On Expressive Power of Looped Transformers: Theoretical Analysis and Enhancement via Timestep Encoding

Kevin Xu [1] **Issei Sato** [1]

Abstract

Looped Transformers provide advantages in parameter efficiency, computational capabilities, and generalization for reasoning tasks. However, their expressive power regarding function approximation remains underexplored. In this paper, we establish the approximation rate of Looped Transformers by defining the modulus of continuity for sequence-to-sequence functions. This reveals a limitation specific to the looped architecture. That is, the analysis prompts the incorporation of scaling parameters for each loop, conditioned on timestep encodings. Experiments validate the theoretical results, showing that increasing the number of loops enhances performance, with further gains achieved through the timestep encoding. Code is available at `https://github.com/kevin671/tmlt`.

1. Introduction

Transformers (Vaswani et al., 2017) have become the standard architecture for a wide range of machine learning tasks, including natural language processing and computer vision. However, they exhibit certain limitations, particularly when applied to complex tasks. The expressive power of Transformers is theoretically constrained (Merrill & Sabharwal, 2023; Feng et al., 2023), and they empirically struggle with reasoning and planning problems (Kambhampati et al., 2024). Although chain-of-thought reasoning (Wei et al., 2022) can mitigate these challenges in some cases, it typically relies on manually crafted prompts or costly intermediate supervision. Moreover, Transformers encounter difficulties with length generalization (Deletang et al., 2023) and require substantial computational resources as the number of model parameters increases (Pope et al., 2022).

To address these limitations, Looped Transformers presents a promising approach. The architecture consists of fixed-size Transformer layers, in which the output is recursively fed back into the input. Looped Transformers exhibit advantages in parameter efficiency thanks to their weight-tying structure (Lan et al., 2020; Takase & Kiyono, 2021; Csordás et al., 2024; Bae et al., 2025), achieving performance comparable to standard Transformers while using fewer parameters. Additionally, they are well suited for size generalization by adjusting the loop count based on task complexity (Dehghani et al., 2019; Fan et al., 2024b). Their recursive structure endows them with the expressive power to emulate iterative algorithms and universal computational capabilities, akin to programmable computers (Giannou et al., 2023). Furthermore, their inductive bias enhances performance on reasoning tasks (Saunshi et al., 2025).

In contrast, the expressive power of Looped Transformers and the properties unique to the looped architecture in function approximation remain unexplored. The expressive power of standard Transformers, on the other hand, has been examined extensively in prior studies. These studies show that Transformers can be universal approximators for continuous permutation-equivariant functions on compact domains (Yun et al., 2020; Kajitsuka & Sato, 2024). Furthermore, their approximation rate has been analyzed by identifying specific properties of the target functions (Takakura & Suzuki, 2023; Jiang & Li, 2024; Wang & E, 2024), providing insights into the underlying characteristics of Transformer architectures. However, these findings cannot be directly extended due to the weight-tying constraints. Although the approximation rate of looped ReLU networks has been established only recently (Zhang et al., 2023), that of Looped Transformers remains unknown.

Our contributions are summarized as follows:

- We establish the approximation rate of Looped Transformers for fixed-length continuous sequence-to-sequence functions, with respect to the number of loops and three newly defined types of moduli of continuity.
- We identify an inherent limitation of Looped Transformers and address it by proposing Timestep-Modulated Looped Transformers (TMLT), which incorporate scaling parameters that are conditioned on timestep encodings.

[1]The University of Tokyo. Correspondence to: Kevin Xu <kevinxu@g.ecc.u-tokyo.ac.jp>, Issei Sato <sato@g.ecc.u-tokyo.ac.jp>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

2. Background

This section defines the Transformer and Looped Transformer architectures, reviews related work, and examines prior theoretical studies on the function approximation capabilities of Transformers and weight-tied networks, thereby clarifying the research question addressed in this paper.

Notations: Vectors are represented by lowercase boldface letters *e.g.*, $\boldsymbol{v}$, and matrices are denoted by uppercase boldface letters *e.g.*, $\boldsymbol{X}$. The i-th element of a vector $\boldsymbol{v}$ is denoted by $\boldsymbol{v}_i$, and the (i, j)-th element of a matrix $\boldsymbol{X}$ is denoted by $\boldsymbol{X}_{i,j}$. The n-th column of a matrix $\boldsymbol{X}$ is denoted by $\boldsymbol{X}_{:,n}$.

Given an input sequence $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{m\times N}$, where $\boldsymbol{x}_i \in \mathbb{R}^m$, and a function $f : \mathbb{R}^m \to \mathbb{R}^m$, the token-wise application of f is denoted by the bold symbol $\boldsymbol{f}$ *i.e.*

$$\boldsymbol{f}(\boldsymbol{X}) = [f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_N)] \in \mathbb{R}^{m\times N}.$$

For $p \in [1, \infty)$, the p-norm, denoted by $\|\cdot\|_p$, represents the entry-wise p-norm. This norm applies to both vectors and matrices *e.g.*, $\|\boldsymbol{X}\|_p$. The L^p-norm of a function is defined for $p \in [1, \infty)$ as:

$$\|f\|_{L^p} := \left(\int_{\Omega} \|f(\boldsymbol{X})\|_p^p \, d\boldsymbol{X} \right)^{1/p},$$

where Ω represents the domain of the function f.

2.1. Transformer Architecture

Given an input sequence $\boldsymbol{X} \in \mathbb{R}^{m\times N}$, composed of N token embedding of dimension size m, the self-attention layers with h heads and head size s, and the feed-forward layer with width size q, are defined as follows:

$$\mathrm{Attn}(\boldsymbol{X}) = \sum_{i=1}^{h} \boldsymbol{W}_O^i \boldsymbol{W}_V^i \boldsymbol{X} \sigma_s \left[(\boldsymbol{W}_K^i \boldsymbol{X})^\top \boldsymbol{W}_Q^i \boldsymbol{X} \right],$$
$$\mathrm{FF}(\boldsymbol{X}_{:,n}) = \boldsymbol{W}_2 \sigma_R(\boldsymbol{W}_1 \boldsymbol{X}_{:,n} + \boldsymbol{b}_1) + \boldsymbol{b}_2,$$

where $\boldsymbol{W}_O^i \in \mathbb{R}^{m\times s}, \boldsymbol{W}_V^i, \boldsymbol{W}_K^i, \boldsymbol{W}_Q^i \in \mathbb{R}^{s\times m}, \boldsymbol{W}_1 \in \mathbb{R}^{q\times m}, \boldsymbol{W}_2 \in \mathbb{R}^{m\times q}, \boldsymbol{b}_1 \in \mathbb{R}^q, \boldsymbol{b}_2 \in \mathbb{R}^m$ are parameters, σ_R denotes ReLU function, and σ_s denotes a softmax operator applied to the columns of the matrix.

