
Constant Bit-size Transformers Are Turing Complete

Qian Li*

Shenzhen International Center For Industrial And Applied Mathematics,
Shenzhen Research Institute of Big Data
Shenzhen, China
liqian.ict@gmail.com

Yuyi Wang*

CRRC Zhuzhou Institute & Tengen Intelligence Institute
Zhuzhou, China
yuyiwang920@gmail.com

Abstract

We prove that any Turing machine running on inputs of arbitrary length can be simulated by a constant bit-size transformer, as long as the context window is sufficiently long. This improves previous works, which require scaling up either the model’s precision or the number of parameters on longer inputs. Furthermore, we prove that the complexity class $\text{SPACE}[s(n)]$ exactly characterizes the expressive power of a constant bit-size transformer with a context window of length $s(n)$. Our approach relies on simulating Post machines, a Turing-complete computational model. Post machines can be modeled as automata equipped with a queue, exhibiting computational behaviors naturally aligned with those of transformers. The behavioral similarity between transformers and Post machines may offer new insights into the mechanisms underlying the reasoning abilities of transformers.

1 Introduction

Transformer-based large language models (LLMs) with chain-of-thought (CoT) steps [Achiam et al., 2023, Gemini et al., 2023, Anthropic, 2024, Dubey et al., 2024, Guo et al., 2025] have demonstrated exceptional capabilities across a wide range of complex reasoning tasks, such as mathematical problem solving and code generation [Hendrycks et al., 2021, Jimenez et al., 2023, Rein et al., 2023]. These remarkable empirical successes raise a fundamental question: What enables transformers to support such general reasoning capabilities, and what are the limits of this capability? Understanding this question is not only of theoretical interest but also of practical significance for further enhancing the reasoning abilities of transformers.

Motivated by this fundamental question, a line of theoretical works [Pérez et al., 2021, Bhattamishra et al., 2020, Merrill and Sabharwal, 2023, 2024, Li et al., 2024, Qiu et al., 2024, Zubić et al., 2024, Merrill and Sabharwal, 2025, Yang et al., 2025] have studied the reasoning abilities of transformers through the lens of expressiveness. A central result from these works is that transformers (using CoT steps) are Turing complete; that is, transformers have sufficient expressive power to simulate arbitrary Turing machines (TMs). However, in all these results, the bit-size of the transformers (i.e., the total number of bits representing the model) is not constant: either the precision bit [Pérez et al., 2021, Bhattamishra et al., 2020, Merrill and Sabharwal, 2024, Qiu et al., 2024] or the embedding dimension [Li et al., 2024] must grow with the input length, meaning that larger models are required to handle longer inputs. This scaling requirement raises a fundamental barrier to approaching transformer-based

*The authors are listed in alphabetical order.

Table 1: Comparing the existing Turing-completeness proofs in terms of required precision, embedding dimension, effective window size, and CoT length. We focus on the nontrivial case where $t(n), s(n) \geq n$. We remark that these constructions typically do not employ the vanilla architecture but instead introduce specific modifications; see Appendix A for a summary.

Source	Precision	Dimension	Window	CoT per TM-step
Pérez et al. [2021]	$O(\log t(n))$	$O(1)$	$n + t(n)$	1
Bhattamishra et al. [2020]	unbounded	$O(1)$	$n + t(n)$	1
Merrill and Sabharwal [2024]	$O(\log t(n))$	$O(1)$	$n + t(n)$	1
Li et al. [2024]*	$O(1)$	$O(\log t(n))$	$O(t(n) \log t(n))$	$O(\log t(n))$ (amortized)
Qiu et al. [2024]	$O(\log t(n))$	$O(1)$	$O(t(n) \log t(n))$	$O(\log t(n))$ (amortized)
This work	$O(1)$	$O(1)$	$s(n)$	$s(n)$

* Any multi-tape TM running in time $t(n)$ can be simulated by a Boolean circuit of size $O(t(n) \log t(n))$ [Arora and Barak, 2009].

AGI [Lecun, 2024, Goldblum et al., 2024], particularly for lifelong learning agents that interact with an open environment and accumulate an ever-growing history (e.g., the AIXI agent [Hutter, 2005]), where the input length grows unboundedly over time.

This motivates the following critical question:

Problem 1: Is it necessary to continue scaling up the bit size of transformers to handle longer inputs?

1.1 Our contribution

We answer Problem 1 in the negative. We prove that *constant bit-size transformers*² are Turing complete. Specifically, given any TM, there exists a transformer with fixed numerical precision and a fixed number of parameters that can simulate the TM on inputs of arbitrary length, provided that the transformer’s context window is sufficiently long. In particular, by applying it to the universal Turing machine, which takes as input a description TM of a TM and an input string x and outputs $\text{TM}(x)$, we can conclude that a single, constant bit-size transformer can compute any computable function, as long as the description of a relevant TM is loaded in the prompt.

Remark 1. *In fact, it is not even necessary to load the task-specific TM description into the prompt or to pre-inject it during pre-training. Instead, the transformer can be instantiated to simulate a TM that performs the meta-task of designing an algorithm for the given reasoning problem. For example, one can design the transformer to simulate Levin’s universal search algorithm [Levin, 1973, 1984, Schmidhuber, 2004], which solves arbitrary search problems, such as theorem proving and planning, as quickly as the fastest possible algorithm in an asymptotic sense. This suggests that, in principle, a single constant bit-size transformer has sufficient expressive power to achieve a form of general reasoning ability.*

We emphasize that the context window length is the minimal resource needed to scale up to handle longer inputs: The context window must be at least as long as the input length n , otherwise the transformer cannot even access the entire input. Moreover, we provide an exact characterization of the expressive power of constant bit-size transformers as a function of their context window length. Specifically, let $\text{WINDOW}[s(n)]$ denote the complexity class consisting of decision problems solvable by a constant bit-size transformer with a context window of length $O(s(n))$, $\text{SPACE}[s(n)]$ the class of decision problems solvable by a TM using $O(s(n))$ space, and $\text{PSPACE} := \bigcup_{k \in \mathbb{N}} \text{SPACE}[n^k]$ the class of decision problems solvable in polynomial space. We now state the main theorem of this paper:

