's output can be made *local*. It uses derivatives near a point, so the algorithm's trace need only agree with the target on an open region of input space.

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.

1.1 Our Results

The Model. The machine has two memories: numerical cells containing exact real numbers and word cells supporting the standard Word-RAM operations. Control flow and addresses depend on word cells. The following informal definition summarizes the interface between the two memories; Definitions 3.1 and 3.2 give the full model.

Definition 1.1 (Numerically-branched algorithms, informal)**.** A *numerically-branched Word-RAM algorithm* operates on real cells using $\{+, -, \times, /, \exp, \ln\}$ and writes comparison outcomes into word cells. With *bit-reading*, it may also apply a fixed map $\chi : \mathbb{R} \to \{0,1\}^w$ whose level sets are countable unions of intervals, and write the resulting word into word memory. Here w is the word size fixed by the input shape. A word may select a real constant to be written into a numerical cell.

This interface admits comparisons, rounding to a fixed representation, and extraction of its bits or fields. It supports rectified activations and top-k selection, as well as softmax stabilization by subtracting a row maximum. The numerical arithmetic remains exact; reading a rounded representation is an operation within this model.

The Theorem.

Theorem 1.2 (Lifting regional circuit bounds, informal; Theorem 3.3)**.** *Let f be defined on a nonempty open set P_*. If every extended arithmetic circuit agreeing with f on some nonempty open subset of P_* has size at least s, then every numerically-branched Word-RAM algorithm with bit-reading computing f on P_* has worst-case time at least s.*

The hypothesis is a *regional* circuit bound: it must hold on every open subset where a circuit agrees with the target. This is the natural form of a lower bound proved by extracting information from derivatives near a point. The theorem applies to arbitrary targets, including piecewise analytic functions, because continuity enters through the machine's arithmetic trace. A computation taking T_A steps yields a circuit with at most T_A gates.

Consequences. The main application preserves the full dependence on the number of layers and heads.

Corollary 1.3 (Transformer evaluation, informal; Corollary 4.1)**.** *Suppose $\omega > 2$. Every numerically-branched Word-RAM algorithm with bit-reading computing the softmax transformer of Saha et al. (2026, Theorem 5.1) takes time at least*

$$LH \cdot N^{\omega - o(1)} - O(LH \cdot N^2).$$

The bound matches the matrix-multiplication exponent across all LH heads. Thus the bound holds even when the algorithm uses intermediate numerical values to choose its operations and memory accesses.

The same extraction extends to the sigmoid and gated-linear-unit constructions of Saha et al. (2026). For ReLU perceptrons, an open region on which the perceptrons act as identities is enough to retain the bound; Corollary C.6 specifies the parameters and Remark C.7 constructs such a region. The ReLU construction demonstrates that one hard open piece can determine the worst-case cost of a piecewise analytic target. For matrix multiplication itself, the transfer takes an especially simple form.

Corollary 1.4 (Matrix multiplication, informal; Corollary 4.2)**.** *Suppose $\omega > 2$. Every numerically-branched Word-RAM algorithm with bit-reading multiplying two $N \times N$ real matrices takes time at least $N^{\omega - o(1)}$.*

1.2 Our Techniques

We build the region by nested shrinking. Starting from the base domain P_*, we process the comparisons one at a time. Each step replaces the current open set by a nonempty open subset on which the next comparison has a single outcome. A comparison whose two sides are equal throughout the current set already has a constant outcome, so we keep the set unchanged. After at most T_A steps, where T_A is the algorithm's running time, the entire execution is fixed on a nonempty open set. Its numerical operations form an eAC of size at most T_A that agrees with the target there.

Bit-reading fits the same construction. Take the current region to be an open ball and let v be the value being read, viewed as a continuous function of the input. If v varies, its image contains a nondegenerate interval. The fibers of χ cover that interval by countably many intervals, so one fiber contains an open subinterval of the image. Restricting to its preimage fixes the extracted word. Constant injections are then fixed constants, and each arithmetic step becomes one circuit gate. Section 3 gives the formal argument.

Three adjustments to the extraction of Saha et al. (2026) are required, because their argument assumes a circuit evaluable everywhere. The first two are local forms of their differentiation and division-elimination theorems (Theorem A.1 and lemma C.1). The third, deferring the slice substitution, is the technical heart of the transformer corollary. Their construction adds auxiliary variables to the attention scores and sets them to zero early in the extraction. We defer that substitution until after the circuit has been made division-free. Section 4 gives the extraction and its cost.

1.3 Other Related Work

Several papers establish hardness for attention under the Strong Exponential Time Hypothesis (SETH) or its consequences. Duman Keles et al. (2023) show that self-attention requires time quadratic in the sequence length, even when it is computed only approximately. Alman & Song (2023) characterize how entry bounds govern the complexity of approximate attention. Alman & Song (2024b) study this dependence for gradient computation, and Alman & Song (2024a) study higher-order attention through Kronecker products. Alman & Yu (2025) shift the target from the cost of attention to the cost of the task. Gupta et al. (2026) study attention at arbitrary temperatures. For a single head at polynomial embedding dimension, they match the cost of rectangular matrix multiplication up to subpolynomial factors under the Generalized High-Dimensional Orthogonal Vectors Hypothesis. A separate literature bounds what a transformer can represent (Merrill & Sabharwal, 2023; Hao et al., 2022; Strobl et al., 2024).

Strassen (1973b) studies branching and program length, with straight-line-program length as the invariant. Bürgisser et al. (1997, Chapter 10) give a textbook treatment using computation trees with equality tests and the degree bound. Meyer auf der Heide & Rollik (1981) simulate a random-access machine over $\{+, -, \times\}$ that evaluates a polynomial by a straight-line program with no loss of time. Shamos & Yuval (1976) bound the number of square-root operations in a program with comparisons by analytic continuation.

Ben-Amram & Galil (2001) carry Ben-Or's topological method to algebraic random-access machines over $\{+, -, \times, /\}$ or over $\{+, -, \times, \lfloor\cdot\rfloor\}$, each optionally extended by a test for integrality, and Mansour et al. (1991) likewise lift a tree bound proved in the presence of floor. Frandsen et al. (2001) prove depth lower bounds for dynamic evaluation of polynomial targets over an algebraically closed field that hold even when branching is free. This allows the bounds to transfer to the integer RAM. A machine following a fixed comparison path executes one straight-line program, as in the Blum–Shub–Smale model (Blum et al., 1989). Work on the hardness of polynomial equation solving also studies branching-free algorithms (Castro et al., 2003).

Casting a real to an integer is excluded from the standard real RAM because a unit-cost machine that reads its registers both as numbers and as bit vectors decides PSPACE in polynomial time (Hartmanis & Simon, 1974). Truncation can also accelerate arithmetic computation. Given k and an integer greater than 2^{2^k}, Bshouty et al. (1992) compute 2^{2^k} in $\Theta(\sqrt{k})$ operations using truncation; without truncation, the bound is $\Omega(k)$. Definition 3.2 admits both in bounded forms: a fixed function of a real value into one w-bit word, and a real constant selected by a word. Erickson et al. (2024) define a real RAM with exactly our split into word and real memory, word-addressed indexing into the real array, and comparison-driven control flow.

Koiran (2000) and Bürgisser (2001) discuss the computational models underlying these comparisons. Fournier & Koiran (1998) show that lower bounds over the reals are not easier than Boolean ones under unrestricted branching. Allender et al. (2009) characterize what unit-cost real arithmetic with sign tests contributes over a Turing machine.

1.4 Discussion

Theorem 3.3 is a statement about two models of computation. Its hypothesis asks only about circuits that agree with the target on some open set, and that is what makes the hypothesis verifiable from a proof that differentiates at a point. This gives the transfer principle a scope beyond transformers: local derivative information can establish hardness for algorithms with numerical branching.

The interface between numerical values and control flow is central to this transfer. Comparisons and the admissible bit-reads of Definitions 3.1 and 3.2 allow successive query outcomes to be fixed on an open region, where the algorithm becomes a circuit. The resulting transformer and matrix-multiplication bounds concern exact evaluation of the stated hard families and assume $\omega > 2$.

1.5 Open Problems

The numerically-branched model computes exactly, whereas a floating-point machine rounds every result. With bit-reading the model can round as well: reading the w-bit floating-point encoding of a rounded value is a bit-reading operation, and writing that value back is a constant injection. This leads to two questions: approximate computation on real inputs and exact computation on a finite grid.

Whether a bound at exponent ω can survive approximation depends on the precision. An approximation Q to a product AB of $N \times N$ matrices has error at most ε in the Frobenius norm if $\|Q - AB\|_F \leq \varepsilon \|A\|_F \|B\|_F$. Sampling $O(1/\varepsilon^2)$ column-row pairs reaches error ε with constant probability (Drineas et al., 2006, Theorem 1), in time $O(N^2/\varepsilon^2)$. When $\varepsilon \geq N^{-\gamma}$ for a constant $\gamma < (\omega - 2)/2$, that time is $N^{\omega - \Omega(1)}$. At sufficiently fine inverse-polynomial precision, we conjecture that the exponent is unchanged. Approximate bilinear algorithms provide an algebraic comparison: those of Bini et al. (1979) convert into exact algorithms at the same exponent (Bini, 1980), while Landsberg & Michałek (2018) give a quadratic border-rank lower bound for matrix multiplication.

The attention pattern of a hardmax head records for each row the position of its largest score. This pattern is piecewise constant, so a circuit with no gates agrees with it on every open piece. Its evaluation cost lies in locating the piece containing the input. Whether a natural target is hard only in locating the piece remains open.

1.6 Paper Organization

Section 2 fixes the circuit model and transformer notation and states the construction we lift. Section 3 gives the machine model and proves the lifting theorem. Section 4 develops the local extraction and derives the machine lower bounds. Section 5 summarizes the resulting transfer from circuit hardness to machine complexity. Proofs not given in the body are in the appendices, each stated beside the result it establishes.

2 Preliminaries

For a positive integer q, write $[q] := \{1, \ldots, q\}$. Throughout, N is the sequence length, m the embedding dimension, L the number of layers, H the number of heads per layer, and ω the exponent of square matrix multiplication. The *input shape* of an input is the list of its dimensions, which fixes how many real coordinates it has: for a transformer it consists of the input length, the embedding dimension and the numbers of layers and heads, and for the product of two $N \times N$ matrices it is the number N. Except for Theorem 2.4 below, the results we import from the literature are collected in Appendix A, in the form in which we use them.

Definition 2.1 (Word-RAM)**.** The Word-RAM with word size w supports unit-cost arithmetic, bitwise, shift, and comparison operations on w-bit words; unit-cost random-access reads and writes to memory; and data-dependent control flow. The word size w is a function of the input shape, large enough that every cell the algorithm uses has a w-bit address. For an algorithm that uses $n^{O(1)}$ cells, where n is the number of input coordinates, $w = \Theta(\log n)$ suffices. For a transformer with $LH \leq N^{O(1)}$, or for $N \times N$ matrix multiplication, this word size is $w = \Theta(\log N)$. The lifting theorem applies to any word size fixed by the input shape.

An extended arithmetic circuit is a straight-line program on exact real numbers.

Definition 2.2 (Extended arithmetic circuit)**.** An *extended arithmetic circuit* (eAC) on inputs $x_1, \dots, x_n$ is a finite sequence of gates $g_1, \dots, g_s$. Each gate g_t applies one operation to one or two operands. The operation is one of the four arithmetic operations $+, -, \times, /$, which take two operands, or the exponential $\exp$ or the logarithm $\ln$, which take one. Each operand is an earlier gate $g_1, \dots, g_{t-1}$, an input $x_1, \dots, x_n$, or a real constant. The *size* of the eAC is s, the number of its gates. An *arithmetic circuit* (AC) is an eAC using only the gates $\{+, -, \times\}$. The *evaluability domain* of an eAC is the set of inputs at which every $/$ gate has a nonzero denominator and every $\ln$ gate a positive argument. The circuit defines a function on its evaluability domain. An eAC *computes* a map $f : S \to \mathbb{R}^k$ on a set S contained in its evaluability domain if k designated wires (gates, inputs, or constants) carry the coordinates of $f(x)$ at every $x \in S$.