Transformer block $\mathrm{TF} : \mathbb{R}^{m\times N} \to \mathbb{R}^{m\times N}$ is defined by

$$\boldsymbol{X}' = \boldsymbol{X} + \mathrm{Attn}(\boldsymbol{X}),$$
$$\mathrm{TF}(\boldsymbol{X}) = \boldsymbol{X}' + \mathbf{FF}(\boldsymbol{X}'),$$

where $\mathbf{FF}$ represents token-wise FF. In other words,

$$\mathrm{TF} = (\mathrm{id} + \mathbf{FF}) \circ (\mathrm{id} + \mathrm{Attn}),$$

where id denotes the identity mapping, where we omit the domain of definition for simplicity. For the analysis of expressive power in Section 3, we exclude layer normalization and our constructive proof relies on the softmax function to approximate the hardmax function as in previous studies (Yun et al., 2020; Kim et al., 2023)

2.2. Looped Transformer

Looped Transformer with a single layer is represented as:

$$\boldsymbol{\mathcal{L}}_2 \circ \mathrm{TF}^{\circ r} \circ \boldsymbol{\mathcal{L}}_1,$$

where $\boldsymbol{\mathcal{L}}_2$ and $\boldsymbol{\mathcal{L}}_1$ represent token-wise affine linear layers, and $\mathrm{TF}^{\circ r}$ denotes the composition of TF applied r times. While we focus on single-layer as (Dehghani et al., 2019; Lan et al., 2020; Yang et al., 2024; Fan et al., 2024b), they can also be implemented with multiple layers as (Csordás et al., 2024; Bae et al., 2025; Saunshi et al., 2025).

Overview of Previous Work The recursive structure was introduced into Transformers (Dehghani et al., 2019), where the number of loops can be adaptively adjusted, allowing for size generalization (Fan et al., 2024a). Looped Transformers are closely related to weight-tying Transformers (Lan et al., 2020; Takase & Kiyono, 2021), achieving performance comparable to standard Transformers using fewer parameters. Deep equilibrium models (Bai et al., 2019), which compute fixed points of iterative layers, are also related. In addition, the recursive structure enables the model to emulate iterative algorithms, including basic computational primitives (Giannou et al., 2023) and learning algorithms (Giannou et al., 2024; Yang et al., 2024). Furthermore, recent studies have demonstrated that Looped Transformers exhibit an inductive bias towards reasoning tasks (Saunshi et al., 2025). To improve performance, more sophisticated architectures, such as mixture-of-experts (Csordás et al., 2024) and relaxed weight-tying (Bae et al., 2025), have been introduced.

2.3. Theoretical Analysis on Expressive Power

We review related work and summarize the comparisons between our problem setting and previous studies in Table 1.

Universality of Transformers The universal approximation theorem for fully connected neural networks (Cybenko, 1989; Hornik et al., 1989) shows that networks of sufficient size can approximate certain classes of functions with arbitrarily low error. For Transformers, the target function class extends to sequence-to-sequence functions. Transformers compute a *contextual mapping* of the input, which requires capturing the entire sequence and computing the token embedding within context (Yun et al., 2020), formulated as:

Definition 2.1 (Yun et al., 2020)**.** Consider a finite set $\mathbb{L} \subset \mathbb{R}^{d\times N}$. A map $\mathrm{CM} : \mathbb{L} \to \mathbb{R}^{1\times N}$ defines a *contextual mapping* if the map satisfies the following:

1. For any $\boldsymbol{L} \in \mathbb{L}$, the N entries in $\mathrm{CM}(\boldsymbol{L})$ are all distinct.
2. For any $\boldsymbol{L}, \boldsymbol{L}' \in \mathbb{L}$, with $\boldsymbol{L} \neq \boldsymbol{L}'$, all entries of $\mathrm{CM}(\boldsymbol{L})$ and $\mathrm{CM}(\boldsymbol{L}')$ are distinct.

Prior studies have shown that Transformers can compute contextual mappings, enabling memorization (Kim et al.,

Table 1. Comparisons of our problem setting with related work on the theoretical analysis of function approximation.

Paper	Model Type	Function Class	Approximation Rate	Looped (Weight-Tying)
Yarotsky (2018)	FFN	Continuous functions	✓	×
Yun et al. (2020)	Transformer	Continuous seq-to-seq functions	×	×
Takakura & Suzuki (2023)	Transformer	γ-smooth infinite-length	✓	×
Kajitsuka & Sato (2024)	Transformer	Continuous seq-to-seq functions	×	×
Jiang & Li (2024)	Transformer	Temporal coupled functions	✓	×
Wang & E (2024)	Transformer	Long but sparse memories	✓	×
Zhang et al. (2023)	FFN	Continuous functions	✓	✓
Ours	Transformer	Continuous seq-to-seq functions	✓	✓

2023) and universal approximation (Yun et al., 2020; Kajitsuka & Sato, 2024).

For Looped Transformers, as the fixed parameters of a single Transformer layer are used, the results of previous studies cannot be directly applied. This leads to the question: *Can Looped Transformers compute contextual mappings?* and *Are they universal approximators?*

Approximation Rate of Transformers Beyond the universality, the approximation rate provides deeper insights into the characteristics of models (Barron, 1993; Yarotsky, 2018). This rate is derived as an upper bound of error in terms of the properties of the target functions and the complexity of the networks. For Transformers, recent studies have investigated these rates and the nature of the target functions (Takakura & Suzuki, 2023; Jiang & Li, 2024; Wang & E, 2024). Specifically, they have shown conditions under which Transformers can overcome the curse of dimensionality (Takakura & Suzuki, 2023) and revealed structures in target functions that Transformers can effectively approximate (Jiang & Li, 2024; Wang & E, 2024).

Our study focuses on understanding the architectural properties of Looped Transformers, particularly in comparison to standard Transformers. To this end, we explore the approximation rate and investigate the properties of target functions that determine their approximation errors.

Expressive Power of Weight-Tied Neural Networks Recently, it has been shown that single *fixed-size* networks can serve as universal approximators in a parameter-efficient manner; that is, the parameter count depends solely on the input dimension, not the approximation error (Zhang et al., 2023). Furthermore, the approximation rate of weight-tied ReLU networks has been established with respect to the number of loops and the modulus of continuity of continuous functions (Zhang et al., 2023). The modulus of continuity for $g : \mathbb{R}^d \to \mathbb{R}$ and $\delta \geq 0$ is defined as:

$$\omega_g(\delta) := \sup \left\{ |g(\boldsymbol{x}) - g(\boldsymbol{x}')| : \|\boldsymbol{x} - \boldsymbol{x}'\|_2 \leq \delta \right\}.$$

Our question is whether the results can be extended to sequence-to-sequence functions and Transformers, which require contextual mappings. For a sequence-to-sequence function $f : \mathbb{R}^{d\times N} \to \mathbb{R}^{d\times N}$, the modulus of continuity can be generalized as:

$$\omega_f(\delta) := \sup \left\{ \|f(\boldsymbol{X}) - f(\boldsymbol{X}')\|_p : \|\boldsymbol{X} - \boldsymbol{X}'\|_2 \leq \delta \right\}$$

We investigate whether this modulus of continuity alone can determine the approximation rate.

For Looped Transformers, it has been shown that they can represent standard Transformers, although their parameter count grows with both the desired approximation accuracy and the sequence length (Saunshi et al., 2025). Moreover, no existing work has established their approximation rate.

3. Approximation Rate Analysis

In this section, we establish the approximation rate of Looped Transformers. We define three types for the modulus of continuity in Section 3.2 that determine the approximation rate. The main results are presented in Section 3.3, followed by a proof sketch in Section 3.4.

3.1. Preliminaries

The target function class of our analysis is continuous functions that Transformers can represent. Specifically, these are *permutation-equivariant* functions, defined as follows:

Definition 3.1 (Yun et al., 2020)**.** A function $f : \mathbb{R}^{d\times N} \to \mathbb{R}^{d\times N}$ is said to be *permutation equivariant* if $f(\boldsymbol{X}\boldsymbol{P}) = f(\boldsymbol{X})\boldsymbol{P}$ holds for any permutation matrix $\boldsymbol{P}$. Let $\mathcal{F}_{\mathrm{PE}}(\Omega)$ denote the set of continuous functions, defined on Ω, that are permutation equivariant.