Theorem 1. *For any non-decreasing function $s(n) \geq n$, we have $\text{WINDOW}[s(n)] = \text{SPACE}[s(n)]$. In particular, $\text{WINDOW}[\text{poly}(n)] = \text{PSPACE}$.*

Previous proofs establishing the Turing completeness of transformers suggested that the context window length must scale with the time complexity for solving the task. In contrast, Theorem 1 implies that it is sufficient for the context window length to scale only with the space complexity instead. This distinction is crucial, since many problems exhibit a significant gap between space

²Transformers are allowed to generate arbitrarily long CoTs before producing the final output.

complexity and time complexity. For instance, solving Boolean satisfication problems, Sokoban puzzles [Hearn and Demaine, 2005], model checking tasks [Sistla and Clarke, 1985], or other PSPACE-complete problems, require only polynomial space by tracking the current configuration, whereas the time complexity is widely conjectured to be superpolynomial [Arora and Barak, 2009] or even $2^{\text{poly}(n)}$ [Impagliazzo and Paturi, 2001]. Recently, Williams [2025] proved that every multitape Turing machine running in time $t(n)$ can be simulated in space only $O(\sqrt{t(n) \log t(n)})$. As a consequence, there are problems solvable in $O(s(n))$ space that require $\tilde{\Omega}(s(n)^2)$ time on multitape Turing machines.

In addition, the simulation of a TM by a transformer in Theorem 1 is uniform and easily generalizable with respect to the context window length. Specifically, to simulate a given TM on increasingly long inputs, the transformer just needs to extend its context window by adjusting the relative positional encoding according to an explicit formula, without changing all other model parameters.

Remark 2. *Aiming for stronger deductive reasoning ability, a line of engineering efforts tend to lengthen context window instead of growing parameter count. Our result formally justifies this strategy: extending the window alone is sufficient. Moreover, a window that grows only polynomially with the input already suffices to solve every PSPACE problem, including game-tree search and mathematical proofs.*

Remark 3. *There is an ongoing debate on whether transformers can approach AGI. A common negative view (e.g. [Lecun, 2024, Goldblum et al., 2024]) argues that transformers, limited by finite size and window, cannot approximate unbounded computation. Our result shows that, at least for deductive reasoning tasks (whose solutions depend only on explicit information), scaling window length alone is theoretically sufficient, and continually scaling up size is not a principled requirement.*

Our proof strategy is to simulate Post machines [Post, 1936] with transformers. A Post machine can be modeled as an automaton equipped with a queue. By storing the TM’s tape within the queue in a cyclic manner, Post machines can faithfully simulate Turing machines, and thus are Turing complete [Post, 1936, Davis et al., 1994]. By viewing the transformer’s context window as a queue-like structure, one can observe that the behavior of Post machines naturally aligns with that of transformers. Leveraging this behavioral similarity, we can then design a transformer that faithfully simulates the step-by-step execution of a Post machine. In general, this alignment offers a new interpretation of the attention mechanism: not only as a statistical aggregator, but also as a discrete computational operation over queue-like structures.

1.2 Other related work

Recent theoretical studies have examined the computational capabilities of transformers under various architectural constraints. Feng et al. [2023], Merrill and Sabharwal [2023, 2024], Li et al. [2024] proved that CoT steps can improve the reasoning capabilities of transformers: Transformers without CoT can only solve problems that can be solved by shallow circuits, whereas allowing polynomially long CoT steps enables the same model to solve any problem in P. In addition, Merrill and Sabharwal [2025] showed that a transformer with depth growing only logarithmically in the input length can recognize regular languages and solve graph connectivity problems without CoT, whereas a constant-depth transformer requires CoT steps of length $\Omega(n)$. We remark that these papers assume either a $O(\log n)$ -bit precision or a $O(\log n)$ -embedding size, and require the context window to be long enough to cover the entire context. The survey by Strobl et al. [2024] provides a comprehensive overview on formal-language expressivity of transformers, clarifying how design choices affect expressivity.

Aiming to reduce the required context window length and the memory, Yang et al. [2025] proposed the PENCIL framework, which incorporates a reduction mechanism to recursively “erase” intermediate CoT steps. This allows the context window length to scale with the space complexity of the computation, rather than the time complexity. We remark that their results still require increasing the bit-size of the transformer as the input length grows.

Schuurmans et al. [2024] considered a generalization of autoregressive decoding where, given a long input, emitted tokens are appended to the end of the sequence as the context window advances, and proved that the resulting system is Turing complete. This work also exploits the similarity between the behaviors of Post machines and Transformers. In fact, they analyze an even restrictive model, the so-called Lag system, where the next token depends solely on the leftmost **two** tokens in the queue

and **not** on the state q of PM. The main technical challenge in their construction is thus to compensate for the absence of access to the state q . As a result, their simulation requires $\Theta(s(n)^3)$ CoT tokens per TM step, whereas ours are only $O(s(n))$.

2 Notations and preliminaries

Let $\mathbb{R}$ be the set of real numbers and $\mathbb{N}$ be the set of natural numbers. For $n \in \mathbb{N}$, let $[n]$ denote $\{1, 2, \dots, n\}$. We use bold lowercase letters to represent vectors or sequences (e.g., $\mathbf{x}$), and bold uppercase letters to represent matrices (e.g., $\mathbf{A}$). Given a vector $\mathbf{x}$ and indices $j \leq i$, we use $\mathbf{x}_{j:i}$ to denote the sub-vector $(x_j, x_{j+1}, \dots, x_i)$. Let $\mathbf{0}_n$ denote the n -dimensional all-zeros vector. Given a vocabulary $\mathcal{V}$, let $\mathcal{V}^n$ denote the set of length- n strings over $\mathcal{V}$, and let $\mathcal{V}^* := \bigcup_{n=1}^{+\infty} \mathcal{V}^n$ denote the set of all finite strings over $\mathcal{V}$.

2.1 Turing machines

A (single-tape) Turing machine (TM) is defined as a tuple $\langle \Sigma, Q, \delta \rangle$ where

- Σ is the tape alphabet, including a designated “blank” symbol $\perp$, a designated “start” symbol $\triangleright$, and numbers 0 and 1.
- Q is a finite set consisting of possible states. We assume Q contains a designated start state q_{start} and a designated halting state q_{halt} .
- $\delta : Q \times \Sigma \rightarrow Q \times \Sigma \times \{\text{Left}, \text{Right}\}$ is a transition function describing the rules used in performing each step of the TM.