The evaluability domain is open, by induction along the gates, since each condition is a strict inequality on a function continuous on the open set where the earlier gates are defined. Allowing an input or a constant to be designated costs no gate. This convention lets the trace of an algorithm that copies an input coordinate to an output slot be read as a circuit of the same size.

The transformers we consider have L layers of H softmax attention heads each.

Definition 2.3 (Attention head and transformer)**.** Fix an input length N, an embedding dimension m, a layer count L, and a head count H. A *transformer* is parameterized by weight matrices $W_{Q,h,\ell}, W_{K,h,\ell}, W_{V,h,\ell} \in \mathbb{R}^{m\times m}$ for $h \in [H]$ and $\ell \in [L]$, together with maps $\Psi_\ell : \mathbb{R}^m \to \mathbb{R}^m$ applied row-wise. The *softmax attention head* (h, ℓ) is the map $f_{h,\ell} : \mathbb{R}^{N\times m} \to \mathbb{R}^{N\times m}$ given by

$$f_{h,\ell}(X) = \mathrm{softmax}\left((XW_{Q,h,\ell})(XW_{K,h,\ell})^\top\right)(XW_{V,h,\ell}),$$

where $\mathrm{softmax}$ acts on each row of its argument, mapping a row $(a_1, \dots, a_N)$ to the row with entries $\exp(a_j)/\sum_{j'} \exp(a_{j'})$. With summation aggregation, in which the outputs of the heads of a layer are added, the transformer maps a token matrix $X^{(0)} \in \mathbb{R}^{N\times m}$ to $X^{(L)}$ through the layer update

$$X^{(\ell)} = \Psi_\ell\left(X^{(\ell-1)} + \sum_{h=1}^{H} f_{h,\ell}(X^{(\ell-1)})\right), \qquad \ell \in [L],$$

where Ψ_ℓ is applied to each row of its argument; we call $X^{(\ell)}$ the *stream* after layer ℓ. The maps Ψ_ℓ are the multi-layer perceptrons (MLPs) of the architecture. Throughout this paper $\Psi_\ell = \mathrm{id}$ unless stated otherwise; this is the setting in which Saha et al. (2026) state the lower bounds we lift. The exceptions are Corollary C.5 and Corollary C.6, where Ψ_ℓ is a gated linear unit, a one-hidden-layer perceptron with a sigmoid-type activation, or a one-hidden-layer perceptron with rectified-linear-unit activation.

The construction that Corollary 4.1 lifts is the following.

Theorem 2.4 (Saha et al., 2026, Theorem 5.1)**.** *There exists a transformer architecture T, a map of the token matrix and the weights in the sense of Definition 2.3, with the following shape. Its input length is $N + 1$ and its embedding dimension is $m = 2N + 3$; it has L layers of H attention heads each; and its input and output dimension are $2N + 3$. This transformer satisfies:*

1. *T is computable by an eAC, each embedding is computable by an arithmetic circuit of size $O(m)$, and every MLP map is the identity.*
2. *Every eAC computing T has size at least $LH \cdot N^{\omega-o(1)}$ when $\omega > 2$.*

The source states the bound of item 2 as $LH \cdot N^{\omega-o(1)} - O(LH \cdot N^2)$, where the subtracted term is the cost of its own postprocessing. When $\omega > 2$ that term is absorbed into the $o(1)$ in the exponent, so the two forms of the bound are equivalent.

3 The Lifting Theorem

3.1 The Numerically-Branched Model

The model combines exact real arithmetic with a word machine that chooses operations and addresses. Comparisons write their outcomes into word memory; bit-reading and constant injection extend this interface in Definition 3.2.

Definition 3.1 (Numerically-branched Word-RAM, restatement of Definition 1.1)**.** A Word-RAM algorithm is *numerically-branched* if its memory cells are partitioned into *word cells* (storing w-bit integers, with w as in Definition 2.1) and *numerical cells* (storing real numbers), and:

(i) (Numerical operations are algebraic.) Every operation that writes a numerical cell, other than a copy between numerical cells, takes operands from numerical cells or real constants and is one of the eAC gates of Definition 2.2, that is, one of $\{+, -, \times, /, \exp, \ln\}$. Division by zero and the logarithm of a nonpositive argument are undefined, and an execution that attempts either is not a valid run. So an algorithm that is correct on a set of inputs performs neither of these undefined operations at any input of that set.

(ii) (Word–numerical crossings are restricted to two kinds.) The only operations that cross the partition between word cells and numerical cells are:

- *Comparisons:* operations that read two numerical cells, or a numerical cell and a real constant, holding values v_1 and v_2, and write a $0/1$ word recording the outcome of a fixed comparison, one of $v_1 < v_2$, $v_1 \leq v_2$, or $v_1 = v_2$;
- *Indexed access:* reads or writes of a numerical cell at an address held in a word cell, treating the numerical cell's value as an opaque real number and writing no word cell.

(iii) (Word cells are unrestricted, and control flow may depend on them and on nothing else.) Word cells and numerical cells occupy separately addressed memories. Word cells are read, written, and combined by the full standard Word-RAM instruction set of Definition 2.1, including indirect addressing within word memory and indexed jumps. Branching may depend on them, including on word cells obtained as outputs of comparisons, and on no other state. The initial contents of word memory depend only on the input shape; the real input occupies numerical cells.

(iv) (Numerical cells are input, written, or constant.) Every numerical cell the algorithm reads, the output cells included, is an input coordinate, the output of an earlier numerical operation of this execution, or a real constant fixed by the input shape. No numerical cell other than the input cells is preloaded with an input-dependent value. The output coordinates are read, when the algorithm halts, from a designated output region of numerical cells disjoint from the cells holding the input. The algorithm is deterministic.

Bit-reading records information about a numerical value in a word, and constant injection converts a word into a selected real constant.

Definition 3.2 (Bit-reading operations)**.** A *bit-reading* operation reads one numerical cell holding a value $v \in \mathbb{R}$ and writes to a word cell the value $\chi(v)$, where $\chi : \mathbb{R} \to \{0,1\}^w$ is a fixed function each of whose fibers, or level sets, is a countable union of intervals. A map with unbounded range, such as the floor function, is admitted in the truncated form the machine can store, $\lfloor v \rfloor$ reduced modulo 2^w or saturated at the ends, which has the same fibers up to that identification. A *constant-injection* operation writes into a numerical cell a real constant determined by the contents of a word cell. A *numerically-branched algorithm with bit-reading* is a numerically-branched Word-RAM algorithm additionally permitted these operations, which widen Definition 3.1(i) and (ii) by these two crossings of the partition and Definition 3.1(iv) by the injected constants.

Every bit-level read at a position fixed by the program falls under Definition 3.2. The floor function, truncation or rounding to any fixed precision, and the extraction of any bit or field of a fixed-point or floating-point representation are all functions of the value whose fibers are countable unions of intervals.

3.2 The Theorem

A regional circuit bound supplies the machine lower bound on the same base domain P_*.

Theorem 3.3 (Numerically-branched Word-RAM Lifting, restatement of Theorem 1.2)**.** *Let $P_* \subseteq \mathbb{R}^D$ be a nonempty open subset, let f be a function defined at least on P_* with values in $\mathbb{R}^k$, and let $s > 0$. Suppose that every eAC C satisfying*

- *the evaluability domain of C contains a nonempty open subset $U \subseteq P_*$, and*

- $C(z) = f(z)$ *for every* $z \in U$,

has size $|C| \geq s$. *Then every numerically-branched Word-RAM algorithm with bit-reading (Definition 3.2) computing* f *on* P_* *(halting and outputting* $f(z)$ *on every input* $z \in P_*$*, with behavior elsewhere unconstrained) has worst-case time at least* s.

Proof. Let A compute f on P_*. Fix the input shape and let T_A be its worst-case running time on P_*. The result is immediate if T_A is infinite, so assume $T_A < \infty$. Call a comparison or a bit-reading operation a query. Start with an open ball $W_0 \subseteq P_*$ and construct nested nonempty open balls $W_0 \supseteq W_1 \supseteq \cdots$ on which successive query outcomes are fixed. The invariant is that the first t query outcomes are constant on W_t. Word memory starts from the same contents and evolves identically until the next query, so the execution prefix and its addresses are fixed. Thus all runs on W_t either reach the same next query or halt along the same remaining trace. Every numerical value in this prefix is a continuous function on W_t, computed by the eAC operations and fixed injected constants.

For a comparison, let δ be the difference of its operands. If δ takes a strictly positive or negative value, continuity supplies a ball on which that sign is fixed. If $\delta \equiv 0$, retain the current ball. Either choice fixes the comparison outcome.

For a bit-read $\chi(v)$, a constant v already gives a constant word. Otherwise, connectedness of W_t implies that $v(W_t)$ contains a nondegenerate interval J. The finitely many fibers of χ cover $\mathbb{R}$ by countably many intervals. At least one of these intervals, I, meets J in an interval with nonempty interior: a countable union of singletons cannot cover J. The set $W_t \cap v^{-1}(\mathrm{int}(I \cap J))$ is nonempty and open, and $\chi(v)$ is constant there. Choose W_{t+1} to be a ball in it.

After at most T_A queries, the execution is fixed on a nonempty open ball W. Its word values, addresses, and injected constants are fixed there. Replace each numerical operation by its eAC gate and each copy by a wire. A constant injection supplies a constant wire, and each output cell designates the wire of its last write, a copied input, or a fixed constant. Word operations and queries add no numerical gates. The resulting circuit has size at most T_A and agrees with f on W. Every gate is evaluable throughout W, because these are valid executions of A. The regional hypothesis gives $T_A \geq s$.

□

The proof requires the final trace to agree with the target on the chosen region. This permits P_* to be an open piece of a larger domain, as in the ReLU application, while the circuit's continuity comes from its gates. The interval-fiber condition ensures that numerical reads have a constant outcome on a smaller ball; Remark B.3 gives a counterexample for an unrestricted read.

4 From Local Circuits to Transformer Evaluation

The lifting theorem reduces the transformer application to a regional circuit bound. The key is to recover every head's matrix product from derivatives of the transformer output at arbitrary auxiliary-variable values. This keeps the extraction inside the region supplied by the machine.

4.1 Local Polynomial Conversion

The extraction uses Lemma C.1, the local conversion for low-degree polynomials. For a polynomial map of degree at most two, an eAC of size s agreeing with it on a nonempty open set gives an arithmetic circuit of size $O(s + n)$ computing it everywhere, where n is the number of inputs. The proof shifts a point of agreement to the origin, runs the formal-power-series construction with the series truncated at degree two, and shifts back. The output is the sum of the coefficient functions; the polynomial identity theorem extends the equality to all inputs. The full construction and cost analysis are in the proof of Lemma C.1.

4.2 Recovering All Heads on an Open Region

We use the affine instance map $\zeta : \mathbb{R}^{D'} \to \mathbb{R}^{D}$ from the construction of Saha et al. (2026), with $D' = LH(3N^2 + N)$ and D the number of token and weight coordinates. Number the heads across all layers by $k \in [LH]$. For each head, the free variables are matrices $A_k, B_k, C_k \in \mathbb{R}^{N\times N}$ and a vector $D_k \in \mathbb{R}^N$. The map ζ places these variables and fixed constants in the token and weight slots so that the score matrix contains $A_k B_k^\top + C_k$. Write $\Phi^{\mathrm{sm}} := T \circ \zeta$ and

$$S_{ki} := \sum_{j=1}^{N} \exp\big((A_k B_k^\top)_{ij} + C_k[i,j]\big).$$

The instance uses the query and key blocks $(A_k \mid C_k)$ and $(B_k \mid I)$, with I the $N \times N$ identity matrix. The value slots contain D_k, and the template columns read by the heads remain fixed through all layers (Saha et al., 2026, Lemmas 5.2–5.3). Its output includes the entries

$$Y_{i,1} = \sum_{k=1}^{LH} \frac{S_{ki}}{1 + S_{ki}}, \qquad Y_{i,j+1} = \sum_{k=1}^{LH} \frac{D_k[j]}{1 + S_{ki}} \quad (i, j \in [N]).$$

We recover all the products from derivatives of two scalar readouts, $f := \sum_i Y_{i,1}$ and $g := \sum_i Y_{i,i+1}$. Differentiating gives

$$\frac{\partial f}{\partial C_k[i,j]} = \frac{\exp\big((A_k B_k^\top)_{ij} + C_k[i,j]\big)}{(1 + S_{ki})^2}, \qquad \frac{\partial g}{\partial D_k[i]} = \frac{1}{1 + S_{ki}}.$$

Both derivatives are positive, and their logarithms recover each product entry:

$$(A_k B_k^\top)_{ij} = \ln \frac{\partial f}{\partial C_k[i,j]} - 2 \ln \frac{\partial g}{\partial D_k[i]} - C_k[i,j]. \tag{1}$$

The identity holds at every value of C_k. Thus derivatives on any open region determine all LH independent matrix products on that same region.