We evaluate both the number of parameters and the *bit complexity*, the maximum number of bits required to represent the network's weights (Vardi et al., 2022; Kim et al., 2023).

In our proofs, we introduce *IDs* for tokens, sequences, and tokens within sequences as theoretical constructs to formalize contextual mappings.

Definition 3.2. A *token ID* is a unique integer assigned to each token. A *sequence ID* uniquely identifies each sentence.

A *contextual token ID* uniquely identifies a specific token within a specific sentence. We denote the set of contextual token IDs as $\mathcal{K} = 0, 1, \ldots, K-1$, with corresponding embeddings $\boldsymbol{y}_k \in \mathbb{R}^d$ for each $k \in \mathcal{K}$.

This notion is defined in Kim et al. (2023), to which we refer for further details, for constructive proofs of contextual mappings. The actual construction of contextual token IDs may vary depending on the specific proof. In our case, we adopt a different construction from that of Kim et al. (2023).

3.2. Definition of modulus of Continuity

As briefly mentioned in the preliminary discussion, we define the modulus of continuity in Eq. 2.3 as:

Definition 3.3 (Modulus of Sequence Continuity)**.** Given a sequence-to-sequence continuous function $f : \mathbb{R}^{d\times N} \to \mathbb{R}^{d\times N}$, the modulus of *sequence continuity* is defined by:

$$\omega_f(\delta) := \sup \left\{ \|f(\boldsymbol{X}) - f(\boldsymbol{X}')\|_p : \|\boldsymbol{X} - \boldsymbol{X}'\|_2 \le \delta \right\}.$$

We omit the subscript p for simplicity. This quantifies how the output sequence shifts relative to differences in input, hence referred to as *sequence continuity*.

We found that this alone is insufficient to determine the approximation rate of Looped Transformers, in contrast to the case of ReLU networks (Zhang et al., 2023). Informally, this issue arises because Transformers compute contextual mappings. We notably identified two additional types of modulus of continuity, defined as follows.

Definition 3.4 (Modulus of Contextual Continuity)**.** Given a sequence-to-sequence continuous function $f : \mathbb{R}^{d\times N} \to \mathbb{R}^{d\times N}$, the modulus of *contextual continuity* is defined by:

$$\omega_f^{\text{cont}}(\delta) := \sup_{n, \boldsymbol{X}, \boldsymbol{X}'} \left\{ \|f(\boldsymbol{X})_{:,n} - f(\boldsymbol{X}')_{:,n}\|_p \right. \\ \left. : \|\boldsymbol{X} - \boldsymbol{X}'\|_2 \le \delta,\ \boldsymbol{X}_{:,n} = \boldsymbol{X}'_{:,n} \right\}.$$

Definition 3.5 (Modulus of Token Continuity)**.** Given a sequence-to-sequence continuous function $f : \mathbb{R}^{d\times N} \to \mathbb{R}^{d\times N}$, the modulus of *token continuity* is defined by:

$$\omega_f^{\text{tok}}(\delta) := \sup_{n, \boldsymbol{X}, \boldsymbol{X}'} \left\{ \|f(\boldsymbol{X})_{:,n} - f(\boldsymbol{X}')_{:,n}\|_p : \right. \\ \left. \|\boldsymbol{X}_{:,n} - \boldsymbol{X}'_{:,n}\|_2 \le \delta,\ \boldsymbol{X}_{:,q} = \boldsymbol{X}'_{:,q}\ (\forall q \neq n) \right\}.$$

The *modulus of contextual continuity* quantifies the variation in the contextual token embeddings induced by perturbations of context. For example, consider the sentences: (1) "I write papers" and (2) "You write books". It measures the difference in the contextual token embedding of the same word 'write' within different contexts.

On the other hand, the *modulus of token continuity* quantifies the variation in the output embedding caused by perturbations to the token itself within the same context such as (1) "I write papers" and (2) "I draft papers".

3.3. Main Result

The result establishes the approximation rate of Looped Transformers in terms of the number of loops and the three types of moduli of continuity of the target function.

Theorem 3.6. *Given a function $f \in \mathcal{F}_{\text{PE}}([0,1]^{d\times N})$, $r > N$, there exists a Looped Transformer, composed of* $\text{TF} : \mathbb{R}^{(17d+9)\times N} \to \mathbb{R}^{(17d+9)\times N}$ *with two heads, head size* 1*, and width size of* $q = 49d + 25$*, and two affine linear maps* $\mathcal{L}_1 : \mathbb{R}^d \to \mathbb{R}^{17d+9}$ *and* $\mathcal{L}_2 : \mathbb{R}^{17d+9} \to \mathbb{R}^d$ *s.t.*

$$\begin{aligned} &\left\| \boldsymbol{\mathcal{L}}_2 \circ \text{TF}^{\circ r} \circ \boldsymbol{\mathcal{L}}_1 - f \right\|_{L^p} \\ &\le (Nd)^{\frac{1}{p}} \left(\omega_f^{\text{tok}}(\delta\sqrt{d}) + \omega_f^{\text{cont}}(\delta\sqrt{Nd}) \right) + \omega_f(\delta\sqrt{Nd}) \\ &\qquad + \mathcal{O}(N^{2/p}\delta^{d/p}) + \mathcal{O}\left(\left((M\delta)^{-1} dN \right)^{1/p} \right), \\ &\qquad \text{where } \delta = \left((r-N)/2 \right)^{-1/((N+1)d+1)}, \end{aligned}$$

where M is the maximum absolute value of the model parameters, and the bit complexity is $\mathcal{O}(\delta^{-(N+1)d})$.

Theorem 3.6 shows that increasing the number of loops r reduces the approximation error. Under infinite-precision weights, this leads to a universal approximation theorem.

Corollary 3.7 (Universality)**.** *The hypothesis space of Looped Transformers, defined by*

$$\begin{aligned} \mathcal{H} := \Big\{ &\boldsymbol{\mathcal{L}_2} \circ \text{TF}^{\circ r} \circ \boldsymbol{\mathcal{L}_1} : [0,1]^{d\times N} \to [0,1]^{d\times N} \mid \\ &m, q \le Cd,\ h = 2,\ s = 1,\ r \in \mathbb{N},\ \boldsymbol{W} \in \mathbb{R}^{n_w} \Big\}, \end{aligned}$$

where C is a positive constant, $\boldsymbol{W}$ denotes the flattened set of all weights in the network, and n_w represents the total number of these weights, is dense in $\mathcal{F}_{\text{PE}}([0,1]^{d\times N})$ w.r.t.the L^p norm.

This approximation analysis highlights the characteristics of Looped Transformers, including both their capabilities and limitations, as summarized below:

- While the number of parameters remains fixed at $O(d)$, independent of the desired approximation accuracy and the sequence length, the error can be reduced by increasing the number of loops.
- Looped Transformers, even with weight-tied self-attention using a hard-max function, can compute contextual mappings and become universal approximators.
- The approximation rate depends on three types of continuity, with contextual and token dependencies unique to Looped Transformers; these dependencies are not present in standard Transformers or looped ReLU networks.

Our contribution lies in establishing the approximation rate with respect to the number of loops, based on novel moduli of continuity that are unique to Looped Transformers.