The machine has a single tape that is infinite to the right and bounded on the left. The tape is equipped with a tape head. Initially, the tape (from the leftmost cell rightward) contains the start symbol $\triangleright$, a finite $\{0, 1\}$ -string x (the input), and the blank symbols $\perp$ on the rest of its cells. The head starts at the leftmost tape cell, and the machine starts in the state q_{start} . The computation then repeats the following step until the machine enters q_{halt} : if the machine is in some state $q \in Q$ and is reading symbol σ from the tape, and if $\delta(q, \sigma) = (q', \sigma', z)$ for some $z \in \{\text{Left}, \text{Right}\}$, then the machine replaces σ with σ' in the current cell, changes state from q to q' , and moves the tape head one cell in direction z .

Space complexity Let $s : \mathbb{N} \rightarrow \mathbb{N}$ be a non-decreasing function. The complexity class $\text{SPACE}[s(n)]$ consists of all decision problems $\{0, 1\}^* \rightarrow \{0, 1\}$ that can be decided by a single-tape Turing machine using at most $O(s(n))$ tape cells on inputs of length n .³ The class $\text{PSPACE} := \bigcup_{k \in \mathbb{N}} \text{SPACE}[n^k]$ is defined as the class of decision problems that can be solved in polynomial space.

2.2 Post machines

A Post machine (PM) [Post, 1936] is defined as a tuple $\langle \Sigma, Q, \delta \rangle$ where

- Σ is the tape alphabet, including a designated “blank” symbol $\perp$, a designated “start” symbol $\triangleright$, and numbers 0 and 1.
- Q is a finite set consisting of possible states. We assume Q contains a designated start state q_{start} and a designated halting state q_{halt} .
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{\text{Stay}, \text{Right}\}$ is the transition function, specifying how the machine updates the tape and state.

The machine has a single tape that is infinite to the right and bounded on the left. The tape is equipped with two tape heads: the *front head* and the *rear head*. Initially, the tape (from leftmost cell rightward) contains the start symbol $\triangleright$, a finite $\{0, 1\}$ -string x (the input), and the blank symbols $\perp$ on the rest of its cells. The front head starts at the leftmost tape cell, the rear head starts at the right end of the input, and the machine starts in state q_{start} . The computation repeats the following step until the machine

³We implicitly assume that $s(n) \geq n$, since n tape cells are required to load an input of length n .

enters q_{halt} : if the machine is in state $q \in Q$ with the front head reading symbol σ from the tape, and if $\delta(q, \sigma) = (q', \sigma', z)$ for some $z \in \{\text{Stay}, \text{Right}\}$, then the machine (i) changes the state from q to q' , (ii) moves the front head either one cell to the right if $z = \text{Right}$, and (iii) moves the rear head one cell to the right, and then write σ' .

One can see that a PM can also be modeled as an automaton equipped with a queue of unbounded size: the queue is initialized as the input x , the front head points to the first element of the queue, and the rear head points to the last one. In each step, the PM reads and deletes the first element of the queue and may append a string to the tail.

Turing completeness of PM The model of PM is Turing complete, i.e., equivalent in computational power to TMs. Specifically,

Theorem 2. *Given any single-tape TM running in $t(n)$ time and $s(n)$ space, there is an equivalent PM running in $O(t(n) \times s(n))$ time and using a $s(n)$ -size queue.*

This theorem has been implicitly proved in several places, e.g., [Umans, 2025] and Section 9.1 of [Pettorossi, 2014]. For completeness, we present a proof in Appendix B. The intuition is as follows. The PM's queue stores the non-blank part of the TM's tape in a cyclic manner: If the TM head changes a symbol σ to σ' and moves right, the queue shifts by deleting σ from the front and appending σ' to the end. If the TM head moves left, the queue cyclically rotates by moving the last symbol to the front, where the appended symbol may need to depend on the leftmost **two** queue symbols. To avoid explicitly pre-reading the next queue cell, we apply a one-step *delayed append* trick: initially, pop the leftmost queue symbol and store it in the finite-state controller, without appending; thereafter, at each step, (i) pop the next leftmost symbol, and (ii) append a symbol determined by the two most recently popped symbols.

2.3 Transformers

For simplicity of notation, we present the definition of a single-head transformer, since only single-head attention is used in our proof. The definition of general multi-head transformers is deferred to Appendix C.

Let $\mathcal{V}$ be a finite vocabulary. A decoder-only transformer is a parameterized neural network TF_θ mapping from $\mathcal{V}^*$ to $\mathcal{V}$. Specifically, a transformer is a composition $\text{TF} := \text{out} \circ \text{dec}_{L-1} \circ \dots \circ \text{dec}_0 \circ \text{pos} \circ \text{emb}$ of four kinds of layers. Given a sequence $\mathbf{v} = (v_1, \dots, v_i) \in \mathcal{V}^i$, it computes the next token v_{i+1} as follows:

1. Token embedding layer (emb): For $j \leq i$, map each $v_j \in \mathcal{V}$ to a vector $\text{emb}(v_j) \in \mathbb{R}^d$. Here, d is called the *embedding size*.

2. Positional encoding layer (pos): For $j \leq i$, add a positional encoding $\text{pos}(i - j) \in \mathbb{R}^d$ to the token embedding $\text{emb}(v_j)$, resulting in the initial input representation $\mathbf{h}_j^0 := \text{emb}(v_j) + \text{pos}(i - j)$. Here, we assume a relative positional encoding scheme, where the encoding depends only on the distance between v_j and v_i .