Let an eAC of size s agree with Φ^{sm} on a nonempty open set U contained in its evaluability domain. Append the two scalar readouts and apply extended Baur–Strassen differentiation, which computes all their first derivatives with $O(s + D')$ gates (Theorem A.1). Reverse differentiation retains the original gates and introduces divisions only by existing denominators or logarithm arguments, so it stays evaluable on U. Agreement on the open set gives equality of derivatives there. Appending the logarithms and subtractions gives an eAC of size $O(s + LH \cdot N^2)$ for all the products on U.

Apply Lemma C.1 with $d = 2$ to this quadratic map with $n = D'$. It gives an arithmetic circuit of size $O(s + LH \cdot N^2)$ computing every product everywhere. Setting the auxiliary variables to constants leaves $r := LH$ independent matrix products. For a size-q arithmetic circuit computing these products, the standard comparison between circuit size and bilinear rank, followed by Schönhage's asymptotic sum inequality, gives

$$2q \;\geq\; \operatorname{rank}\left(\bigoplus_{k=1}^{r} \langle N, N, N\rangle\right) \;\geq\; rN^{\omega}.$$

Here $\langle N, N, N\rangle$ denotes the matrix-multiplication tensor, $\bigoplus$ its direct sum on disjoint variables, and rank the minimum number of bilinear products representing that map (Strassen, 1973a; Schönhage, 1981). Theorem A.2 records this bound and its derivation. All conversion constants are independent of L, H, N, so

$$O(s + LH \cdot N^2) \geq LH \cdot N^{\omega}, \qquad s \geq LH \cdot N^{\omega - o(1)} - O(LH \cdot N^2).$$

This establishes the regional circuit bound required by Theorem 3.3.

Delaying substitution is what makes the circuit bound regional. The original extraction specializes the auxiliary matrices to zero early in the argument. Here the subtraction in Equation (1) keeps them free until the circuit is division-free, when specialization is valid everywhere. The machine's execution region can therefore be used directly.

4.3 The Machine Lower Bound

Corollary 4.1 (restatement of Corollary 1.3)**.** *Suppose $\omega > 2$. Let T be the transformer of Theorem 2.4 with softmax attention. Every numerically-branched Word-RAM algorithm with bit-reading computing T uses time at least*

$$LH \cdot N^{\omega - o(1)} - O(LH \cdot N^2).$$

Proof. Let an algorithm $\mathcal{A}$ evaluate T in worst-case time $T_{\mathcal{A}}$. Fix the input shape and compose it with ζ in the existing numerical memory. Choose one input slot for each free variable, preload the template constants, and copy repeated variables into their other slots. This uses $O(LH \cdot N^2)$ operations with fixed addresses, preserving the word size and the numerical interface. The resulting algorithm computes Φ^{sm} on $\mathbb{R}^{D'}$. The regional bound above and Theorem 3.3 give

$$T_{\mathcal{A}} + O(LH \cdot N^2) \geq LH \cdot N^{\omega - o(1)} - O(LH \cdot N^2).$$

Absorbing the instance-construction cost into the additive term proves the bound. □

4.4 Other Perceptrons and Matrix Multiplication

For a GLU activation σ with $\sigma(c) \neq 0$ at some c, fixing the gate to $\sigma(c)$ and scaling the value branch by $1/\sigma(c)$ makes the perceptron the identity (Saha et al., 2026, Proposition B.2). The transformer lower bound therefore extends to these GLUs. For the sigmoid-type activations satisfying Corollary C.5(b), a regional layer-peeling argument recovers all head products with the same $O(s + LH \cdot N^2)$ extraction cost. Corollary C.5 gives the activation conditions and full induction.

The bound also holds for ReLU perceptrons whose parameters make every MLP the identity on a common nonempty open region of positive pre-activations. On that region, the transformer agrees with the identity-MLP construction, so Theorem 3.3 applies directly. Corollary C.6 and remark C.7 give the parameter conditions and construct such a region.

Matrix multiplication uses the polynomial-conversion step directly.

Corollary 4.2 (Matrix multiplication in numerically-branched Word-RAM, restatement of Corollary 1.4)**.** *Suppose $\omega > 2$. Every numerically-branched Word-RAM algorithm with bit-reading computing the matrix product AB for $A, B \in \mathbb{R}^{N \times N}$ uses time at least*

$$N^{\omega - o(1)}.$$

Proof. Apply Lemma C.1 with $d = 2$ to the product map on $\mathbb{R}^{2N^2}$. An eAC of size s agreeing with it on an open set gives an arithmetic circuit of size $O(s + N^2)$ computing it everywhere. The rank bound above with $r = 1$ yields $s \geq N^{\omega - o(1)} - O(N^2)$. For $\omega > 2$, the quadratic term is absorbed into the exponent's $o(1)$. Theorem 3.3 with $P_* = \mathbb{R}^{2N^2}$ gives the result. □

5 Conclusion

Regional circuit hardness gives a general route to lower bounds for machines with numerical branching and admissible bit-reading. Fixing an execution on an open region turns its numerical operations into a circuit whose size is bounded by the running time.

For the transformer family of Saha et al. (2026), local derivative extraction recovers all LH matrix products on the region supplied by the machine. Combined with lifting, it preserves the matrix-multiplication exponent and the aggregate dependence on layers and heads when $\omega > 2$. The arithmetic cost of exact evaluation therefore persists under the value-dependent control flow of the numerically-branched Word-RAM model.

AI USE STATEMENT

The authors posed the research problem and guided the development of the results. We used Claude (Anthropic), under the authors' direction, to assist in developing the proofs and finding relevant literature.

In the language of the ICLR 2027 AI Policy for Authors, AI took part in the required-disclosure tasks of developing models, formulating claims, providing proof ingredients, and writing proofs, and in the recommended-disclosure tasks of drafting, literature search, and reference formatting. The remaining required tasks concern empirical work or translation and do not arise. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Eric Allender, Peter Bürgisser, Johan Kjeldgaard-Pedersen, and Peter Bro Miltersen. On the complexity of numerical analysis. *SIAM Journal on Computing*, 38(5):1987–2006, 2009. doi: 10.1137/070697926.

Josh Alman and Zhao Song. Fast attention requires bounded entries. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. doi: 10.52202/075280-2755.

Josh Alman and Zhao Song. How to capture higher-order correlations? Generalizing matrix softmax attention to Kronecker computation. In *International Conference on Learning Representations (ICLR)*, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/123fd8a56501194823c8e0dca00733df-Abstract-Conference.html.

Josh Alman and Zhao Song. The fine-grained complexity of gradient computation for training large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b. doi: 10.52202/079017-1963.

Josh Alman and Hantao Yu. Fundamental limitations on subquadratic alternatives to transformers. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/76012c39b10425b189c38850da053893-Abstract-Conference.html.

Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22(3):317–330, 1983. doi: 10.1016/0304-3975(83)90110-x.

Amir M. Ben-Amram and Zvi Galil. Topological lower bounds on algebraic random access machines. *SIAM Journal on Computing*, 31(3):722–761, 2001. doi: 10.1137/S0097539797329397.

Dario Bini. Relations between exact and approximate bilinear algorithms. Applications. *Calcolo*, 17(1):87–97, 1980. doi: 10.1007/bf02575865.

Dario Bini, Milvio Capovani, Francesco Romani, and Grazia Lotti. $O(n^{2.7799})$ complexity for $n \times n$ approximate matrix multiplication. *Information Processing Letters*, 8(5):234–235, 1979. doi: 10.1016/0020-0190(79)90113-3.

Lenore Blum, Mike Shub, and Steve Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 21(1):1–46, 1989. doi: 10.1090/S0273-0979-1989-15750-9.

Nader H. Bshouty, Yishay Mansour, Baruch Schieber, and Prasoon Tiwari. Fast exponentiation using the truncation operation. *Computational Complexity*, 2(3):244–255, 1992. doi: 10.1007/bf01272076.

Peter Bürgisser. On implications between P-NP-hypotheses: Decision versus computation in algebraic complexity. In *Mathematical Foundations of Computer Science (MFCS)*, volume 2136 of *Lecture Notes in Computer Science*, pp. 3–17, 2001. doi: 10.1007/3-540-44683-4_2.

Peter Bürgisser, Michael Clausen, and M. Amin Shokrollahi. *Algebraic Complexity Theory*, volume 315 of *Grundlehren der mathematischen Wissenschaften*. Springer, 1997. doi: 10.1007/978-3-662-03338-8.

David Castro, Marc Giusti, Joos Heintz, Guillermo Matera, and Luis M. Pardo. The hardness of polynomial equation solving. *Foundations of Computational Mathematics*, 3(4):347–420, 2003. doi: 10.1007/s10208-002-0065-7.

Petros Drineas, Ravi Kannan, and Michael W. Mahoney. Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. *SIAM Journal on Computing*, 36(1):132–157, 2006. doi: 10.1137/s0097539704442684.

Feyza Duman Keles, Pruthuvi Mahesakya Wijewardena, and Chinmay Hegde. On The computational complexity of self-attention. In *Proceedings of the 34th International Conference on Algorithmic Learning Theory (ALT)*, volume 201 of *Proceedings of Machine Learning Research*, pp. 597–619, 2023. URL https://proceedings.mlr.press/v201/duman-keles23a.html.

Jeff Erickson, Ivor van der Hoog, and Tillmann Miltzow. Smoothing the gap between NP and $\exists\mathbb{R}$. *SIAM Journal on Computing*, 53(6):FOCS20–102–FOCS20–138, 2024. doi: 10.1137/20M1385287. Conference version in FOCS 2020; published online in 2022.

Hervé Fournier and Pascal Koiran. Are lower bounds easier over the reals? In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 507–513, 1998. doi: 10.1145/276698.276864.

Gudmund Skovbjerg Frandsen, Johan P. Hansen, and Peter Bro Miltersen. Lower bounds for dynamic algebraic problems. *Information and Computation*, 171(2):333–349, 2001. doi: 10.1006/inco.2001.3046.

Shreya Gupta, Boyang Huang, Barna Saha, Yinzhan Xu, and Christopher Ye. Subquadratic algorithms and hardness for attention with any temperature. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/8a01099096c85890b1d1aff3c6b4ea56-Abstract-Conference.html.

Yiding Hao, Dana Angluin, and Robert Frank. Formal language recognition by hard attention transformers: Perspectives from circuit complexity. *Transactions of the Association for Computational Linguistics*, 10:800–810, 2022. doi: 10.1162/tacl_a_00490.

Juris Hartmanis and Janos Simon. On the power of multiplication in random access machines. In *Proceedings of the 15th Annual Symposium on Switching and Automata Theory (SWAT)*, pp. 13–23, 1974. doi: 10.1109/swat.1974.20.

Pascal Koiran. Circuits versus trees in algebraic complexity. In *Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 1770 of *Lecture Notes in Computer Science*, pp. 35–52, 2000. doi: 10.1007/3-540-46541-3_3.

Steven G. Krantz and Harold R. Parks. *A Primer of Real Analytic Functions*. Birkhäuser, 2nd edition, 2002. doi: 10.1007/978-0-8176-8134-0.

Joseph M. Landsberg and Mateusz Michałek. A $2n^2-\log_2(n)-1$ lower bound for the border rank of matrix multiplication. *International Mathematics Research Notices*, 2018(15):4722–4733, 2018. doi: 10.1093/imrn/rnx025.

Yishay Mansour, Baruch Schieber, and Prasoon Tiwari. Lower bounds for computations with the floor operation. *SIAM Journal on Computing*, 20(2):315–327, 1991. doi: 10.1137/0220020.

William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545, 2023. doi: 10.1162/tacl_a_00562.