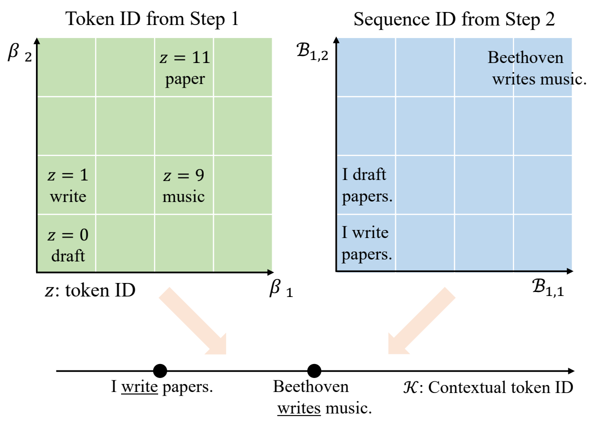


Figure 1. The networks construct contextual token IDs by combining token IDs with sequence IDs.

Furthermore, the additional dependency can amplify the approximation error, revealing a limitation inherent to Looped Transformers. A detailed discussion of this issue, along with improvement methods, is provided in Section 4.

3.4. Proof Sketch

This section presents a proof sketch, emphasizing distinctions from prior studies and challenges unique to the looped architecture. The formal proof is provided in Appendix A.

The basic strategy involves approximating the continuous target function f with a piecewise constant function $\bar{f}$, which is approximated by the network, denoted by $\tilde{f}$. For $\delta^{-1} \in \mathbb{N}$, $\delta^{-1} \geq 2$, the input space $[0,1]^{d\times N}$ is divided into discretized cubes with width δ, denoted by $\{Q_{\boldsymbol{\mathcal{B}}}\}_{\boldsymbol{\mathcal{B}}\in\{0,1,\ldots,\delta^{-1}-1\}^{d\times N}}$. Each cube is assigned a representative point $\hat{\boldsymbol{X}}_{\boldsymbol{\mathcal{B}}} \in Q_{\boldsymbol{\mathcal{B}}}$, and the piecewise constant function $\bar{f}$ is then defined as:

$$\bar{f}(\boldsymbol{X}) = f(\hat{\boldsymbol{X}}_{\boldsymbol{\mathcal{B}}}) \quad \text{where } \boldsymbol{\mathcal{B}} \text{ satisfies } \boldsymbol{X} \in Q_{\boldsymbol{\mathcal{B}}}. \tag{1}$$

The approximation with networks consists of three steps. First, the network assigns a *token ID* to each token. Second, it assigns a *sequence ID*. The combination of the *token ID* and *sequence ID* constitutes the *contextual token IDs* as in Fig. 1. Finally, these are mapped to embeddings that represent the output of the target function at each token.

Step 1. Token-wise Quantization. The network uses the feed-forward network to assign each input token, denoted by $\boldsymbol{X}_{:,n}$, to a token ID, denoted by z, in a token-wise manner.

$$\boldsymbol{X}_{:,n} \in [0,1]^d \to z \in \{0,1,\ldots,\delta^{-d}-1\}. \tag{2}$$

Step 2. Contextual Mapping. The network, given N token IDs computes their sequence ID. We notice that the result of previous studies on Transformers (Yun et al., 2020; Kim et al., 2023) cannot be directly applied to Looped Transformers due to the following distinctions:

- Yun et al. (2020) employed both sparse and uniform attention mechanisms, whereas Looped Transformers are limited to a single fixed attention layer.
- Kim et al. (2023) used N layers to store N parameters required for representing the target function, whereas Looped Transformers have a fixed parameter size.

Notably, we found that Looped Transformers with N-loops can compute contextual mapping. Let $\boldsymbol{z} \in \{0,1,\ldots,\delta^{-d}-1\}^N$ represent a sequence consist of N ordered and distinct token IDs, satisfying $\boldsymbol{z}_1 > \boldsymbol{z}_2 > \cdots > \boldsymbol{z}_N$. The network then maps $\boldsymbol{z}$ to a sequence ID through an inner product with $\boldsymbol{u} = (\delta^{-d(N-1)},\ldots,\delta^{-d},1)$, which satisfies

$$|\boldsymbol{u}^\top \boldsymbol{z} - \boldsymbol{u}^\top \boldsymbol{z}'| > 1, \quad \text{if } \boldsymbol{z} \neq \boldsymbol{z}'. \tag{3}$$

This guarantees that the network assigns distinct sequence IDs for different $\boldsymbol{z}$. Combined with token IDs, the network computes contextual mapping. The key idea is that the network requires only δ^{-d} to represent $\boldsymbol{u}$, allowing it to be implemented with Looped Transformers.

Step 3. Function Value Mapping. The network maps the contextual token IDs into the target embeddings in a token-wise manner, using $K-1$ loops to sequentially map $k = 0,1,\ldots,K-1$ to $\tilde{\boldsymbol{y}}_k \in \mathbb{R}^d$, which approximates $\boldsymbol{y}_k$, in each iteration. In our constructive proofs, we design both the set of contextual token IDs and their ordering.

Weight-tied feed-forward networks cannot map accurately, and the error can only be bounded by the maximum difference between adjacent contextual token embeddings, *i.e.*

$$|(\tilde{\boldsymbol{y}}_k - \boldsymbol{y}_k)_i| \leq \max_{k'\in\mathcal{K}} |(\boldsymbol{y}_{k'} - \boldsymbol{y}_{k'-1})_i| \tag{4}$$

holds for $k \in \mathcal{K}$ and $i = 1,\ldots,d$.

Generally, the following inequality holds, for $\boldsymbol{x} \in \mathbb{R}^m$,

$$\max_i |\boldsymbol{x}_i| \leq \|\boldsymbol{x}\|_p \leq m^{\frac{1}{p}} \max_i |\boldsymbol{x}_i|. \tag{5}$$

That is, by controlling the p-norm, $\|\boldsymbol{y}_k - \boldsymbol{y}_{k-1}\|_p$, the error in Eq. 4 can bounded. We require $\mathcal{K}$ to be designed such that the differences between neighboring contextual token embeddings are bounded *w.r.t.*the p-norm.

To illustrate our idea, consider the following sentences:

(1) I <u>write</u> papers. ; I write <u>papers</u>. (different token ID with same sequence ID)

(2) I <u>write</u> papers. ; You <u>write</u> books. (same token ID with different sequence ID)

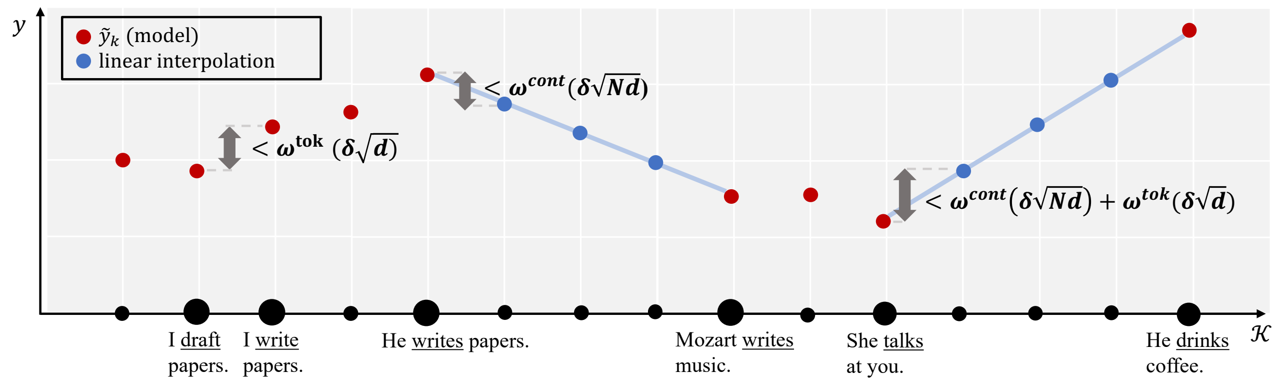


Figure 2. Approximation error and modulus of continuity. The linear interpolation technique reduces the error by a factor of $1/\delta^{-1}$.