3. Decoder Layer (dec _{ℓ} for $\ell = 0, \dots, L - 1$): Each decoder layer dec_ℓ consists of two sublayers: an self-attention layer, followed by a fully-connected feed-forward network. Residual connections are applied around each sub-layer⁴. Here, following common practice Pérez et al. [2021], Merrill and Sabharwal [2024, 2023], Qiu et al. [2024], Yang et al. [2025], we use hardmax as a realistic abstraction of softmax. In fact, hardmax is the instance-wise limit of zero temperature limit of softmax [Yang et al., 2025]. Consider a vector $\mathbf{s} \in \mathbb{R}^n$. If the maximum value in $\mathbf{s}$ appears at t positions, then the hardmax function is defined coordinate-wise as:

$$\text{hardmax}(\mathbf{s})_j := \begin{cases} \frac{1}{t}, & \text{if } s_j = \max_k s_k, \\ 0, & \text{otherwise.} \end{cases}$$

3.1. Single-head self-attention layer: Compute the attention score

$$\mathbf{s}_i^\ell = \text{hardmax} \left(\langle \mathbf{h}_1^\ell \cdot \mathbf{Q}^\ell, \mathbf{h}_i^\ell \cdot \mathbf{K}^\ell \rangle, \dots, \langle \mathbf{h}_i^\ell \cdot \mathbf{Q}^\ell, \mathbf{h}_i^\ell \cdot \mathbf{K}^\ell \rangle \right),$$

⁴For simplicity, we ignore layer normalizations from the standard transformer architecture [Radford et al., 2019] in our analysis. With a little more technical treatment as in [Li et al., 2024], our results can be extended to transformers with layer normalizations.

and then

$$\mathbf{a}_i^\ell = \sum_{j=1}^i s_{i,j}^\ell \cdot v^\ell(\mathbf{h}_i^\ell), \text{ where } v^\ell(h) := \mathbf{h} \cdot \mathbf{V}^\ell \text{ and } s_{i,j}^\ell \text{ is the } j\text{-th entry of } \mathbf{s}_i^\ell.$$

Here, $\mathbf{Q}^\ell, \mathbf{K}^\ell, \mathbf{V}^\ell \in \mathbb{R}^{d \times d}$ are parametrized matrices. Then, this sublayer returns

$$\mathbf{h}_i^{\ell+0.5} := \mathbf{W}^\ell \cdot \mathbf{a}_i^\ell + \mathbf{b}^\ell + \mathbf{h}_i^\ell,$$

where $\mathbf{W}^\ell \in \mathbb{R}^{d \times d}$ and $\mathbf{b}^\ell \in \mathbb{R}^d$ are parametrized.

3.2. Feed-forward network: Apply a multi-layer fully-connected ReLU neural network FF to $\mathbf{h}_i^{\ell+0.5}$, and returns

$$\mathbf{h}_i^{\ell+1} = \text{FF}(\mathbf{h}_i^{\ell+0.5}) + \mathbf{h}_i^{\ell+0.5}.$$

4. Output layer (out): The final output representations $\mathbf{h}_i^L$ are projected onto the vocabulary space using a linear transformation followed by a arg max function:

$$v_{i+1} := \text{out}(\mathbf{h}_i^L) = \arg \max(\mathbf{W}^{\text{out}} \cdot \mathbf{h}_i^L + \mathbf{b}^{\text{out}}), \text{ where } \mathbf{W}^{\text{out}} \in \mathbb{R}^{|\mathcal{V}| \times d} \text{ and } \mathbf{b}^{\text{out}} \in \mathbb{R}^{|\mathcal{V}|}.$$

We say that a transformer operates with a *context window* of length s if the generation of v_{i+1} depends only on the last s tokens $v_{i-s+1}, \dots, v_i$. A transformer is said to be of p -bit precision if each parameter is represented using p bits. We define the *bit-size* of a transformer TF_θ as $p \times |\theta|$, meaning that the entire model can be represented by using $p \times |\theta|$ bits. In this paper, when we refer to transformers, we mean transformers that are allowed to generate arbitrarily long immediate CoT steps, unless stated otherwise.

Definition 1. We define $\text{WINDOW}[s(n)]$ as complexity class consisting of the problems $\{0, 1\}^* \rightarrow \{0, 1\}$ that can be solved by a constant bit-size transformer with context window of length $O(s(n))$ on inputs of length n .

3 Proof of main results

In this section, we prove Theorem 1, which characterizes the expressive power of constant bit-size transformers with context window of length $s(n)$ in terms of the complexity class $\text{SPACE}[s(n)]$. In particular, it implies that constant bit-size transformers are Turing complete.

First, we prove that any constant bit-size transformer with a context window of length $s(n)$ can be simulated by a Turing machine using $O(s(n))$ space.

Theorem 3. For any non-decreasing function $s(n) \geq n$, $\text{WINDOW}[s(n)] \subseteq \text{SPACE}[s(n)]$.

Proof. Let $L : \{0, 1\}^* \rightarrow \{0, 1\}$ be a decision problem that can be solved by a constant bit-size transformer using a context window of length $s(n)$. We sketch a Turing machine using $O(s(n))$ space that solves L by simulating the transformer. The Turing machine maintains a buffer of size $s(n)$ storing the latest $s(n)$ tokens in the transformer's context. To compute the next token generated by the transformer, the Turing machine simulates the computation procedure described in Section 2.3, using an additional $O(s(n))$ working space. After generating the next token, the Turing machine updates the buffer by appending the new token and deleting the oldest token that slides out of the window, and then cleans the working space. By iteratively repeating this procedure, one can see that the Turing machine can faithfully simulate the transformer's behavior and solve the problem L . $\square$

In the following, we prove the reverse direction, which is the main technical part.

Theorem 4. Let TM be a single-tape Turing machine that, on input $x \in \{0, 1\}^n$, uses at most $s(n)$ space and runs for at most $t(n)$ steps. There exists a constant bit-size transformer with a context window of length $O(s(n))$ that, on input x , takes $O(t(n) \cdot s(n))$ CoT steps and then outputs $\text{TM}(x)$.

As a consequence, we have $\text{WINDOW}[(s(n))] \supseteq \text{SPACE}[s(n)]$ for $s(n) \geq n$.

Proof. Let TM be a single-tape Turing machine that runs in $t(n)$ steps and uses at most $s(n)$ tape cells. By Theorem 2, the TM can be simulated by a Post machine that runs in $O(t(n) \cdot s(n))$ time and

uses a queue of size $s(n)$. We slightly adapt the Post machine to ensure that the queue size remains exactly $s(n)$ throughout the computation, or equivalently, that the distance between the front and rear heads on the tape remains fixed at $s(n)$. Specifically, we pad the input x with $s(n) - n$ copies of a special symbol $\#$ to the right initially, and in each computation step, both the front and rear heads always move one cell to the right. One can easily see that such an adapted PM $= (\Sigma, Q, \delta)$ simulates the Turing machine as well. Furthermore, without loss of generality, we assume $\Sigma = \{0, 1, \#\}$, $Q = \{0, 1\}^c$ for some $c \in \mathbb{N}$, and that $q_{start} \notin \delta(\sigma, q)$ for any (σ, q) , meaning that once the computation starts, the Post machine leaves q_{start} immediately and never comes back.