Friedhelm Meyer auf der Heide and Anton Rollik. Random access machines and straight-line programs. In *Fundamentals of Computation Theory (FCT)*, volume 117 of *Lecture Notes in Computer Science*, pp. 259–264, 1981. doi: 10.1007/3-540-10854-8_29.

Barna Saha, Yinzhan Xu, Christopher Ye, and Hantao Yu. On the computational hardness of transformers. In *Proceedings of the 58th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 1073–1084, 2026. doi: 10.1145/3798129.3800819. Full version at arXiv:2603.11332.

Arnold Schönhage. Partial and total matrix multiplication. *SIAM Journal on Computing*, 10(3): 434–455, 1981. doi: 10.1137/0210032.

Michael Ian Shamos and Gideon Yuval. Lower bounds from complex function theory. In *Proceedings of the 17th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 268–273, 1976. doi: 10.1109/sfcs.1976.19.

Volker Strassen. Vermeidung von Divisionen. *Journal für die reine und angewandte Mathematik*, 264:184–202, 1973a. doi: 10.1515/crll.1973.264.184.

Volker Strassen. Berechnung und Programm. II. *Acta Informatica*, 2(1):64–79, 1973b. doi: 10.1007/BF00571464.

Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What formal languages can transformers express? A survey. *Transactions of the Association for Computational Linguistics*, 12:543–561, 2024. doi: 10.1162/tacl_a_00663.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. URL `https://papers.nips.cc/paper/7181-attention-is-all-you-need`.

A Imported Results

This appendix collects the results we import, in the form in which we use them.

Theorem A.1 (Extended Baur–Strassen; after Baur & Strassen (1983), cf. Saha et al. (2026, Theorem 4.4))**.** *Let C be an eAC of size s on n inputs with a single designated output, and let F be the function it defines on its evaluability domain V. Then there is an eAC of size $O(s+n)$ whose evaluability domain contains V. On V it computes F together with all n first-order partial derivatives $\partial F/\partial x_1, \ldots, \partial F/\partial x_n$.*

The classical Baur–Strassen theorem (Baur & Strassen, 1983) bounds the *nonscalar* complexity of F together with all n of its first-order partials by $3L(F)$, where $L(F)$ is the nonscalar complexity of F alone, the least number of multiplications and divisions, other than by constants, in a circuit computing F. Its proof transforms the circuit by reverse-mode differentiation, and the same transformation gives an absolute constant when every operation is charged, which is the measure we use. The extension to eACs with $\exp, \ln$ gates, with the differentiation rules $\partial \exp(g) = \exp(g)\partial g$ and $\partial \ln(g) = \partial g/g$, is Saha et al. (2026, Theorem 4.4), whose proof charges every gate and obtains the constant 5.

Theorem 4.4 of Saha et al. (2026) is stated for a globally defined F and delivers the partial derivatives alone. Theorem A.1 retains F together with its partials throughout the original circuit's evaluability domain. This local form follows from the same reverse-mode construction, detailed below.

The local form follows because reverse-mode differentiation is syntactic. The derivative circuit keeps the gates of C and adds gates that compute one number per wire, the adjoint of that wire. The adjoints are updated while the gates of C are processed from last to first. Write $\bar{u}$ for the adjoint of a wire u. Before any gate is processed, $\bar{u}$ is the constant 1 at the designated output and the constant 0 at every other wire. Processing a gate g adds terms to the adjoint of each of its operands, named a and b below, according to the operation of g:

- a gate $g = a \pm b$ adds $\bar{g}$ to $\bar{a}$ and $\pm\bar{g}$ to $\bar{b}$;

- a gate $g = a \times b$ adds $\bar{g} \times b$ to $\bar{a}$ and $\bar{g} \times a$ to $\bar{b}$;
- a gate $g = a/b$ adds $\bar{g}/b$ to $\bar{a}$ and subtracts $(\bar{g} \times g)/b$ from $\bar{b}$, where the second rule uses $\partial g/\partial b = -a/b^2 = -g/b$;
- a gate $g = \exp(a)$ adds $\bar{g} \times g$ to $\bar{a}$, since $\exp' = \exp$;
- a gate $g = \ln(a)$ adds $\bar{g}/a$ to $\bar{a}$.

An operand that is a constant receives no adjoint.

Every gate is a real-analytic function of the inputs on the open set V, so the chain rule applies there. By induction over the processed gates, each final adjoint $\bar{x}_i$ therefore equals $\partial F/\partial x_i$ on V. An input that no gate reads keeps its initial adjoint, the constant 1 if it is the designated output and 0 otherwise. That constant is its partial derivative, and it is designated at no gate.

It remains to bound the size of the derivative circuit and to check its evaluability domain. The derivative circuit keeps the original gates, so it also computes F. Each original gate contributes at most five new gates, so the derivative circuit has size at most $6s$. Every division it performs is by the denominator b of an original $/$ gate or by the argument a of an original $\ln$ gate. On V the first is nonzero and the second is positive, so the derivative circuit's evaluability domain contains V. The transformation introduces no new $\exp$ or $\ln$ gate.

Theorem A.2 (Matrix-multiplication lower bound; Saha et al. (2026, Lemma 4.5), after Schönhage (1981))**.** *Let ω be the matrix-multiplication exponent. For $A, B \in \mathbb{R}^{N \times N}$ with distinct indeterminate entries, every arithmetic circuit over $\{+, -, \times\}$ (with scalar constants) computing all N^2 entries of the product AB has size at least $N^{\omega - o(1)}$. More generally, for every $r \geq 1$, computing r independent such products, over pairwise disjoint sets of indeterminates, requires size at least $r \cdot N^{\omega - o(1)}$, where the $o(1)$ does not depend on r. The source's bound also allows division gates.*

The error term is uniform in r, as follows from three facts used in the proof. The *rank* of a bilinear map is the least number of products of a linear form in the first matrix's entries with one in the second's from which every output is a linear combination, and its *asymptotic rank* is the limit of the m-th root of the rank of its m-th tensor power. First, the rank of the r products is at most twice the size of a circuit computing them, a bound the source attributes to Strassen (1973a). Second, the asymptotic rank of the r products is at most their rank. Third, by the asymptotic sum inequality of Schönhage (1981), the asymptotic rank is at least $r \cdot N^{\omega}$. Together they give size at least $r \cdot N^{\omega}/2$ for all r and N.

The last imported result passes from agreement on an open set to agreement everywhere.

Theorem A.3 (Identity theorem for real-analytic functions; see e.g. Krantz & Parks (2002))**.** *Let $\Omega \subseteq \mathbb{R}^n$ be open and connected and let $g, h : \Omega \to \mathbb{R}$ be real-analytic. If $g = h$ on a nonempty open subset of Ω, then $g = h$ on all of Ω.*

B Proofs for Section 3

This appendix proves the results of Section 3, beginning with the theorem.

Theorem 3.3 (Numerically-branched Word-RAM Lifting). *Let $P_* \subseteq \mathbb{R}^D$ be a nonempty open subset, let f be a function defined at least on P_* with values in $\mathbb{R}^k$, and let $s > 0$. Suppose that every eAC C satisfying*

- *the evaluability domain of C contains a nonempty open subset $U \subseteq P_*$, and*
- *$C(z) = f(z)$ for every $z \in U$,*

has size $|C| \geq s$. Then every numerically-branched Word-RAM algorithm with bit-reading (Definition 3.2) computing f on P_* (halting and outputting $f(z)$ on every input $z \in P_*$, with behavior elsewhere unconstrained) has worst-case time at least s.

Proof of Theorem 3.3. Let A be a numerically-branched Word-RAM algorithm with bit-reading computing f on P_* with worst-case time T_A over inputs in P_*. If T_A is infinite there is nothing

to prove, so assume that it is finite. Step 1 proves the theorem for an A that performs neither bit-reading nor constant injection; this case carries the construction in its simplest form. Then Step 2 proves the theorem as stated, reusing that construction and adding one case to it.

Step 1. Comparisons only. Suppose for this step that A performs neither bit-reading nor constant injection, that is, A is an algorithm of Definition 3.1. We locate a nonempty open set $W \subseteq P_*$ on which the sequence of comparison outcomes of A is constant, and then apply the hypothesis to the trace of A on W. All regions in the construction lie inside P_*, where every execution is valid.

We construct W by shrinking: it is the last set of a sequence $W_0 \supseteq W_1 \supseteq \cdots$ of nonempty open sets, with one step for each comparison. Set $W_0 := P_*$. Suppose the nonempty open set $W_t \subseteq P_*$ has been constructed so that all executions of A on inputs in W_t agree in their first t comparison outcomes. Then they agree in their entire operation sequence up to the $(t+1)$st comparison, for the following reason. By Definition 3.1(ii)–(iii), control flow, and every address used by an indexed access, depends only on word cells. Word memory evolves identically on W_t up to that point, for two reasons. It is initialized from the input shape, which is the same for every input in $P_* \subseteq \mathbb{R}^D$. By Definition 3.1(ii)–(iii), it is written only by word operations and by comparison outcomes, and the first t of those outcomes agree. In particular, either every execution on W_t performs a $(t+1)$st comparison or none does. If none does, stop and set $W := W_t$.

Otherwise let v_1 and v_2 be the two numerical values compared at the $(t+1)$st comparison, and let $\delta := v_1 - v_2$ be their difference. Along the common trace prefix, δ is a fixed composition of gates from $\{+, -, \times, /, \exp, \ln\}$ applied to the input. Every operation in the prefix executes successfully on all of W_t, because the executions are actual runs of A, which does not crash on the inputs where it computes f. Hence δ is real-analytic, and in particular continuous, on W_t; only its continuity is used below.

The sets $\{z \in W_t : \delta(z) < 0\}$ and $\{z \in W_t : \delta(z) > 0\}$ are open because δ is continuous. There are three cases. If the first set is nonempty, take it as W_{t+1}. If the first set is empty and the second is nonempty, take the second as W_{t+1}. Otherwise $\delta \equiv 0$ on W_t, so the outcome is the same at every input, and we set $W_{t+1} := W_t$. In every case W_{t+1} is a nonempty open subset of P_* on which the first $t+1$ outcomes are constant. Indeed, the $(t+1)$st comparison tests one of $v_1 < v_2$, $v_1 \leq v_2$ and $v_1 = v_2$. Its outcome is therefore a function of the sign of the difference δ, and that sign is constant on W_{t+1}. Since every execution halts within T_A operations, the construction stops with $W := W_t$ for some $t \leq T_A$.

On W the trace of A is a fixed sequence of at most T_A operations, with fixed read and write addresses. By Definition 3.1, its numerical operations therefore form an eAC C_W of size at most T_A. The construction is that of Step 1 in the proof of Proposition B.2, with its fixed addresses supplied by the shrinking rather than by obliviousness. That construction uses three facts, and each holds here. The operation sequence and the read and write addresses are fixed. Every write to a numerical cell other than a copy is a gate of the eAC alphabet (Definition 3.1(i)). And the numerical cells obey the conventions of Definition 3.1(iv). A comparison writes only a word cell, so, like a word operation, it contributes no gate. Each numerical operation becomes one gate, whose operands are the wires of the cells it reads.

The circuit C_W meets both conditions of the hypothesis with $U := W$. The evaluability domain of C_W contains W, because every run on W is valid. And $C_W(z) = f(z)$ for every $z \in W$, because when the fixed trace halts, each output cell holds a fixed designated wire of C_W. That wire is one of three things. If the result of a gate last reached the cell, directly or through copies, the wire is that gate. If an input or a constant last reached the cell through copies, the wire is that input or constant. If the trace never writes the cell, the wire is the constant the cell holds initially by Definition 3.1(iv). Definition 2.2 lets an input or a constant be designated at no gate. The hypothesis applied with $U := W$ gives $|C_W| \geq s$, hence $T_A \geq s$.

Step 2. Bit-reading. Now let A be arbitrary again. We extend the shrinking construction of Step 1 to the bit-reading and constant-injection operations. Call a comparison or a bit-reading operation a *query*. Start from a ball $W_0 \subseteq P_*$ and maintain the invariant that W_t is a nonempty open *ball* on which the outcomes of the first t queries are constant (shrinking any nonempty open set to a ball inside it is harmless). The invariant asks for a ball because the bit-reading case below uses the connectedness of W_t.