We found that none of the moduli of continuity, defined in Section 3.2, alone can bound the difference between 'write and 'papers' in (1). In contrast, the error of 'write' in (2) can be bounded by the contextual continuity, ω_f^{cont}. Thus, we designed contextual token IDs such that, basically, identical or similar tokens with different sequence IDs are positioned adjacent to each other, as shown in Fig. 2. To reduce errors in corner cases, linear interpolation is applied; further details are provided in Appendix A. This allows us to obtain the following error bound.

$$\max_{k' \in \mathcal{K}} \|\boldsymbol{y}_{k'} - \boldsymbol{y}_{k'-1}\|_p \leq \omega_f^{\text{tok}}(\delta\sqrt{d}) + \omega_f^{\text{cont}}(\delta\sqrt{Nd}). \quad (6)$$

Substituting $\boldsymbol{x} = \boldsymbol{y}_k - \boldsymbol{y}_{k-1}$ into Eq. 5, with Eq. 4 and Eq. 6, the following result holds:

$$|(\tilde{\boldsymbol{y}}_k - \boldsymbol{y}_k)_i| \leq \omega_f^{\text{tok}}(\delta\sqrt{d}) + \omega_f^{\text{cont}}(\delta\sqrt{Nd}), \quad (7)$$

for $i = 1, \dots, d$ and $k \in \mathcal{K}$.

Concatenated into a Single Transformer Layer In the final construction, we show that the composition of the three sub-networks from Steps 1, 2, and 3 can be implemented within a single Transformer block. While our proof strategy follows Zhang et al. (2023), their approach necessitates an additional layer. In contrast, we show that a single Transformer block suffices, as detailed in Appendix A.

Deriving Approximation Rate Lastly, we analyze the approximation error of our construction and establish the approximation rate in terms of the number of loops.

With the triangle inequality, we obtain the following:

$$\|\tilde{f} - f\|_{L^p} \leq \int \|\tilde{f}(\boldsymbol{X}) - f(\boldsymbol{X})\|_p d\boldsymbol{X} \quad (8)$$

$$\leq \int \|\tilde{f}(\boldsymbol{X}) - \bar{f}(\boldsymbol{X})\|_p d\boldsymbol{X} + \int \|\bar{f}(\boldsymbol{X}) - f(\boldsymbol{X})\|_p d\boldsymbol{X}$$
$$+ \mathcal{O}(N^{2/p}\delta^{d/p}) + \mathcal{O}\left(\left((M\delta)^{-1}dN\right)^{1/p}\right), \quad (9)$$

where $\mathcal{O}(N^{2/p}\delta^{d/p})$ arises from the case where identical tokens appear in sequences, and $\mathcal{O}\left(\left((M\delta)^{-1}dN\right)^{1/p}\right)$ results from the restriction on the norm of weights.

Considering the error within cubes in Eq. 1, we obtain

$$\int \|\bar{f}(\boldsymbol{X}) - f(\boldsymbol{X})\|_p d\boldsymbol{X} \leq \omega_f(\delta\sqrt{Nd}). \quad (10)$$

Since, generally, the norm of sequences can be bounded by the maximum norm of the token-wise vectors as

$$\|f(\boldsymbol{X})\|_p \leq (Nd)^{\frac{1}{p}} \max_{i,n} |f(\boldsymbol{X})_{i,n}|, \quad (11)$$

the error between $\tilde{f}$ and $\bar{f}$ can be bounded by

$$\int \|\tilde{f}(\boldsymbol{X}) - \bar{f}(\boldsymbol{X})\|_p d\boldsymbol{X} \leq (Nd)^{\frac{1}{p}} \max_{k' \in \mathcal{K}} |\tilde{\boldsymbol{y}}_{k'} - \boldsymbol{y}_{k'}|. \quad (12)$$

Substituting $\boldsymbol{x} = \tilde{\boldsymbol{y}}_k - \boldsymbol{y}_k$ into Eq. 5, and using Eq. 7 and Eq. 12, we obtain:

$$\int \|\tilde{f}(\boldsymbol{X}) - \bar{f}(\boldsymbol{X})\|_p d\boldsymbol{X}$$
$$\leq (Nd)^{\frac{1}{p}} \left(\omega_f^{\text{tok}}(\delta\sqrt{d}) + \omega_f^{\text{cont}}(\delta\sqrt{Nd})\right). \quad (13)$$

We then express δ in terms of the number of loops r to determine the approximation rate. We use $\delta^{-1} - 1$ loops for Step 1, N loops for Step 2, and $2\delta^{-(N+1)d} - 1$ loops for Step 3, with 1 loop required to connect each step *i.e.*

$$r = \delta^{-1} + 2\delta^{-(N+1)d} + N. \quad (14)$$

Now, δ can be bounded in terms of the number of loops as:

$$\delta^{-1} + 2\delta^{-(N+1)d} = r - N \quad (15)$$
$$\Rightarrow \delta^{-1} \cdot 2\delta^{-(N+1)d} \geq r - N \quad (\delta^{-1} \geq 2) \quad (16)$$
$$\Leftrightarrow \delta \leq \left(\frac{r - N}{2}\right)^{-1/((N+1)d+1)}. \quad (17)$$

By combining Eq. 10 and Eq 13 with Eq. 9, and substituting Eq. 17, we obtain Theorem 3.6.

4. From Theory to Practice: Introducing Timestep Encoding

The theoretical result in Section 3 highlights a limitation of the looped architecture. We show that a variant of architecture can overcome this limitation.

4.1. Motivation

Limitation Specific to Looped Transformers The approximation rate in Theorem 3.6 includes two additional moduli of continuity, which can lead to increased errors, reflecting a limitation inherent to Looped Transformers.

We can identify the cause of additional dependency in the error in Eq. 4, caused by weight-tied feed-forward networks. This can be formalized as follows:

Lemma 4.1. *Given $\boldsymbol{y}_k \in \mathbb{R}^d$ for $k = 0, 1, \ldots, K-1$ with*

$$|(\boldsymbol{y}_k - \boldsymbol{y}_{k-1})_i| \leq \varepsilon_i \quad \text{for } i = 1, \ldots, d,$$

there exists a feed-forward layer $\mathrm{FF} : \mathbb{R}^{12d} \to \mathbb{R}^{12d}$ with a width size of $18d$, and two affine linear maps $\mathcal{L}_1 : \mathbb{R} \to \mathbb{R}^{12d}$ and $\mathcal{L}_2 : \mathbb{R}^{12d} \to \mathbb{R}^d$ s.t.

$$\left|\left(\mathcal{L}_2 \circ (\mathrm{id} + \mathrm{FF})^{\circ(K-1)} \circ \mathcal{L}_1(k) - \boldsymbol{y}_k\right)_i\right| \leq \varepsilon_i, \tag{18}$$

for $i = 1, \ldots, d$ and $k = 0, 1, \ldots, K-1$.

This shows that large variations in the target function may lead to approximation errors, raising the question of whether inequality in Eq. 18 can be replaced with equality.