We now construct a constant bit-size transformer TF with a context window of length $s(n)$ to faithfully simulate PM. The intuition is as follows:

- The $s(n)$ -size queue is simulated by the $s(n)$ -long context window, where the first element in the queue (or the tape cell pointed by the front head) corresponds to the oldest token in the window, and the last element in the queue (or the tape cell pointed by the rear head) corresponds to the newest token. Besides, we will let the vocabulary of the transformer be $\mathcal{V} = \Sigma \times Q$, so that a token also tracks the PM state information.
- As mentioned above, in each computation step of PM, the first element in the queue is removed, and a new element is added, so that the queue size remains the same. Correspondingly, in each CoT step of the transformer, the oldest token in the window slides out, and a new token is appended.
- The transition function of PM takes the last element in the queue and the current state as input, and outputs the next state and the next element. Correspondingly, by carefully choosing the parameters, the self-attention layer retrieves the last element from the oldest token in the window, and the current state from the current token; subsequently, a feed-forward network is used to implement the transition function δ .

We formally define the transformer TF as follows: let $\mathcal{V} = \Sigma \times Q = \{0, 1, \#\} \times \{0, 1\}^c$.

1. Token embedding layer Map each token $v = (\sigma, q)$ to a vector $\text{emb}(\sigma, q) \in \mathbb{R}^{c+8}$ as follows:

$$\text{emb}(\sigma, q) = \begin{cases} (1, 1, 0, 0, q, 0, 0, 0, 0), & \text{if } \sigma = 0; \\ (1, 0, 1, 0, q, 0, 0, 0, 0), & \text{if } \sigma = 1; \\ (1, 0, 0, 1, q, 0, 0, 0, 0), & \text{if } \sigma = \#. \end{cases} \quad (1)$$

2. Positional encoding layer Add a relative positional encoding $\text{pos}(i - j) \in \mathbb{R}^{c+8}$ to the token embedding $\text{emb}(v_j)$ where

$$\text{pos}(i - j) = \begin{cases} \mathbf{0}_{c+8}, & \text{if } 0 < i - j < s(n) - 1; \\ (\mathbf{0}_{c+7}, 1), & \text{if } i - j = 0, \\ (\mathbf{0}_{c+7}, 2), & \text{if } i - j = s(n) - 1, \end{cases} \quad (2)$$

Then return $\mathbf{h}_j^0 = \text{emb}(v_j) + \text{pos}(i - j)$.

3. Decoder layer The transformer has only one decoder layer, and the decoder layer has only one attention head. In the self-attention layer, we set the matrix $\mathbf{K}, \mathbf{Q}, \mathbf{V}, \mathbf{W} \in \mathbb{R}^{d \times d}$ and $\mathbf{b} \in \mathbb{R}^d$ where $d = c + 8$ as follows:

- Matrix $\mathbf{K}$: it has only one non-zero entry $K_{c+8,1} = 1$;
- Matrix $\mathbf{Q}$: it has only one non-zero entry $Q_{c+8,c+8} = 1$;
- Matrix $\mathbf{V}$: it has four non-zero entries: $V_{c+5,2} = V_{c+6,3} = V_{c+7,4} = V_{c+8,c+8} = 1$.

Besides, we set the matrix $\mathbf{W} \in \mathbb{R}^{d \times d}$ to be the identity matrix, and $\mathbf{b} = \mathbf{0}_{c+8}$.

The feed-forward network FF is designed to simulate transition function δ . Specifically, we let

$$\text{FF}(\mathbf{h}) + \mathbf{h} = \begin{cases} \mathbf{e}_{(\#, q_{start})}, & \text{if } h_{c+8} = 2; \\ \mathbf{e}_{\delta(0, q)}, & \text{if } \mathbf{h}_{4:c+8} = (q, 1, 0, 0, 3); \\ \mathbf{e}_{\delta(1, q)}, & \text{if } \mathbf{h}_{4:c+8} = (q, 0, 1, 0, 3); \\ \mathbf{e}_{\delta(\#, q)}, & \text{if } \mathbf{h}_{4:c+8} = (q, 0, 0, 1, 3); \end{cases} \quad (3)$$

Here, $\mathbf{e}_{(\sigma', q')} \in \mathbb{R}^{|\mathcal{V}|}$ is the unit vector where the entry indexed by (σ', q') is 1 and any other entry is 0.

4. Output layer We set W^{out} to be the identity matrix and $\mathbf{b}^{\text{out}}$ the all-zeros vector. Then the output layer outputs $v_{i+1} := \arg \max(\mathbf{h}_i^1)$.

In the following, we show that the transformer TF faithfully simulates the Post machine PM. Let $(\sigma_1, q_1), (\sigma_2, q_2), \dots$ denote the execution log of PM running on $x \in \{0, 1\}^n$, where σ_i is the i -th element added to the queue and q_i is the state when adding σ_i . Note that $(\sigma_i, q_i) = (x_i, q_{start})$ for $i \leq n$, $(\sigma_i, q_i) = (\#, q_{start})$ for $n+1 \leq i < s(n)$, and $\delta(\sigma_{i-s(n)+1}, q_i) = (\sigma_{i+1}, q_{i+1})$ for $i \geq s(n)$. Let $v_1, v_2, \dots$ denote the token sequence in the context of TF when it takes $v_1 = (x_1, q_{start}), \dots, v_n = (x_n, q_{start})$ as input. We show by induction that for each $i \geq 1$, we have $v_i = (\sigma_i, q_i)$.

Base case: $i \leq n$ This case is trivial.