As in Step 1, all executions on W_t agree in their operation sequence and addresses up to the $(t+1)$st query. The reason is the same: word memory is now written by word operations and by the outcomes of queries, and the first t of those outcomes are constant on W_t. So either every execution on W_t reaches a $(t+1)$st query, the same one and reading the same cells, or none does. If none does, stop and set $W := W_t$. A comparison is handled exactly as before, with δ now a fixed composition of gates applied to the input and to the values injected so far. Those values are the same at every input of W_t, so δ is still analytic there. For this comparison, the rule of Step 1 yields a nonempty open set, and W_{t+1} is taken to be a ball inside that set.

At a bit-reading step, let v be the value read and χ the function the step applies to it (Definition 3.2). Along the common trace prefix, v is a real-analytic function on W_t, and in particular a continuous one. If v is constant on W_t, the written word is constant, and we set $W_{t+1} := W_t$. Otherwise the image $v(W_t)$ contains two distinct points. This image is the continuous image of the connected ball W_t, so it is a connected subset of $\mathbb{R}$. Hence it contains the nondegenerate interval J between those two points.

The codomain of χ is finite, so χ has finitely many fibers, and each fiber is a countable union of intervals. So $\mathbb{R}$ is covered by countably many intervals, on each of which χ is constant. The interval J is uncountable, and it is the union of its intersections with these countably many intervals, so one of these intervals, call it I, meets J in uncountably many points. Then $J \cap I$ is an interval with more than one point, so its interior is a nonempty open interval, and χ is constant on it. The set $W_t \cap v^{-1}(\mathrm{int}(J \cap I))$ is open because v is continuous, and nonempty because $J \subseteq v(W_t)$. Let W_{t+1} be a ball inside it, as the invariant requires. On W_{t+1} the value v lies in $\mathrm{int}(J \cap I)$, where χ is constant, so the written word is constant.

Constant-injection operations write reals determined by word cells, which are constant on the current region, so they inject fixed real constants. Since every execution halts within T_A operations, the construction stops after at most T_A queries, with a final ball W.

On the final region W, every word value arising during execution, including every bit-reading output, is a fixed constant, so the control flow, the operation sequence, and all injected constants are fixed. The numerical operations of the trace therefore again form an eAC of size at most T_A that agrees with f on W. Bit-reading contributes no numerical gate, and injected constants are eAC constants. Each output cell designates the value it holds when the trace halts: a wire as listed at the end of Step 1, or an injected constant that reached it directly or through copies. As in Step 1, every gate of that trace executes successfully at every point of W, so the evaluability domain of that eAC contains W. The hypothesis applied with $U := W$ completes the proof. □

B.1 Oblivious Arithmetic Algorithms

For an algorithm whose operation sequence is fixed by its input shape, the circuit is obtained directly from its trace.

Definition B.1 (Oblivious arithmetic Word-RAM)**.** A Word-RAM algorithm is *oblivious arithmetic* if two conditions hold. (i) Every operation that writes a numerical cell, other than a copy between numerical cells, takes operands from numerical cells or real constants and comes from $\{+, -, \times, /, \exp, \ln\}$. (ii) The sequence of operations it executes on a given input depends only on the input shape, and not on the values of numerical cells. In addition, the following conventions apply. Word cells, which hold w-bit words, and numerical cells, which hold real numbers, occupy separately addressed memories, and the real input occupies numerical cells. The initial contents of word memory depend only on the input shape. Control flow, and every address at which a cell is read or written, depend on word cells alone. No operation reads a numerical cell and writes a word cell, so in particular the algorithm performs no comparison between real values. Every numerical cell the algorithm reads, the output cells included, is an input coordinate, the output of an earlier numerical operation, or a real constant fixed by the input shape. No numerical cell other than the input cells is preloaded with an input-dependent value. The output coordinates are read, when the algorithm halts, from a designated output region of numerical cells disjoint from the cells holding the input. Division by zero and the logarithm of a nonpositive argument are undefined, so an execution attempting one is not a valid run. The other conventions are repeated in Definition 3.1, which admits comparisons and so relaxes both clause (ii) and the convention that no operation reads a numerical cell and writes a word cell.

Proposition B.2 (Oblivious arithmetic Word-RAM bound)**.** *Suppose $\omega > 2$. Let T be the transformer of Theorem 2.4. Every oblivious arithmetic Word-RAM algorithm that computes T uses time at least*

$$LH \cdot N^{\omega - o(1)}.$$

Proof of Proposition B.2. Let A be an oblivious arithmetic Word-RAM algorithm computing T. We read the trace of A as an eAC computing T and apply the eAC lower bound of Theorem 2.4 to it. Fix the input shape: input length $N + 1$, embedding dimension $2N + 3$, and L layers of H heads each. By Definition B.1, the sequence of operations A executes depends only on this shape, not on the values of numerical cells. Hence the trace of A on every input of this shape is a fixed sequence of operations

$$\mathcal{S} = \mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{T_A},$$

where T_A is the running time of A on inputs of the fixed shape.

Step 1. The trace gives rise to an eAC. Split the operations of $\mathcal{S}$ into three kinds. The *word operations* act only on word cells. The copies act between numerical cells. The *numerical operations* act on numerical cells via gates from the alphabet $\{+, -, \times, /, \exp, \ln\}$. Every operation that writes a cell is of one of these kinds, for two reasons. By Definition B.1(i), every operation that writes a numerical cell is a numerical operation or a copy between numerical cells. By the convention of Definition B.1 that no operation reads a numerical cell and writes a word cell, every operation that writes a word cell is a word operation. An operation that writes no cell, such as a jump, contributes no value. Copies are treated in the construction below. Let

$$\mathcal{S}_{i_1}, \mathcal{S}_{i_2}, \ldots, \mathcal{S}_{i_{s_A}}$$

be the subsequence of numerical operations, listed in the order they appear in $\mathcal{S}$, where s_A is their number.

We construct an eAC C_A of size s_A as follows. The inputs of C_A are the initial values of the numerical cells holding the input (the entries of the token matrix X and the entries of the weights W). The k-th gate of C_A corresponds to the operation $\mathcal{S}_{i_k}$:

- Its gate type is the operation performed by $\mathcal{S}_{i_k}$ (one of $+, -, \times, /, \exp, \ln$).
- Its operands are determined by which numerical cells $\mathcal{S}_{i_k}$ reads from and which real constants it takes. Each such read reaches its value directly or through a chain of copies between numerical cells, and copies contribute no gate. The value read is one of three things. (i) It is an initial numerical input, that is, an input wire of C_A. (ii) It is the output of an earlier numerical operation $\mathcal{S}_{i_j}$ with $j < k$, that is, an earlier gate of C_A. (iii) It is a real constant hard-coded into A or preloaded in numerical memory by the input shape, that is, a constant of C_A.

For each output cell, the designated wire of C_A is the value the cell holds when A halts. That wire is one of three things. If the result of a numerical operation last reached the cell, directly or through a chain of copies, the wire is the gate of that operation. If an input wire or a constant last reached the cell through copies, the wire is that input or constant. If A never writes the cell, the wire is its initial content, which by Definition B.1 is a real constant fixed by the input shape. Definition 2.2 permits an input or a constant to be designated, at no gate.

By Definition B.1(ii) and the addressing argument below, the cells that $\mathcal{S}_{i_k}$ reads and writes are determined by the input shape, and not by intermediate numerical values. This determination covers operands addressed through word cells. Word memory is initialized from the shape. By the convention of Definition B.1 that no operation reads a numerical cell and writes a word cell, word memory is never written from a numerical cell. So every address is itself a function of the shape. This fixed addressing is what makes the k-th gate of C_A well defined: an eAC gate's operands are part of the fixed circuit structure, not a function of intermediate values.

Write $T_W(X)$ for the output of T on the token matrix X with weights W. Since A computes T correctly, $C_A(X, W) = T_W(X)$ for every numerical input (X, W) in the domain of T. The evaluability domain of C_A contains every input of this shape, because a valid run of A divides by no

zero and takes the logarithm of no nonpositive argument. Hence C_A is an eAC of size s_A computing T in the sense of Definition 2.2.

Step 2. Applying the eAC lower bound. By Theorem 2.4, every eAC computing T has size at least $LH \cdot N^{\omega - o(1)}$. Since the s_A numerical operations are among the T_A operations of the trace, $T_A \geq s_A$, and we conclude

$$T_A \geq LH \cdot N^{\omega - o(1)}. \qquad \square$$

The model of Definition 3.1 has no rounding.

Remark B.3 (The interval-fiber condition). The condition that every fiber of χ be a countable union of intervals cannot be dropped from Definition 3.2. Take $\chi := \mathbf{1}_{\mathbb{Q}}$, whose fiber over 0 is the set of irrationals. That set has uncountably many connected components, each a single point, whereas a countable union of intervals has countably many. Indeed, each component of a countable union of intervals contains one of the intervals, and distinct components contain distinct ones. An eAC is continuous on its evaluability domain, so no eAC agrees with $\mathbf{1}_{\mathbb{Q}}$ on a nonempty open subset of $\mathbb{R}$, and the hypothesis of Theorem 3.3 holds for $f = \mathbf{1}_{\mathbb{Q}}$, $P_* = \mathbb{R}$ and every $s > 0$. Yet one read of this χ followed by one constant injection computes f exactly, in $O(1)$ operations.

The next proposition shows that an algorithm whose arithmetic rounds every result into a finite set cannot compute the transformer exactly on any ball.

Proposition B.4 (Finite precision computes finitely many values)**.** *Consider the variant of a numerically-branched Word-RAM algorithm with bit-reading in which each numerical operation of Definition 3.1(i), that is, each of $+, -, \times, /, \exp, \ln$, returns a value in a finite set $F \subset \mathbb{R}$. For instance, each operation may return its exact result rounded to an element of F. The set F is fixed for each input shape. Call such an algorithm* finite-precision*. The finite w-bit floating-point numbers are an example of such a set F. Copies between numerical cells, bit-reading operations (Definition 3.2), constant injections, word operations, and comparisons are unrestricted.*

Then, at every input, each coordinate of a function computed by a finite-precision algorithm is an element of F, an input coordinate, or one of finitely many real constants. Those constants are fixed by the algorithm and the input shape, and we call this three-case description the trichotomy*. Let d be the number of input coordinates and let $B \subseteq \mathbb{R}^d$ be an open ball. Let g be a continuous target with a coordinate that, on every subball of B, is nonconstant and is not equal to any single input coordinate. For such a target, no finite-precision algorithm computes g exactly on B. The softmax transformer of Theorem 2.4, as a map of the token matrix and the weights, meets the hypothesis on g for every ball B: it has a coordinate that is, on every subball, nonconstant and equal to no single input coordinate.*

Proof of Proposition B.4. We first prove the trichotomy, then derive the consequence by a covering argument, and finally check that the softmax transformer is a target of the kind the consequence covers.

Each output coordinate is read from a numerical cell (Definition 3.1(iv)). By induction over the writes to numerical cells, such a cell holds one of three things, which are the three cases of the trichotomy. It may hold the result of a numerical operation; such a result is an element of F. It may instead hold a value copied from an input cell, directly or through a chain of copies; such a value is an input coordinate. Or it may hold a real constant: one hard-coded in the program, one preloaded in numerical memory by the input shape, or one injected from a word cell. The induction goes through because a cell never written holds its initial content, which is an input coordinate or a preloaded constant, and a copy holds whatever its source held.

The constants of the third kind are finitely many and fixed by the algorithm and the input shape. The program hard-codes finitely many of them, and numerical memory, whose cells are addressed by w-bit words, holds at most 2^w constants preloaded by the input shape. The program also has finitely many injection instructions, and each writes a value determined by a w-bit word (Definition 3.2), so each can write at most 2^w values. The word size w is fixed by the input shape.

The case of a copied input value is why the hypothesis on g asks for more than nonconstancy. An algorithm that copies its input cells into the output region performs no arithmetic operation at all, and it computes the identity exactly.

Now suppose, for a contradiction, that a finite-precision algorithm A computes g exactly on B, and fix a coordinate of g as in the hypothesis. Write g_c for that coordinate; it is continuous on B, and at every point of B it equals the corresponding output coordinate of A. The trichotomy is pointwise, so which of its cases holds may vary with the input. Consider the following finitely many subsets of B. For each element a of F, take the set of points at which g_c equals a. For each of the d input coordinates z_i, take the set of points at which g_c equals z_i. For each of the finitely many constants of the third kind, take the set of points at which g_c equals that constant. By the trichotomy, these sets cover B. Each of them is closed in B, because it is the zero set in B of the continuous function $g_c - a$, $g_c - z_i$, or g_c minus the constant.