Improving Approximation Rate of Looped Transformers To eliminate this dependency, we introduce *time-dependent* parameters. Specifically, we modify the feed-forward layers by adding a scaling vector for each loop step as follows:

$$\mathrm{FF}(\boldsymbol{X}) \to \boldsymbol{\eta}(t) \odot \mathrm{FF}(\boldsymbol{X}) \quad \text{for the } t\text{-th loops},$$

where $\odot$ is an element-wise product,$t \in \mathbb{N}$ is the loop index, and $\boldsymbol{\eta}(t) \in \mathbb{R}^d$ is the scaling parameter for each loop. This method is analogous to Hypernetworks (Ha et al., 2016). With the definition

$$(\mathrm{id} + \boldsymbol{\eta} \odot \mathrm{FF})^r := (\mathrm{id} + \boldsymbol{\eta}(r) \odot \mathrm{FF}) \circ \cdots \circ (\mathrm{id} + \boldsymbol{\eta}(1) \odot \mathrm{FF}),$$

we show that this model can memorize labels exactly.

Theorem 4.2. *Given $\boldsymbol{y}_k \in \mathbb{R}^d$ for $k = 0, 1, \ldots, K-1$, there exists a feed-forward layer $\mathrm{FF} : \mathbb{R}^{4d} \to \mathbb{R}^{4d}$ with a width size of $6d$, $\boldsymbol{\eta}(t) \in \mathbb{R}^{4d}$ for $t = 1, \ldots, K-1$, and two affine linear maps $\mathcal{L}_1 : \mathbb{R} \to \mathbb{R}^{4d}$ and $\mathcal{L}_2 : \mathbb{R}^{4d} \to \mathbb{R}^d$ s.t.*

$$\left|\left(\mathcal{L}_2 \circ (\mathrm{id} + \boldsymbol{\eta} \odot \mathrm{FF})^{\circ(K-1)} \circ \mathcal{L}_1(k) - \boldsymbol{y}_k\right)_i\right| = 0,$$

for $i = 1, \ldots, d$ and $k = 0, 1, \ldots, K-1$.

The proof is provided in Appendix B. For implementation, adding parameters per loop increases the total parameter count proportionally. Thus, we introduce timestep encoding.

4.2. Timestep-Modulated Looped Transformer

We employ timestep encodings to condition scaling parameters on the loop index (timestep). This method is inspired by adaptive instance normalization (Peebles & Xie, 2023).

To condition on timesteps, frequency embeddings are processed through a two-layer MLP with hidden size matching the Transformer block and SiLU activation. Let $\mathrm{TE}(t) \in \mathbb{R}^d$ denote timestep embeddings, defined as:

$$\mathrm{TE}(t) = \boldsymbol{W}_3 \cdot \mathrm{SiLU}(\boldsymbol{W}_4 \cdot \mathrm{PE}(t) + \boldsymbol{b}_4) + \boldsymbol{b}_3,$$

where $\boldsymbol{W}_3, \boldsymbol{W}_4 \in \mathbb{R}^{d \times d}$, $\boldsymbol{b}_3, \boldsymbol{b}_4 \in \mathbb{R}^d$, and $\mathrm{PE}(t) \in \mathbb{R}^d$ denotes the timestep encoding function that maps the timestep into a d-dimensional embedding, *s.t.*

$$\mathrm{PE}(t)_{2i} = \sin(t/10000^{2i/d}),$$
$$\mathrm{PE}(t)_{2i+1} = \cos(t/10000^{2i/d}).$$

We use the root mean square layer normalization (RMSNorm) (Zhang & Sennrich, 2019), which is widely used in several recent LLMs (et al., 2023; Team, 2024), defined as:

$$\bar{\boldsymbol{x}} = \boldsymbol{\alpha} \odot \frac{\boldsymbol{x}}{\mathrm{RMS}(\boldsymbol{x})}, \text{ where } \mathrm{RMS}(\boldsymbol{x}) = \sqrt{\frac{1}{d}\sum_{i=1}^{d} \boldsymbol{x}_i^2},$$

where $\boldsymbol{\alpha} \in \mathbb{R}^d$ is a gain parameter for rescaling. We define time-dependent RMSNorm, denoted by RMSN, as:

$$\mathrm{RMSN}(\boldsymbol{x}, t) = \boldsymbol{\alpha}(t) \odot \frac{\boldsymbol{x}}{\mathrm{RMS}(\mathbf{x})}$$

where $\boldsymbol{\alpha}(t) \in \mathbb{R}^d$ is a time-dependent parameter generated by a network. With scaling parameters, the time-dependent Transformer block is defined as follows:

$$\boldsymbol{X}' = \boldsymbol{X} + \boldsymbol{\gamma}_1(t) \odot \mathrm{Attn}(\mathbf{RMSN}_1(\boldsymbol{X}, t)),$$
$$\mathrm{TF}(\boldsymbol{X}, t) = \boldsymbol{X}' + \boldsymbol{\gamma}_2(t) \odot \mathbf{FF}(\mathbf{RMSN}_2(\boldsymbol{X}', t)),$$

where $\gamma_1(t), \gamma_2(t) \in \mathbb{R}^d$ are time-dependent parameters applied token-wise, as well as RMSNorm.

The time-dependent vector parameters are generated as:

$$\boldsymbol{\alpha}_1(t), \boldsymbol{\alpha}_2(t), \boldsymbol{\gamma}_1(t), \boldsymbol{\gamma}_2(t) = \boldsymbol{W}_5 \cdot \mathrm{SiLU}(\mathrm{TE}(t)) + \boldsymbol{b}_5,$$

where $\boldsymbol{W}_5 \in \mathbb{R}^{4d \times d}$ and $\boldsymbol{b}_5 \in \mathbb{R}^d$.

5. Experiments

This section presents experimental results supporting our theoretical findings. We used Looped Transformers with varying numbers of loops, both with and without timestep encoding, and compared to standard Transformers. We assess approximation capabilities based on test evaluation, as we observe a strong correlation between train and test performance. The details are provided in Appendix C.

Table 2. Test accuracy for reasoning tasks. Performance improves as the number of loops increases..

Task	**TF**	**Looped TF**				**w/ Timestep Encoding**			
	L=6	**r=4**	**r=8**	**r=16**	**r=32**	**r=4**	**r=8**	**r=16**	**r=32**
Sudoku	0.0	0.0	0.0	65.6	87.9	0.0	0.0	62.0	**90.2**
Countdown	53.8	28.3	52.7	81.0	88.1	33.2	54.4	80.2	**90.5**
	L=12	**r=5**	**r=10**	**r=50**	**r=100**	**r=5**	**r=10**	**r=50**	**r=100**
LCS (60)	70.0	66.0	81.8	98.6	96.9	68.5	80.5	**99.3**	97.1
LCS (100)	39.8	39.6	45.1	93.5	98.2	36.7	45.6	98.1	**98.6**
ED (40)	54.2	41.4	57.9	85.4	90.4	44.8	63.5	94.5	**96.1**
ED (60)	41.4	23.8	32.6	47.3	47.7	26.6	38.9	57.3	**88.3**

5.1. Problem Setting

We evaluate the model on two types of tasks. The first consists of reasoning problems known to be challenging for standard Transformers. These are used to examine whether increasing the number of loops and incorporating timestep encodings can enhance performance. The second includes core Transformer benchmarks, such as in-context learning and language modeling.