Inductive case: $n+1 \leq i < s(n)$ In the self-attention layer, observing that $\mathbf{K} \cdot \mathbf{h}_i^0 = (\mathbf{0}_{c+7}, 1) = 1$ and

$$\mathbf{Q} \cdot \mathbf{h}_j^0 = \text{pos}(i-j) = \begin{cases} \mathbf{0}_{c+8}, & \text{if } 1 \leq j < i, \\ (\mathbf{0}_{c+7}, 1), & \text{if } j = i, \end{cases} \quad (4)$$

we have

$$\mathbf{s}_i = \text{hardmax}(\langle \mathbf{h}_1^0 \cdot \mathbf{Q}, \mathbf{h}_i^0 \cdot \mathbf{K} \rangle, \dots, \langle \mathbf{h}_i^0 \cdot \mathbf{Q}, \mathbf{h}_i^0 \cdot \mathbf{K} \rangle) = (0, \dots, 0, 1) \in \mathbb{R}^i$$

and

$$\mathbf{a}_i = \sum_{j=1}^i s_{i,j} \cdot (\mathbf{V} \cdot \mathbf{h}_j^0) = \sum_{j=1}^i s_{i,j} \cdot (\mathbf{0}_{c+4}, \mathbf{h}_{j,2:4}^0, \mathbf{h}_{j:c+8}^0) = (\mathbf{0}_{c+4}, \mathbf{h}_{i,2:4}^0, 1).$$

Then

$$\mathbf{h}_i^{0.5} = \mathbf{W} \cdot \mathbf{a}_i + \mathbf{b} + \mathbf{h}_i^0 = \mathbf{a}_i + \mathbf{h}_i^0 = (\mathbf{h}_{i,1:c+4}^0, \mathbf{h}_{i,2:4}^0, 2).$$

In the feed-forward network layer, $\mathbf{h}_i^{0.5}$ is mapped to $\text{FF}(\mathbf{h}_i^{0.5}) + \mathbf{h}_i^{0.5}$, which is $\mathbf{e}_{(\#, q_{start})}$ according to Equation (3). Finally, the output layer outputs $(\#, q_{start})$ as desired.

Inductive case: $i \geq s(n)$ In the self-attention layer, observing that $\mathbf{K} \cdot \mathbf{h}_i^0 = (\mathbf{0}_{c+7}, 1) = 1$ and

$$\mathbf{Q} \cdot \mathbf{h}_j^0 = \text{pos}(i-j) = \begin{cases} 0, & \text{if } i-s(n)+1 < j < i; \\ 1, & \text{if } j = i, \\ 2, & \text{if } j = i-s(n)+1, \end{cases} \quad (5)$$

we have

$$\mathbf{s}_i = \text{hardmax}(\langle \mathbf{h}_{i-s(n)+1}^0 \cdot \mathbf{Q}, \mathbf{h}_i^0 \cdot \mathbf{K} \rangle, \dots, \langle \mathbf{h}_i^0 \cdot \mathbf{Q}, \mathbf{h}_i^0 \cdot \mathbf{K} \rangle) = (1, 0, \dots, 0) \in \mathbb{R}^{s(n)}$$

and

$$\mathbf{a}_i = \sum_{j=i-s(n)+1}^i s_{i,j} \cdot (\mathbf{0}_{c+4}, \mathbf{h}_{j,2:4}^0, \mathbf{h}_{j:c+8}^0) = (\mathbf{0}_{c+4}, \mathbf{h}_{i-s(n)+1,2:4}^0, 2).$$

Then

$$\mathbf{h}_i^{0.5} = \mathbf{a}_i + \mathbf{h}_i^0 = (\mathbf{h}_{i,1:c+4}^0, \mathbf{h}_{i-s(n)+1,2:4}^0, 3).$$

In the feed-forward network layer, noting that by the induction hypothesis, we have

$$\mathbf{h}_{i,5:c+4}^{0.5} = \mathbf{h}_{i,5:c+4}^0 = q_i, \text{ and } \mathbf{h}_{i,c+5:c+7}^{0.5} = \mathbf{h}_{i-s(n)+1,2:4}^0 = \sigma_{i-s(n)+1}.$$

The vector $\mathbf{h}_i^{0.5}$ is mapped to $\text{FF}(\mathbf{h}_i^{0.5}) + \mathbf{h}_i^{0.5}$, which is $\mathbf{e}_{\delta(\sigma_{i-s(n)+1}, q_i)}$ according to Equation (3). Recall that $\delta(\sigma_{i-s(n)+1}, q_i) = (\sigma_{i+1}, q_{i+1})$, then one can see that the output layer outputs (σ_{i+1}, q_{i+1}) as desired.

Now, we have shown that $v_i = (\sigma_i, q_i)$ for any $i \geq 1$ and thus finished the proof. $\square$

Remark 4. As we can see from the proof, to simulate the given TM on longer inputs, what we need to do is just change the relative positional encoding according to Equation (2), without changing all other model parameters.

Remark 5. We briefly compare our construction with existing Turing-completeness proofs and highlight why our construction achieves constant bit-size. In [Pérez et al., 2021] and related follow-ups, the query may attend to many earlier tokens; when multiple positions tie for the maximum score, the hardmax outputs fractions such as $1/t$, which requires the precision to grow with t . The proof of Li et al. [2024] encodes the simulated circuit directly into the absolute positional encoding, rather than into the parameters. Consequently, as the circuit expands, the positional encoding must also grow in dimension, so the embedding size is no longer constant.

In contrast, our construction leverages the automaton-like behavior of Post machines, which is simpler and more regular than the head movements of Turing machines or the wiring patterns of circuits. As a result, at each CoT step the query attends to exactly one token, specifically the token located $s(n)$ positions earlier. This fixed-offset attention can be implemented purely through a relative positional encoding, without any additional arithmetic operations.

In particular, the input to hardmax is always one-hot, so is the resulted attention distribution. As an alternative of hardmax, one could replace hardmax with the right-most hardmax function [Yang et al., 2024].

Combining Theorems 3 and 4, we immediately conclude our main theorem.

Theorem 1. For any non-decreasing function $s(n) \geq n$, we have $\text{WINDOW}[s(n)] = \text{SPACE}[s(n)]$. In particular, $\text{WINDOW}[\text{poly}(n)] = \text{PSPACE}$.