One of these finitely many closed sets has nonempty interior. Indeed, suppose that none of the sets had nonempty interior. Then the complement in B of each set would be open and dense in B. A finite intersection of open dense subsets of B is dense in B. For two such subsets O_1 and O_2 and a nonempty open set $Q \subseteq B$, the set $Q \cap O_1$ is nonempty and open, so it meets O_2. Induction on the number of subsets extends this from two subsets to finitely many. So the intersection of the complements would be nonempty, and a point in it would lie in none of the sets, although the sets cover B. Hence one of the sets contains a subball $B' \subseteq B$. On B' the coordinate g_c is then constantly equal to one element of F, or constantly equal to one fixed constant, or equal to one input coordinate z_i. The first two contradict nonconstancy on B', and the third contradicts the second half of the hypothesis, so no finite-precision algorithm computes g exactly on B.

Finally, we check that the softmax transformer of Theorem 2.4 has a coordinate that, on every subball of every ball, is nonconstant and equal to no single input coordinate. Each output entry is real-analytic on the whole input space, being the input entry in the same position plus the sum of the head outputs over all layers. Such an entry is neither constant nor identically equal to any input coordinate. To see this, consider the weight setting where one head's value weights are the identity and every other weight is zero. There the added term is the average of the rows of the token matrix. So the entry is a linear form in the token entries with at least two nonzero coefficients. Such a form is not constant and equals no single token entry. At this setting every weight coordinate is constant, equal to 0 or 1, so the entry equals no weight coordinate either. A real-analytic function on $\mathbb{R}^d$ that is not identically equal to a given real-analytic function differs from it on every ball, by the identity theorem (Theorem A.3). So the entry is nonconstant, and equal to no single input coordinate, on every subball of every ball. □

C Proofs for Section 4

We now supply the eAC-side ingredient and complete the proof of Corollary 4.1. Let $\zeta : \mathbb{R}^{D'} \to \mathbb{R}^D$ denote the affine instance map implicit in the proof of Saha et al. (2026, Theorem 5.1), where D is the number of input coordinates of T, token entries and weights together. Its number of free variables is $D' = LH(3N^2 + N)$. The heads of all layers are numbered $1, \dots, LH$. For each $k \in [LH]$ the map ζ routes the entries of free matrices $A_k, C_k \in \mathbb{R}^{N \times N}$ into the query-side slots of head k. They form the $N \times 2N$ matrix $\tilde{A}_k = (A_k | C_k)$, whose columns are those of A_k followed by those of C_k. The map routes the entries of a free matrix $B_k \in \mathbb{R}^{N \times N}$ together with identity constants into the key-side slots, forming $\tilde{B}_k = (B_k | I)$ in the same way. It routes a free vector $D_k \in \mathbb{R}^N$ into the designated value slots, and it fixes every remaining token and weight slot to the constants of their construction (Saha et al., 2026, Lemma 5.3), which we call the *template* of ζ.

Let $\Phi^{\mathrm{sm}} := T \circ \zeta$ be the transformer as a function of the free variables alone. For each head k and row $i \in [N]$, write $S_{ki} := \sum_j \exp((\tilde{A}_k \tilde{B}_k^\top)_{ij})$ for the exponential row sum of head k at row i. By Saha et al. (2026, Lemma 5.3), the output of Φ^{sm} contains the entries

$$Y_{i,1} = \sum_{k=1}^{LH} \frac{S_{ki}}{1 + S_{ki}} \qquad \text{and} \qquad Y_{i,j+1} = \sum_{k=1}^{LH} \frac{D_k[j]}{1 + S_{ki}} \qquad (i, j \in [N]),$$

identically in the free variables.

The ingredient has two steps. The first, Lemma C.1, converts an eAC that agrees with a low-degree polynomial map on an open set into an arithmetic circuit that computes the map everywhere. The second, Lemma C.3, uses that conversion to prove the regional eAC bound for Φ^{sm}.

Lemma C.1 (Local eAC-to-AC for low-degree polynomials)**.** *Let $d \geq 2$ be a constant. Let $f : \mathbb{R}^n \to \mathbb{R}^k$ be a map each of whose components is a polynomial of degree at most d. Suppose there is an eAC C of size s with evaluability domain $V \subseteq \mathbb{R}^n$, and a nonempty open set $U \subseteq V$ such that $C(x) = f(x)$ for every $x \in U$. Then there is an arithmetic circuit (using only $\{+, -, \times\}$, no $/, \exp, \ln$ gates) of size $O(d^2(s+n))$ computing f on all of $\mathbb{R}^n$. For constant d this is $O(s+n)$.*

Proof of Lemma C.1. The proof shifts coordinates so that a point of U becomes the origin, runs the formal-power-series construction of Saha et al. (2026, Theorem 4.1) from that base point with the series truncated at degree d, and shifts back. Fix a point $x_0 \in U$ and define the shifted map $\tilde{f} : \mathbb{R}^n \to \mathbb{R}^k$ by $\tilde{f}(y) := f(x_0 + y)$. The shifted map is again a polynomial map of degree at most d. Let $\tilde{C}$ be the eAC obtained from C by prepending a gate to each input wire i. That gate adds $x_0^{(i)}$, the ith coordinate of x_0. Then $|\tilde{C}| = s + n$, and the evaluability domain of $\tilde{C}$ is $\tilde{V} := V - x_0$. The circuit $\tilde{C}$ agrees with $\tilde{f}$ on the nonempty open neighborhood $\tilde{U} := U - x_0$ of the origin.

At $y = 0$ (corresponding to $x = x_0 \in V$), every $/$ gate of $\tilde{C}$ has a nonzero denominator and every $\ln$ gate has a positive argument. The formal-power-series construction in the proof of Saha et al. (2026, Theorem 4.1) uses only the values of the gates at the origin together with their finite-degree expansions in an auxiliary real parameter z. The construction therefore goes through with x_0 as the base point, with two changes. The series are truncated at degree d rather than at degree 2. And since f need not be homogeneous, the output is the sum of the $d+1$ coefficient functions rather than the coefficient of z^2, as described below.

Each gate then maintains $d+1$ coefficient functions, and each gate operation costs $O(d^2)$ arithmetic operations on them. For $\times$ these are truncated Cauchy products. For the inverse, the exponential, and the logarithm of a power series $a = \sum_j a_j z^j$ they are the order-d recurrences obtained by comparing coefficients, for $1 \leq j \leq d$, in $a \cdot b = 1$, in $e' = a'e$ and in $a\lambda' = a'$. The inverse b has $b_0 = 1/a_0$ and $b_j = -b_0 \sum_{i=1}^{j} a_i b_{j-i}$. The exponential e has $e_0 = \exp(a_0)$ and $e_j = \sum_{i=1}^{j} (i/j)\, a_i e_{j-i}$. The logarithm λ has $\lambda_0 = \ln(a_0)$ and $\lambda_j = \left(a_j - \sum_{i=1}^{j-1} (i/j)\, \lambda_i a_{j-i}\right) / a_0$. Each coefficient takes $O(d)$ operations, and a division gate multiplies by the inverse of its denominator. The recurrence for the inverse needs a nonzero constant term, the one for the logarithm needs a positive constant term, and the one for the exponential needs nothing. The constant terms are the gate values at $y = 0$, which meet these conditions as shown above. The recurrences use only $+$, $-$, $\times$, and multiplication by real constants. The constants are the constant terms themselves, their reciprocals, exponentials and logarithms, and the rationals p/q for $1 \leq p \leq q \leq d$. So the coefficient functions are polynomials in y computed by an arithmetic circuit over $\{+, -, \times\}$.

Moreover, for every y in a ball $B \subseteq \tilde{U}$ around the origin, the coefficient functions of a gate are the Taylor coefficients in z at $z = 0$ of the gate's value at the point zy. The reason is twofold. First, $zy \in B \subseteq \tilde{V}$ for $|z| \leq 1$. Second, the recurrences are the power-series identities for $\times$, $/$, $\exp$, and $\ln$ at a point where each operation is defined.

Let $\tilde{D}$ be the arithmetic circuit that computes these coefficient functions and, for each designated wire, outputs the sum of the wire's $d+1$ coefficient functions, at $O(d)$ further gates per wire. For $y \in B$, the value of a designated wire at zy is the corresponding coordinate of $\tilde{f}(zy)$. As a function of z this coordinate is a polynomial of degree at most d. Its coefficients are the homogeneous components of that coordinate evaluated at y, and its value at $z = 1$ is the coordinate of $\tilde{f}(y)$. Hence the sum of the coefficient functions agrees with $\tilde{f}$ on B, and, both sides being polynomials in y, on all of $\mathbb{R}^n$. The designated wires that are gates or shifted inputs number at most $s + n$, and a designated constant is output as itself, so $\tilde{D}$ is an arithmetic circuit of size $O(d^2(s+n))$ computing $\tilde{f}$ on all of $\mathbb{R}^n$. Compose $\tilde{D}$ with the inverse shift $x \mapsto x - x_0$, which adds n subtraction gates. The result is an arithmetic circuit D of size $O(d^2(s+n))$ computing f on all of $\mathbb{R}^n$. □

The lemma is a local form of a conversion that Saha et al. (2026) prove globally, and the next remark compares the two.

Remark C.2 (Comparison with the eAC-to-AC conversion of Saha et al. (2026)). The eAC-to-AC conversion of Saha et al. (2026, Theorem 4.1) establishes the corresponding statement for homogeneous quadratic forms, and under the stronger hypothesis that the eAC C is evaluable on all of $\mathbb{R}^n$.

Their proof generalizes the formal-power-series argument of Strassen (1973a) from $\{+, -, \times, /\}$-circuits to eACs by handling $\exp$ and $\ln$ gates as additional cases in the inductive construction of each gate's expansion in the auxiliary parameter. The construction depends only on the value of each gate at a single base point (which they take to be the origin) and on local power-series identities; no other property of the global evaluability domain is used.

Our Lemma C.1 makes this locality explicit. Shifting a base point $x_0 \in U$ to the origin gives a circuit that is evaluable at the origin and agrees with the shifted target on a neighborhood of it. These are the two facts about the base point that the construction uses. The first supplies the constants of the recurrences, and the second, with a polynomial identity, supplies the correctness of the output. The conditions Saha et al. (2026) impose at the base point, nonzero denominators for $/$ gates and positive arguments for $\ln$ gates, are exactly the conditions for x_0 to lie in the evaluability domain V.

The adaptation has three costs. It adds $2n$ shift gates, n before the construction and n after it. Truncating the series at degree d costs a factor $O(d^2)$. And each output becomes a sum of its $d + 1$ coefficient functions.

With the local conversion in hand, we prove the regional eAC bound for the transformer.

Lemma C.3 (Local eAC lower bound for the transformer of Saha et al. (2026))**.** *Let $\omega > 2$. Every eAC whose evaluability domain contains a nonempty open set $U \subseteq \mathbb{R}^{D'}$ and which agrees with Φ^{sm} on U has size at least $LH \cdot N^{\omega - o(1)} - O(LH \cdot N^2)$.*

Proof of Lemma C.3. We first derive an identity that recovers every product entry $(A_k B_k^\top)_{ij}$ from two first derivatives of the output. We then use the Extended Baur–Strassen Theorem to compute those derivatives inside every eAC that agrees with Φ^{sm} on U, so that the matrix-multiplication lower bound applies. Consider the scalar readouts $f := \sum_i Y_{i,1}$ and $g := \sum_i Y_{i,i+1}$, each a sum of N output entries. Since $(\tilde{A}_k \tilde{B}_k^\top)_{ij} = (A_k B_k^\top)_{ij} + C_k[i,j]$, the chain rule gives, identically in the free variables,

$$\frac{\partial f}{\partial C_k[i,j]} = \frac{\partial S_{ki}/\partial C_k[i,j]}{(1+S_{ki})^2} = \frac{\exp\left((A_k B_k^\top)_{ij} + C_k[i,j]\right)}{(1+S_{ki})^2}, \qquad \frac{\partial g}{\partial D_k[i]} = \frac{1}{1+S_{ki}},$$

exactly as computed in the proof of Saha et al. (2026, Theorem 5.1). Both derivatives are strictly positive at every point of $\mathbb{R}^{D'}$, and consequently their logarithms satisfy

$$\ln \frac{\partial f}{\partial C_k[i,j]} - 2 \ln \frac{\partial g}{\partial D_k[i]} - C_k[i,j] = (A_k B_k^\top)_{ij}.$$

This identity holds for every value of the auxiliary matrices C_k.