5.1.1. Reasoning Tasks

Dynamic Programming is a method for solving complex problems by breaking them down into simpler sub-problems. We use edit distance (ED) and longest common subsequence (LCS) tasks with varying input lengths. Each task has 10^6 train samples and 10^3 test samples.

Sudoku is a constrained satisfaction problem that involves filling a 9×9 grid with digits from 1 to 9, such that each digit appears exactly once in every row, column, and predefined 3×3 sub-grid. The grid is flattened into a sequence representation. Unlike (Yang et al., 2023), we use the dataset from (Radcliffe, 2020), sampling 3M instances for training and 100K for testing.

Countdown is a game in which a given set of input numbers must be combined using basic arithmetic operations to reach a target number (Yao et al., 2023; Gandhi et al., 2024). We consider cases with 4 input numbers and target numbers ranging from 10 to 100, where 10% of the target numbers are reserved for evaluation. We generate 5M samples for training and 1K samples for testing.

5.1.2. In-Context and Language Modeling

The in-context learning problem is to learn the function class from a given sequence, which was investigated with Looped Transformers (Yang et al., 2024) without timestep encodings. We use decision tree functions. For the language modeling task, we use the WikiText-103 (Merity et al., 2017) dataset, containing over 100 million tokens from Wikipedia articles. Details are in Appendix C.2 and Appendix C.3.

5.2. Results

The results in Table 2 demonstrate that increasing the number of loops improves performance on reasoning tasks, with higher loop counts significantly outperforming standard Transformers. Furthermore, incorporating timestep encodings leads to additional gains; in particular, for the edit distance task with input size $n = 60$, the model with loop counts $r = 100$ achieves significantly better performance when timestep encodings are incorporated.

Table 3. MSE (↓) on the in-context learning task.

	TF L=12	**Looped r=12**	**w/ Timestep r=12**
Test	8.6e-03	1.4e-02	**1.7e-03**

Table 4. Perplexity (↓) on the WikiText-103 dataset.

	TF L=12	**Looped r=24**	**w/ Timestep r=24**
Train	**15.9**	17.1	**15.9**
Test	20.5	20.6	**19.6**

As evidenced by the results in Table 3 and Table 4, the use of timestep encodings leads to performance gains in both in-context learning and language modeling.

6. Conclusion

We establish the approximation rate of Looped Transformers with respect to the number of loops and the moduli of continuity of the target function. Our analysis reveals a limitation of Looped Transformers, which is addressed by timestep encodings. To the best of our knowledge, this study is the first to investigate the function approximation capabilities of Looped Transformers. Extending the analysis to multiple layers, varying input lengths, and characterizing optimal memorization capacity presents promising avenues for future research. Beyond expressivity, investigating estimation performance and enhancing training stability constitute important challenges moving forward.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number JP24H00709.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Bae, S., Fisch, A., Harutyunyan, H., Ji, Z., Kim, S., and Schuster, T. Relaxed recursive transformers: Effective parameter sharing with layer-wise loRA. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=WwpYSOkkCt.

Bai, S., Kolter, J. Z., and Koltun, V. Deep equilibrium models. *Advances in neural information processing systems*, 32, 2019.

Barron, A. R. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.

Bartlett, P., Maiorov, V., and Meir, R. Almost linear vc dimension bounds for piecewise polynomial networks. *Advances in neural information processing systems*, 11, 1998.

Csordás, R., Irie, K., Schmidhuber, J., Potts, C., and Manning, C. D. MoEUT: Mixture-of-experts universal transformers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=ZxVrkm7Bjl.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, L. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Deletang, G., Ruoss, A., Grau-Moya, J., Genewein, T., Wenliang, L. K., Catt, E., Cundy, C., Hutter, M., Legg, S., Veness, J., and Ortega, P. A. Neural networks and the chomsky hierarchy. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WbxHAzkeQcn.

et al., H. T. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Fan, Y., Du, Y., Ramchandran, K., and Lee, K. Looped transformers for length generalization, 2024a. URL https://arxiv.org/abs/2409.15647.

Fan, Y., Du, Y., Ramchandran, K., and Lee, K. Looped transformers for length generalization, 2024b. URL https://arxiv.org/abs/2409.15647.

Feng, G., Zhang, B., Gu, Y., Ye, H., He, D., and Wang, L. Towards revealing the mystery behind chain of thought: A theoretical perspective. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=qHrADgAdYu.

Gandhi, K., Lee, D. H. J., Grand, G., Liu, M., Cheng, W., Sharma, A., and Goodman, N. Stream of search (sos): Learning to search in language. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=2cop2jmQVL.

Garg, S., Tsipras, D., Liang, P., and Valiant, G. What can transformers learn in-context? a case study of simple function classes. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=flNZJ2eOet.

Giannou, A., Rajput, S., Sohn, J.-Y., Lee, K., Lee, J. D., and Papailiopoulos, D. Looped transformers as programmable computers. In *Proceedings of the 40th International Conference on Machine Learning*, 2023. URL https://proceedings.mlr.press/v202/giannou23a.html.

Giannou, A., Yang, L., Wang, T., Papailiopoulos, D., and Lee, J. D. How well can transformers emulate in-context newton's method? *arXiv preprint arXiv:2403.03183*, 2024. URL https://arxiv.org/abs/2403.03183.

Ha, D., Dai, A., and Le, Q. V. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Jiang, H. and Li, Q. Approximation rate of the transformer architecture for sequence modeling. *arXiv preprint arXiv:2305.18475*, 2024. URL https://arxiv.org/abs/2305.18475.

Kajitsuka, T. and Sato, I. Are transformers with one layer self-attention using low-rank weight matrices universal approximators? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=nJnky5K944.

Kambhampati, S., Valmeekam, K., Guan, L., Verma, M., Stechly, K., Bhambri, S., Saldyt, L. P., and Murthy, A. B. Position: LLMs can't plan, but can help planning in LLM-modulo frameworks. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=Th8JPEmH4z.

Kim, J., Kim, M., and Mozafari, B. Provable memorization capacity of transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=8JCg5xJCTPR.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Loshchilov, I. and Hutter, F. Fixing weight decay regularization in adam, 2018. URL https://openreview.net/forum?id=rk6qdGgCZ.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Merrill, W. and Sabharwal, A. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545, 2023.

Peebles, W. and Xie, S. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Pope, R., Douglas, S., Chowdhery, A., Devlin, J., Bradbury, J., Levskaya, A., Heek, J., Xiao, K., Agrawal, S., and Dean, J. Efficiently scaling transformer inference, 2022. URL https://arxiv.org/abs/2211.05102.

Radcliffe, D. G. 3 million sudoku puzzles with ratings, 2020.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

Saunshi, N., Dikkala, N., Li, Z., Kumar, S., and Reddi, S. J. Reasoning with latent thoughts: On the power of looped transformers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=din0lGfZFd.

Takakura, S. and Suzuki, T. Approximation and estimation ability of transformers for sequence-to-sequence functions with infinite dimensional input. In *Proceedings of the 40th International Conference on Machine Learning*, 2023. URL https://proceedings.mlr.press/v202/takakura23a.html.

Takase, S. and Kiyono, S. Lessons on parameter sharing across layers in transformers. *arXiv preprint arXiv:2104.06022*, 2021.

Team, G. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

Vardi, G., Yehudai, G., and Shamir, O. On the optimal memorization power of reLU neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=MkTPtnjeYTV.

Vaswani, A., Shazeer, N. M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Neural Information Processing Systems*, 2017.