4 Conclusions and Discussions

In this work, we have shown that a constant bit-size transformer, with a sufficiently long context window, can simulate any Turing machine on arbitrarily long inputs. We further proved that the complexity class $\text{SPACE}[s(n)]$ precisely characterizes the expressive power of constant bit-size transformers with a context window of length $s(n)$, suggesting that the context window length needs to scale only with the space complexity rather than the time complexity as suggested by previous results. Our proof leverages the behavioral similarity between transformers and Post machine, which offers new insights into the mechanism underlying the reasoning abilities of transformers. We list some future directions:

- *Simulation efficiency:* Our construction requires $O(t(n) \cdot s(n))$ CoT steps, which is potentially prohibitive for practical applications. It would be interesting and important to investigate whether this slowdown can be avoided without compromising optimality in other aspects.
- *Positional encodings:* Our construction employs a nonstandard relative positional encoding. It is open whether it can be replaced with fixed absolute positional encodings or other standard relative positional encodings. Moreover, our positional encoding explicitly depends on the assumed space upper bound, and it is unclear how this bound could be inferred automatically.
- *Learnability and empirical validation:* Our contribution is purely about expressiveness. Whether standard training procedures can learn such behavior is an important open problem. How well our construction behaves in practice also remains to be investigated.

5 Acknowledgments

The authors thank the anonymous NeurIPS reviewers for their valuable suggestions.

Qian Li’s work was supported by Shenzhen-Hong Kong Science and Technology Innovation Cooperation Zone Project (No.HZQSW-S-KCCYB-2024016). Yuyi Wang’s work was supported by the Hunan Provincial Natural Science Foundation (Grant No.2024JJ5128).

References

- J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- Anthropic. The claude 3 model family: Opus, sonnet, haiku, 2024.
- S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.
- S. Bhattamishra, A. Patel, and N. Goyal. On the computational power of transformers and its implications in sequence modeling. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 455–475, 2020.
- M. Davis, R. Sigal, and E. J. Weyuker. *Computability, complexity, and languages: fundamentals of theoretical computer science*. Elsevier, 1994.
- A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- G. Feng, B. Zhang, Y. Gu, H. Ye, D. He, and L. Wang. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36: 70757–70798, 2023.
- T. Gemini, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- M. Goldblum, M. A. Finzi, K. Rowan, and A. G. Wilson. Position: The no free lunch theorem, kolmogorov complexity, and the role of inductive biases in machine learning. 2024.
- D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- R. A. Hearn and E. D. Demaine. Pspace-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1-2):72–96, 2005.
- D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.
- M. Hutter. *Universal artificial intelligence: Sequential decisions based on algorithmic probability*. Springer Science & Business Media, 2005.
- R. Impagliazzo and R. Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.
- C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.
- Y. Lecun. Meta ai, open source, limits of llms, agi & the future of ai, 2024. <https://www.youtube.com/watch?v=5t1vTLU7s40>.
- L. A. Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973.
- L. A. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61(1):15–37, 1984.
- Z. Li, H. Liu, D. Zhou, and T. Ma. Chain of thought empowers transformers to solve inherently serial problems. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

- W. Merrill and A. Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545, 2023.
- W. Merrill and A. Sabharwal. The expressive power of transformers with chain of thought. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.
- W. Merrill and A. Sabharwal. A little depth goes a long way: The expressive power of log-depth transformers. *CoRR*, abs/2503.03961, 2025.
- J. Pérez, P. Barceló, and J. Marinkovic. Attention is turing-complete. *J. Mach. Learn. Res.*, 22: 75:1–75:35, 2021. URL <https://jmlr.org/papers/v22/20-302.html>.
- A. Pettorossi. *Elements of computability, decidability, and complexity*. Aracne editrice Srl, 2014.
- E. L. Post. Finite combinatory processes—formulation1. *The journal of symbolic logic*, 1(3):103–105, 1936.
- R. Qiu, Z. Xu, W. Bao, and H. Tong. Ask, and it shall be given: Turing completeness of prompting. *CoRR*, abs/2411.01992, 2024.
- A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.
- J. Schmidhuber. Optimal ordered problem solver. *Machine Learning*, 54:211–254, 2004.
- D. Schuurmans, H. Dai, and F. Zanini. Autoregressive large language models are computationally universal. *arXiv preprint arXiv:2410.03170*, 2024.
- A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM (JACM)*, 32(3):733–749, 1985.
- L. Strobl, W. Merrill, G. Weiss, D. Chiang, and D. Angluin. What formal languages can transformers express? a survey. *Transactions of the Association for Computational Linguistics*, 12:543–561, 2024.
- C. Umans. Decidability and tractability: Solution set 3, 2025. <https://users.cms.caltech.edu/~umans/cs21/soln3.pdf>.
- R. R. Williams. Simulating time with square-root space. *arXiv preprint arXiv:2502.17779*, 2025.
- A. Yang, D. Chiang, and D. Angluin. Masked hard-attention transformers recognize exactly the star-free languages. In A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.
- C. Yang, N. Srebro, D. McAllester, and Z. Li. Pencil: Long thoughts with short memory. *arXiv preprint arXiv:2503.14337*, 2025.
- N. Zubić, F. Soldá, A. Sulser, and D. Scaramuzza. Limits of deep learning: Sequence modeling through the lens of complexity theory. *arXiv preprint arXiv:2405.16674*, 2024.

A Modifications on Transformers in Turing Completeness Proofs

Existing proofs of Turing-completeness for Transformers typically departure from the vanilla architecture and make specific modifications. Pérez et al. [2021] consider a Transformer with an absolute positional encoding given by the triplet $(i, 1/i, 1/i^2)$ at position i . In their construction, the attention score is defined as the negative absolute value of the dot product, and the attention mechanism uses average-hard attention. Moreover, the feed-forward layers employ sigmoid activations in place of ReLUs. Merrill and Sabharwal [2024] dispense with positional encodings altogether and instead adopt *Strict Causal Masking*, where attention at position i can access all tokens up to position $i - 1$ but not the current token i . Their construction also uses average-hard attention and introduces a *Projected Pre-Norm*: an arbitrary linear projection is allowed before layer normalization. In the proof of Li et al. [2024], the simulated circuit is directly encoded into the absolute positional encoding, rather than into the parameters. Qiu et al. [2024] consider Transformers with nonstandard absolute positional encodings. In their construction, the query, key, and value maps in the attention sublayer are implemented as ReLU networks rather than linear transformations. This work employ nonstandard *relative* positional encodings.

B Proof of Theorem 2

Let $\text{TM} = (\Sigma, Q, \delta)$ be a single-tape Turing machine that runs in time $t(n)$ and space $s(n)$. We construct an equivalent Post machine $\text{PM} = (\Sigma', Q', \delta')$ that simulates TM within $O(t(n) \cdot s(n))$ time and uses a queue of size at most $s(n)$.