Fix a nonempty open $U \subseteq \mathbb{R}^{D'}$ and an eAC as in the statement, with evaluability domain containing U and agreeing with Φ^{sm} on U, and let s be its size. Appending the two readouts ($O(N)$ addition gates) yields eACs of size $s + O(N)$ agreeing with f and with g on U. Apply the Extended Baur–Strassen Theorem (Theorem A.1) to each, with $n = D' = LH(3N^2 + N)$ inputs. It gives an eAC of size $O(s + LH \cdot N^2)$ that computes all first-order partials of the two circuit functions, with an evaluability domain containing the original one. The two circuit functions agree with f and g on the open set U, and both are continuously differentiable there. So the computed partials agree with $\partial f/\partial C_k[i,j]$ and $\partial g/\partial D_k[i]$ on U. On U these values are strictly positive, so $\ln$ gates may be applied to them. Append $O(LH \cdot N^2)$ further $\ln$, subtraction, and constant-multiplication gates, including one subtraction of the *input variable* $C_k[i,j]$ per output. By the logarithmic identity displayed above, the result is an eAC of size $O(s + LH \cdot N^2)$ whose outputs on U are the quantities $(A_k B_k^\top)_{ij}$ for all k, i, j.

The map so computed is a polynomial map of degree 2 in the inputs, constant in the C- and D-variables. So Lemma C.1 yields an arithmetic circuit over $\{+, -, \times\}$ of size $O(s + LH \cdot N^2)$ computing it on all of $\mathbb{R}^{D'}$. Substituting arbitrary constants for the C- and D-variables (a substitution into a division-free circuit, adding no gates) leaves an arithmetic circuit computing the LH independent matrix products $(A_k B_k^\top)_k$ of pairwise disjoint indeterminate matrices. The matrix-multiplication lower bound (Theorem A.2) then gives $O(s + LH \cdot N^2) \geq LH \cdot N^{\omega - o(1)}$. That is, $s \geq LH \cdot N^{\omega - o(1)} - O(LH \cdot N^2)$, the absolute constant in the $O(\cdot)$ being absorbed into the $N^{-o(1)}$. The absorption is legitimate even though the number $r = LH$ of products may grow with N, for

two reasons. The constant depends on none of N, L and H. And the $o(1)$ of Theorem A.2 does not depend on r. □

Remark C.4 (Deferring the slice substitution). The proof of Saha et al. (2026, Theorem 5.1) substitutes $C \leftarrow 0$ into the derivative circuit before taking logarithms. The regional extraction retains C as an input and subtracts it after taking logarithms. It then applies Lemma C.1 and specializes the resulting division-free circuit. This order makes every intermediate operation valid on the original region of agreement, including regions disjoint from the slice $C = 0$.

The regional bound and Theorem 3.3 together give the corollary.

Corollary 4.1. *Suppose $\omega > 2$. Let T be the transformer of Theorem 2.4 with softmax attention. Every numerically-branched Word-RAM algorithm with bit-reading computing T uses time at least*

$$LH \cdot N^{\omega - o(1)} - O(LH \cdot N^2).$$

Proof of Corollary 4.1. Let $\mathcal{A}$ be a numerically-branched Word-RAM algorithm with bit-reading computing $(X, W) \mapsto T_W(X)$ in worst-case time $T_{\mathcal{A}}$. The input consists of the token matrix and the attention weights. We turn $\mathcal{A}$ into an algorithm computing Φ^{sm} at an additive cost of $O(LH \cdot N^2)$, and then apply Theorem 3.3 to it, with the regional bound of Lemma C.3 as its hypothesis.

Hardwire the affine instance map ζ into $\mathcal{A}$ without new cells, on inputs of shape (N, L, H), as follows. For each free variable, take as its input cell one cell of the input region of $\mathcal{A}$ into which ζ routes it. Preload every input cell of $\mathcal{A}$ to which ζ assigns a constant with that constant. Preload every further slot into which ζ routes a free variable with 0. Definition 3.1(iv) allows both preloadings, since these cells are no longer inputs. A straight-line prefix, whose addresses are fixed by the shape, copies each free variable into its further slots, if any, at $O(LH \cdot N^2)$ copies in all. The prefix makes the program depend on the shape (N, L, H). The application of Theorem 3.3 fixes this shape throughout. This yields a numerically-branched Word-RAM algorithm with bit-reading that uses the cells and the word size of $\mathcal{A}$ and computes Φ^{sm} on $\mathbb{R}^{D'}$ in time $T_{\mathcal{A}} + O(LH \cdot N^2)$.

By Lemma C.3, every eAC agreeing with Φ^{sm} on a nonempty open subset of $\mathbb{R}^{D'}$ has size at least $LH \cdot N^{\omega - o(1)} - O(LH \cdot N^2)$. So Theorem 3.3 with $P_* = \mathbb{R}^{D'}$ gives $T_{\mathcal{A}} + O(LH \cdot N^2) \geq LH \cdot N^{\omega - o(1)} - O(LH \cdot N^2)$, and the claim follows. □

The same method extends to transformers whose MLPs are not the identity.

Corollary C.5 (Sigmoid- and GLU-MLP transformers in numerically-branched Word-RAM)**.** *Suppose $\omega > 2$.*

- *(a) Let T' be the transformer of Theorem 2.4 with its identity MLPs replaced by the gated linear units of Saha et al. (2026, Proposition B.2), for an arbitrary activation σ with $\sigma(c) \neq 0$ for some c. Then every numerically-branched Word-RAM algorithm with bit-reading computing T' uses time at least $LH \cdot N^{\omega - o(1)} - O(LH \cdot N^2)$.*
- *(b) Let T'' be the transformer of Theorem 2.4 with its identity MLPs replaced by the one-hidden-layer perceptrons of Saha et al. (2026, Proposition B.8), for an activation σ that is real-analytic on $\mathbb{R}$ and eAC gradient efficient in the sense of Saha et al. (2026, Definition B.7). That is, $\sigma(0) \neq \sigma(1)$, and there is a function h with $\sigma' = h \circ \sigma$ such that σ, σ^{-1} and h are computable by eACs of size $O(1)$. Then the same bound holds for T''.*

Proof of Corollary C.5. Part (a). By Saha et al. (2026, Proposition B.2), the gated linear unit with the weights chosen there computes the identity on all of $\mathbb{R}^m$. So T' and the transformer T of Theorem 2.4 are the same function of the token matrix and the attention weights, the MLP weights being fixed. An algorithm computing T' therefore computes T, and Corollary 4.1 gives the bound. The identity follows from $x \mapsto (x/z) \odot (\sigma(c)\mathbf{1}) = x$ for $z := \sigma(c)$.

Part (b). We apply the unparametrized form of Theorem 3.3, with $P_* = \mathbb{R}^{D'}$, and verify its hypothesis. The hypothesis requires a *regional* eAC bound: a bound on every eAC agreeing with $\Phi'' := T'' \circ \zeta$ on a nonempty open set that its evaluability domain contains, where ζ is the instance map of Appendix C. We carry out the layer-peeling extraction of Saha et al. (2026, Lemma B.10) regionally, with the substitution $C \leftarrow 0$ deferred at every layer as in Lemma C.3.

What Φ'' computes. The perceptron weights are fixed at the values of Saha et al. (2026, Proposition B.8), as the MLP weights are in part (a). The perceptron of Saha et al. (2026, Proposition B.8) applies σ entrywise to the first $N+1$ entries of its row and, to the remaining entries, an affine function of σ that fixes 0 and 1. On the template of ζ the remaining columns hold only 0 and 1 and no head writes into them, so they are preserved at every layer (Saha et al., 2026, Claim B.9). Since every head reads only those columns, each head k outputs at every layer the matrix $Y^{(k)}$ of Saha et al. (2026, Lemma 5.2). In the rows $i \in [N]$ its entries are $Y^{(k)}[i,1] = S_{ki}/(1+S_{ki})$ and $Y^{(k)}[i,j+1] = D_k[j]/(1+S_{ki})$, with S_{ki} the exponential row sum defined at the start of Appendix C.

Fix $i \in [N]$ and a column $c \in \{1, i+1\}$. Write $u_{i,c}^{(\ell)}$ for the entry (i,c) of the stream entering the perceptron of layer ℓ, and $x_{i,c}^{(\ell)} := \sigma(u_{i,c}^{(\ell)})$ for the entry leaving it, for $\ell \in [L]$. Let $x_{i,c}^{(0)}$ denote the entry (i,c) of the input token matrix. Then $x_{i,c}^{(0)} = 0$, because the first $N+1$ columns of the template are zero, and

$$u_{i,c}^{(\ell)} = x_{i,c}^{(\ell-1)} + \sum_{k \text{ in layer } \ell} Y^{(k)}[i,c], \qquad \ell \in [L].$$

The output entries of Φ'' are $Y_{i,c} = x_{i,c}^{(L)}$, and the readouts are $f := \sum_i x_{i,1}^{(L)}$ and $g := \sum_i x_{i,i+1}^{(L)}$. Each $u_{i,c}^{(\ell)}$ is real-analytic on $\mathbb{R}^{D'}$, being built from the activation σ, the exponential, sums, and divisions by $1+S_{ki} > 0$. It depends on the variables of a head k only when k lies in a layer $t \le \ell$. Since the perceptron acts entrywise, the entry (i,c) of the stream is fed by the entries (i,c) of the head outputs alone.

For a head k of layer ℓ and $j \in [N]$, the chain rule through the layers $\ell, \dots, L$ therefore gives, identically in the free variables,

$$\frac{\partial f}{\partial C_k[i,j]} = \pi_{i,1}^{(\ell)} \cdot \frac{\exp\big((A_k B_k^\top)_{ij} + C_k[i,j]\big)}{(1+S_{ki})^2}, \qquad \frac{\partial g}{\partial D_k[i]} = \pi_{i,i+1}^{(\ell)} \cdot \frac{1}{1+S_{ki}},$$
$$\pi_{i,c}^{(\ell)} := \prod_{t=\ell}^{L} \sigma'(u_{i,c}^{(t)}).$$

Here $\pi_{i,c}^{(\ell)}$, defined on the second line, is the product of the activation derivatives at the entries (i,c) entering the perceptrons of layers ℓ through L. The two formulas rest on four facts. First, the variable $C_k[i,j]$ enters the summands of f only through $Y^{(k)}[i,1]$. Likewise, the variable $D_k[i]$ enters the summands of g only through $Y^{(k)}[i,i+1]$. Second, neither $x_{i,c}^{(\ell-1)}$ nor a head of layer ℓ other than k depends on these variables. Third, the layers above ℓ see them only through $x_{i,c}^{(\ell)}$. Fourth, the second factor of each formula is the derivative of $Y^{(k)}[i,c]$ computed in Lemma C.3.

A region where the activation derivatives do not vanish. The extraction below divides by the products $\pi_{i,c}^{(\ell)}$, so we first shrink the region of agreement to one on which none of them vanishes. Let $U \subseteq \mathbb{R}^{D'}$ be a nonempty open set on which some eAC agrees with Φ'', as in the regional bound that the hypothesis needs. Let $Z \subseteq \mathbb{R}$ be the zero set of σ'. Since σ is real-analytic and $\sigma(0) \neq \sigma(1)$, the derivative σ' is real-analytic and not identically zero, so Z is closed and discrete. Each $u_{i,c}^{(t)}$ is nonconstant on $\mathbb{R}^{D'}$, for the following reason. Take any head k of layer t. The entry $u_{i,1}^{(t)}$ is strictly increasing in $C_k[i,1]$, since S_{ki} is strictly increasing in $C_k[i,1]$ and the map $S \mapsto S/(1+S)$ is strictly increasing. Likewise, $u_{i,i+1}^{(t)}$ is strictly increasing in $D_k[i]$, with slope $1/(1+S_{ki}) > 0$.