Queue alphabet We define the queue alphabet $\Sigma' := \{\sigma, \hat{\sigma}, \tilde{\sigma} \mid \sigma \in \Sigma\}$. Here, $\hat{\cdot}$ and $\tilde{\cdot}$ are temporary markers used for simulating a left move of TM's head. Specifically, $\hat{\cdot}$ temporarily marks the current head cell, while $\tilde{\cdot}$ temporarily marks its left neighbor.

Delayed append trick PM maintains a *logical queue* L which represents the non-blank segment of TM's tape in a cyclic manner. Formally, we realize L as $L = R \circ Q_{\text{phys}}$ where

- R is a register in the finite control storing the leftmost symbol of L , and
- Q_{phys} is the actual queue content. Thus $|Q_{\text{phys}}| = |L| - 1$.

With this representation, the appended symbol is no longer restricted to depend only on the leftmost symbol of L ; it can now be determined by the two leftmost symbols.

Initialization Initially, TM's tape contains the start symbol $\triangleright$, followed by the input x , and then blanks. Accordingly, the logical queue L is initialized to $(\triangleright, x)$: the first symbol $\triangleright$ is placed in the register R , and the remaining symbols x are stored into Q_{phys} . During the simulation, we will keep the following invariant:

- Right before each simulated step, the leftmost cell of L corresponds to TM's head cell; its symbol is either unmarked or carries the $\tilde{\cdot}$ mark, while all other cells of L are unmarked.

Simulation of one step Suppose TM is in state q and its head reads σ from cell i . Let $\delta(q, \sigma) = (q', \sigma', z)$ with $z \in \{\text{Left}, \text{Right}\}$. This means that TM overwrites cell i with σ' , change state from q to q' , and moves its head one cell in direction z .

- *Move Right.* PM deletes the leftmost logical symbol and appends σ' to L , thereby overwriting tape cell i . If the resulting leftmost logical symbol is $\triangleright$, indicating that TM has extended its non-blank segment by one blank cell to the right, then PM additionally appends a blank symbol $\perp$.
- *Move Left.* PM deletes the leftmost logical symbol and appends $\hat{\sigma}'$ to L , thereby overwriting tape cell i . It then cyclically rotates L until the second leftmost logical symbol carries the $\hat{\cdot}$ mark. At this point, the first and second of L correspond to tape cells $i - 1$ and i , respectively. PM then adds a mark $\tilde{\cdot}$ to the first symbol and removes the $\hat{\cdot}$ mark from the second, after one additionally cyclic sweep of L ; concretely,
 1. delete the leftmost element and append the same symbol with the mark $\tilde{\cdot}$; then

2. delete the leftmost element and append the same symbol with no mark; then
3. cyclically rotate L until the leftmost logical symbol carries the $\sim$ mark.

Analysis By induction on the number of simulated steps, one can check that the invariant is maintained throughout the simulation, and hence PM faithfully simulate TM. For a right move, PM performs $O(1)$ queue operations. For a left move, it performs two full sweeps of L , costing $O(|L|) = O(s(n))$ operations. Therefore, the overall running time is $O(t(n) \cdot s(n))$, completing the proof.

C Multi-head Transformers

A multi-head Transformer employs the following *multi-head self-attention layer* in place of the single-head self-attention layer, while all other components remain the same as in the single-head transformer.

Multi-head self-attention layer: For each head $k = 1, 2, \dots, H$, compute the attention score

$$s_{k,i}^\ell = \text{hardmax} \left(\langle \mathbf{h}_1^\ell \cdot \mathbf{Q}_k^\ell, \mathbf{h}_i^\ell \cdot \mathbf{K}_k^\ell \rangle, \dots, \langle \mathbf{h}_i^\ell \cdot \mathbf{Q}_k^\ell, \mathbf{h}_i^\ell \cdot \mathbf{K}_k^\ell \rangle \right),$$

and then

$$\mathbf{a}_{i,k}^\ell = \sum_{j=1}^i s_{k,i,j}^\ell \cdot v_k^\ell(\mathbf{h}_i^\ell), \text{ where } v_k^\ell(h) := \mathbf{h} \cdot \mathbf{V}_k^\ell$$

Here, $\mathbf{Q}_k^\ell, \mathbf{K}_k^\ell, \mathbf{V}_k^\ell \in \mathbb{R}^{d \times d/H}$ are parametrized matrices. By concatenating the H heads, we obtain $\mathbf{a}_i^\ell = ((\mathbf{a}_{i,1}^\ell)^T, \dots, (\mathbf{a}_{i,H}^\ell)^T)^T \in \mathbb{R}^d$. Then, this sublayer returns

$$\mathbf{h}_i^{\ell+0.5} := \mathbf{W}^\ell \cdot \mathbf{a}_i^\ell + \mathbf{b}^\ell + \mathbf{h}_i^\ell,$$

where $\mathbf{W}^\ell \in \mathbb{R}^{d \times d}$ and $\mathbf{b}^\ell \in \mathbb{R}^d$ are parametrized.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [NA].
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes]" is generally preferable to "[No]", it is perfectly acceptable to answer "[No]" provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No]" or "[NA]" is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading “NeurIPS Paper Checklist”,**
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers.**

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope?

Answer: [Yes].

Justification: In Section 3, we prove the theoretical claim.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes].

Justification: Section 4 discusses the limitations.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[Yes\]](#) .

Justification: We provide the setting and preliminaries in Section 2, and the complete proof in Section 3.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[NA\]](#) .

Justification: The paper does not include experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.

- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [NA] .

Justification: This paper does not include experiments requiring code

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [NA] .

Justification: This paper does not include experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [NA] .

Justification: This paper does not include experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [NA] .

Justification: This paper does not include experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes] .

Justification: The research conducted in this paper conforms with the NeurIPS Code of Ethics in every respect.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA] .

Justification: This paper is a foundational research and not tied to particular applications.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA] .

Justification: This paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [NA] .

Justification: This paper does not use existing assets

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA] .

Justification: This paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA] .

Justification: This paper does not involve crowdsourcing experiments nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA] .

Justification: This paper does not involve crowdsourcing experiments nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA] .

Justification: In this paper, the LLM is used only for writing and editing, and the core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.