Hence the closed set $E_{t,i,c} := \{z \in \mathbb{R}^{D'} : u_{i,c}^{(t)}(z) \in Z\}$ has empty interior. Indeed, suppose an open ball lay inside it. The image of that ball under $u_{i,c}^{(t)}$ would be connected and would lie in the discrete set Z, so it would be a single point. Then $u_{i,c}^{(t)}$ would be constant on the ball and, by the identity theorem (Theorem A.3), on all of $\mathbb{R}^{D'}$, which it is not. A nonempty open set minus a closed set with empty interior is nonempty and open. So removing the finitely many $E_{t,i,c}$ from U one at a time leaves a nonempty open $U' \subseteq U$ on which every $\sigma'(u_{i,c}^{(t)})$, and hence every $\pi_{i,c}^{(\ell)}$, is nonzero.

The extraction. Let $\mathcal{C}$ be an eAC of size s with evaluability domain containing U and agreeing with Φ'' on U. Let U' be as above. Append the readouts f and g and apply Theorem A.1 to each, as in Lemma C.3. The result is an eAC of size $O(s + LH \cdot N^2)$, with evaluability domain containing U, that computes on U all partials $\partial f/\partial C_k[i,j]$ and $\partial g/\partial D_k[i]$. A copy of $\mathcal{C}$ kept beside it, at s further gates, carries the outputs $Y_{i,1}$ and $Y_{i,i+1}$.

The maps σ^{-1} and h are defined on the range $\sigma(\mathbb{R})$ of the activation. The two identities $\sigma^{-1} \circ \sigma = \mathrm{id}$ and $\sigma' = h \circ \sigma$, which hold on $\mathbb{R}$, require this. An eAC computes a map only on a set inside its evaluability domain (Definition 2.2), so the eACs of size $O(1)$ for σ^{-1} and for h are evaluable on $\sigma(\mathbb{R})$. Every argument fed below to the eACs for σ^{-1} and for h is a value $x_{i,c}^{(\ell)}$ of σ with $\ell \geq 1$. We append gates layer by layer, from layer L down to layer 1. At each layer ℓ we maintain wires carrying $x_{i,c}^{(\ell)}$ and $\pi_{i,c}^{(\ell)}$, for all $i \in [N]$ and $c \in \{1, i+1\}$. By the identities above and induction over the appended gates, every gate is evaluable on U' and every wire named carries the stated function on U'.

At the start, $x_{i,c}^{(L)}$ is the output $Y_{i,c}$ and $\pi_{i,c}^{(L)} = \sigma'(u_{i,c}^{(L)}) = h(x_{i,c}^{(L)})$, at $O(N)$ gates. At layer ℓ, for each head k of the layer and all $i, j \in [N]$, form two quotients. They are $q_{k,i,j} := (\partial f/\partial C_k[i,j])/\pi_{i,1}^{(\ell)}$ and $r_{k,i} := (\partial g/\partial D_k[i])/\pi_{i,i+1}^{(\ell)}$. Both divisions are legal on U', and there the quotients equal the two strictly positive fractions displayed above. Then $(A_k B_k^\top)_{ij} = \ln q_{k,i,j} - 2\ln r_{k,i} - C_k[i,j]$ on U', exactly as in Lemma C.3, at $O(HN^2)$ gates for the layer. The head readouts are $Y^{(k)}[i,1] = 1 - r_{k,i}$ and $Y^{(k)}[i,i+1] = D_k[i] \cdot r_{k,i}$, since $S/(1+S) = 1 - 1/(1+S)$. This is the source's recomputation of the layer's readouts, with S_{ki} read off $r_{k,i}$ at generic C rather than re-exponentiated from the recovered products at $C = 0$.

If $\ell \geq 2$, three assignments pass from layer ℓ to layer $\ell - 1$. First, $u_{i,c}^{(\ell)} := \sigma^{-1}(x_{i,c}^{(\ell)})$. Second, $x_{i,c}^{(\ell-1)} := u_{i,c}^{(\ell)} - \sum_{k \text{ in layer } \ell} Y^{(k)}[i,c]$. Third, $\pi_{i,c}^{(\ell-1)} := h(x_{i,c}^{(\ell-1)}) \cdot \pi_{i,c}^{(\ell)}$. The three take $O(HN)$ gates, and they are the induction step of Saha et al. (2026, Lemma B.10). The factor $h(x_{i,c}^{(\ell-1)})$ equals $\sigma'(u_{i,c}^{(\ell-1)})$ because $x_{i,c}^{(\ell-1)} = \sigma(u_{i,c}^{(\ell-1)})$. After layer 1 the circuit has size $O(s + LH \cdot N^2)$ and its evaluability domain contains U'. Its outputs on U' are $(A_k B_k^\top)_{ij}$ for all k, i, j.

Closing. The map computed on U' is a polynomial map of degree 2 in the inputs, constant in the C- and D-variables. So Lemma C.1 yields an arithmetic circuit of size $O(s + LH \cdot N^2)$ computing it on all of $\mathbb{R}^{D'}$. Substituting constants for the C- and D-variables and applying Theorem A.2 to the LH independent products gives $s \geq LH \cdot N^{\omega - o(1)} - O(LH \cdot N^2)$, as at the end of Lemma C.3. The hypothesis of Theorem 3.3 therefore holds for Φ'' with $P_* = \mathbb{R}^{D'}$ and this s, and hardwiring ζ as in the proof of Corollary 4.1 gives the stated bound. □

A rectified-linear MLP with $W_2 W_1 = I$ and $W_2 b_1 + b_2 = 0$ computes the identity wherever all its rectifiers are active, so for it Theorem 3.3 is applied on a region P_* where every rectifier is active.

Corollary C.6 (ReLU-MLP transformers in numerically-branched Word-RAM)**.** *Suppose $\omega > 2$. Let T^{ReLU} be the transformer whose attention layers are those of Theorem 2.4, with the free variables of the affine instance map $\zeta : \mathbb{R}^{D'} \to \mathbb{R}^D$ of Appendix C in their slots, and whose MLPs are the row-wise rectified-linear maps $x \mapsto W_2 \mathrm{ReLU}(W_1 x + b_1) + b_2$ with parameters fixed so that*

(i) $W_2 W_1 = I$ and $W_2 b_1 + b_2 = 0$, and

(ii) the set

$$P_* := \{z \in \mathbb{R}^{D'} : \textit{every rectified-linear pre-activation is positive at the instance } z\}$$

is nonempty; it is then open, each pre-activation being a continuous function of the instance.

Then every numerically-branched Word-RAM algorithm with bit-reading computing T^{ReLU} uses time at least

$$LH \cdot N^{\omega - o(1)} - O(LH \cdot N^2).$$

Proof of Corollary C.6. On P_*, every rectifier acts as the identity, so each MLP acts as $x \mapsto W_2(W_1x + b_1) + b_2 = x$ by (i). Hence $T^{\mathrm{ReLU}}|_{P_*}$ equals the restriction to P_* of $\Phi^{\mathrm{sm}} = T^{\mathrm{id}} \circ \zeta$, where T^{id} is the identity-MLP softmax transformer of Theorem 2.4.

We verify the hypothesis of Theorem 3.3 with the same P_* and with $s := LH \cdot N^{\omega - o(1)} - O(LH \cdot N^2)$. Let $\mathcal{C}$ be an eAC whose evaluability domain contains a nonempty open subset $U \subseteq P_*$ with $\mathcal{C} = T^{\mathrm{ReLU}}$ on U. On U we have $T^{\mathrm{ReLU}} = \Phi^{\mathrm{sm}}$, so $\mathcal{C}$ agrees on U with Φ^{sm}. Hence Lemma C.3 gives $|\mathcal{C}| \geq s$ directly.

The hypothesis of Theorem 3.3 is thus satisfied. The target T^{ReLU} is already expressed in the free variables of ζ, so the stated Word-RAM bound follows. □

The corollary asks for two conditions on the parameters, and the next remark shows that they can be met together.

Remark C.7 (The two conditions are simultaneously satisfiable, and the degenerate case). Conditions (i) and (ii) of Corollary C.6 are compatible. Take $W_1 = W_2 = I$, $b_1 = c\mathbf{1}$, and $b_2 = -c\mathbf{1}$, where the real constant c is fixed below. The MLP is then $x \mapsto \mathrm{ReLU}(x + c\mathbf{1}) - c\mathbf{1}$, which satisfies (i). It acts as the identity exactly where every coordinate of the stream is at least $-c$. Fix an instance z_0 and take c larger than the largest absolute coordinate, over all layers, of the stream of the identity-MLP transformer T^{id} at the input $\zeta(z_0)$. Then every pre-activation is positive at z_0. The proof is by induction on the layers: once the rectifiers of the earlier layers act as identities, the stream entering layer ℓ is that of T^{id}, whose coordinates exceed $-c$. Every pre-activation being a continuous function of the instance, P_* is then a nonempty open neighborhood of z_0, so (ii) holds. This example gives a piecewise analytic target with a hard open region. Indeed, in the last layer an output entry $(i, j+1)$ is $\max(u, -c)$ for the entry u entering the perceptron, and u is affine and strictly increasing in $D_k[j]$ for a head k of that layer, so it takes every real value, and the output has a corner where $u = -c$, at which it is not real-analytic.

A second construction implements the identity globally. Suppose the parameters are chosen so that the rectified-linear MLP computes the identity on all of $\mathbb{R}^m$. This happens for the classical pair $W_1 = (I| - I)^\top$ and $W_2 = (I| - I)$, for which $\mathrm{ReLU}(x) - \mathrm{ReLU}(-x) = x$ identically. For this pair both conditions fail: (i) because $W_2W_1 = 2I$, and (ii) because the two pre-activations x and $-x$ are never simultaneously positive. The target then equals Φ^{sm} on all of $\mathbb{R}^{D'}$, however, so Lemma C.3 and the unparametrized form of Theorem 3.3 apply directly, with the same conclusion.

Remark C.8 (Perceptron constructions and regional extraction). The GLU case uses the identity realization of Saha et al. (2026, Proposition B.2); the ReLU case uses identity on an open region. Both then apply Lemma C.3. For the sigmoid case, Saha et al. (2026, Proposition B.8, Claim B.9, and Lemma 5.2) supply the perceptron weights, preservation of the template columns, and single-head readouts. These give nested compositions through the layers. The proof of Corollary C.5(b) adapts the layer-peeling induction of Saha et al. (2026, Lemma B.10) to an open set with nonzero activation derivatives, keeping the auxiliary matrices free at each layer. The extraction recovers all products with $O(s + LH \cdot N^2)$ gates from a circuit of size s.

The last corollary concerns matrix multiplication itself.

Corollary 4.2 (Matrix multiplication in numerically-branched Word-RAM). *Suppose $\omega > 2$. Every numerically-branched Word-RAM algorithm with bit-reading computing the matrix product AB for $A, B \in \mathbb{R}^{N \times N}$ uses time at least*

$$N^{\omega - o(1)}.$$

Proof of Corollary 4.2. We verify the hypothesis of Theorem 3.3 for the matrix product, with $P_* = \mathbb{R}^{2N^2}$. The matrix-product map $(A, B) \mapsto AB$ is bilinear, hence a polynomial map of degree 2 on $\mathbb{R}^{2N^2}$. By Theorem A.2, every arithmetic circuit over $\{+, -, \times\}$ computing the matrix product has size at least $N^{\omega - o(1)}$. Since the matrix product is a quadratic polynomial map, Lemma C.1 extends this bound to every eAC whose evaluability domain contains a nonempty open subset of $\mathbb{R}^{2N^2}$ on which the eAC agrees with the product. Indeed, Lemma C.1 turns such an eAC of size s into an arithmetic circuit of size $O(s + N^2)$ computing the product, so $O(s + N^2) \geq N^{\omega - o(1)}$. That is, $s \geq N^{\omega - o(1)} - O(N^2)$ with the constant factor absorbed into the $o(1)$. Since $\omega > 2$, this

is $N^{\omega-o(1)}$. The hypothesis of Theorem 3.3 is thus satisfied with $P_* = \mathbb{R}^{2N^2}$ and this s, and the bound transfers to numerically-branched Word-RAM. □