Wang, M. and E, W. Understanding the expressive power and mechanisms of transformer for sequence modeling. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=0o7Rd5jngV.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., brian ichter, Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. Chain of thought prompting elicits reasoning in large language models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

Yang, L., Lee, K., Nowak, R. D., and Papailiopoulos, D. Looped transformers are better at learning learning algorithms. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=HHbRxoDTxE.

Yang, Z., Ishay, A., and Lee, J. Learning to solve constraint satisfaction problems with recurrent transformer. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=udNhDCr2KQe.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. R. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=5Xc1ecxO1h.

Yarotsky, D. Optimal approximation of continuous functions by very deep relu networks. In *Conference on learning theory*, pp. 639–649. PMLR, 2018.

Yun, C., Bhojanapalli, S., Rawat, A. S., Reddi, S., and Kumar, S. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ByxRM0Ntvr.

Zhang, B. and Sennrich, R. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

Zhang, S., Lu, J., and Zhao, H. On enhancing expressive power via compositions of single fixed-size relu network. In *International Conference on Machine Learning*, pp. 41452–41487. PMLR, 2023.

A. Proofs for Theorem 3.6

The main theorem incorporates a restriction on the norm of weights, leading to errors when approximating discontinuous functions, such as step functions with ReLU or hardmax functions with softmax. We first establish the approximation rate assuming that weights can take arbitrary precision real values, as outlined below. Then, we account for the bit complexity of bounded weights to complete the proof of the main Theorem 3.6.

Theorem A.1. *Given a function $f \in \mathcal{F}_{\mathrm{PE}}([0,1]^{d\times N})$, $r > N$, there exists a Looped Transformer, composed of* TF : $\mathbb{R}^{(17d+9)\times N} \to \mathbb{R}^{(17d+9)\times N}$ *with two heads, head size* 1*, a width size of $q = 49d + 25$, and two affine linear maps $\mathcal{L}_1 : \mathbb{R}^d \to \mathbb{R}^{17d+9}$ and $\mathcal{L}_2 : \mathbb{R}^{17d+9} \to \mathbb{R}^d$ s.t.*

$$\left\| \boldsymbol{\mathcal{L}}_2 \circ \mathrm{TF}^{\circ r} \circ \boldsymbol{\mathcal{L}}_1 - f \right\|_{L^p} \le (Nd)^{\frac{1}{p}} \left(\omega_f^{\mathrm{tok}}(\delta\sqrt{d}) + \omega_f^{\mathrm{cont}}(\delta\sqrt{Nd}) \right) + \omega_f(\delta\sqrt{Nd}) + \mathcal{O}(N^{2/p}\delta^{d/p}), \tag{19}$$

where $\delta = \left((r-N)/2\right)^{-1/((N+1)d+1)}$.

A.1. Proof of Theorem A.1

Proof. Since any continuous function can be approximated by a piecewise constant function with arbitrarily small errors, we approximate $f \in \mathcal{F}_{\mathrm{PE}}([0,1]^{d\times N})$ with piece-wise constant function $\bar{f} : [0,1]^{d\times N} \to \mathbb{R}^{d\times N}$. We choose $\delta^{-1} \in \mathbb{N}, \delta^{-1} \ge 2$, determining how finely the input is divided: we divide the input space $[0,1]^{d\times N}$ into *δ-discretized cubes*, denoted by $\{\boldsymbol{Q}_{\boldsymbol{\mathcal{B}}}\}$ for $\boldsymbol{\mathcal{B}} \in \{0,1,\dots,\delta^{-1}-1\}^{d\times N}$ defined by

$$\boldsymbol{Q}_{\boldsymbol{\mathcal{B}}} \coloneqq \Big\{ \mathbf{X} \in [0,1]^{d\times N} : \mathbf{X}_{i,n} \in \big[\boldsymbol{\mathcal{B}}_{i,n}\delta, (\boldsymbol{\mathcal{B}}_{i,n}+1)\delta\big), \quad i = 1,2,\dots,d,\ n = 1,2,\dots,N \Big\}. \tag{20}$$

Each cube $\boldsymbol{Q}_{\boldsymbol{\mathcal{B}}}$ is associated with a representative point $\hat{\boldsymbol{X}}_{\boldsymbol{\mathcal{B}}}$, defined as the vertex of $\boldsymbol{Q}_{\boldsymbol{\mathcal{B}}}$ with the minimal ℓ_1 norm. Then, we define the piecewise constant function $\bar{f}$ for $\boldsymbol{X} \in [0,1]^{d\times N}$ as

$$\bar{f}(\boldsymbol{X}) \coloneqq f(\hat{\boldsymbol{X}}_{\boldsymbol{\mathcal{B}}}). \tag{21}$$

Since we can bound the error within each cube, we have:

$$\max_{\boldsymbol{X}\in[0,1]^{d\times N}} \{\|\bar{f}(\boldsymbol{X}) - f(\boldsymbol{X})\|_p\} \le \omega_f(\delta\sqrt{Nd}). \tag{22}$$

Our construction consists of three steps to approximate $\bar{f}$, as outlined below.

1. The network, with $\delta^{-1}-1$ loops, maps the input space $[0,1]^d$ token-wise to the coordinates $\boldsymbol{\beta} \in \{0,1,\dots,\delta^{-1}-1\}^d$ of discretized cubes, and then bijectively maps these coordinates to integers, representing *token IDs* in the set $\{0,1,\dots,\delta^{-d}-1\}$, using a δ^{-1}-base system; for example, if $d = 2$ and $\delta^{-1} = 3$, then coordinates $(\beta_1,\beta_2) = (2,1)$ are mapped to the integer $2\times 3^1 + 1\times 3^0 = 7$.

2. The network, with N loops, computes a contextual mapping from the set of N distinct token IDs into the set of *contextual token ID*. *Contextual token IDs* refer to token IDs assigned to each token within the context of a *sequence ID*.

3. The network, with $2\delta^{-(N+1)d}-1$ loops, approximately maps *contextual token IDs* into the output embeddings of each token in a token-wise manner. To achieve a small approximation error, the network has to be designed so that neighboring IDs correspond to similar output token embeddings. Furthermore, *dummy indices* are used to reduce the error.

The details for each step are provided below.

Step 1. Token-wise Quantization. The input space for each token $\boldsymbol{x} \in [0,1]^d$ are divided into δ-discretized cubes denoted by $\{Q_{\boldsymbol{\beta}}\}$ for $\boldsymbol{\beta} \in \{0,1,\dots,\delta^{-1}-1\}^d$, defined as

$$Q_{\boldsymbol{\beta}} \coloneqq \Big\{ \boldsymbol{x} \in [0,1]^d : \boldsymbol{x}_i \in \big[\boldsymbol{\beta}_i\delta,\ (\boldsymbol{\beta}_i+1)\delta\big), \quad \text{for all } i = 1,2,\dots,d \Big\}. \tag{23}$$

By Lemma A.5, there exists a feed-forward layer $\mathrm{FF}^{(1)} : \mathbb{R}^{5d} \to \mathbb{R}^{5d}$ of width size $q = 7d$, and two affine linear maps $\mathcal{L}_1^{(1)} : \mathbb{R}^d \to \mathbb{R}^{5d}$ and $\mathcal{L}_2'^{(1)} : \mathbb{R}^{5d} \to \mathbb{R}^d$ such that

$$\mathcal{L}_2'^{(1)} \circ \left(\mathrm{id} + \mathrm{FF}^{(1)}\right)^{\circ(\delta^{-1}-1)} \circ \mathcal{L}_1^{(1)}(\boldsymbol{x}) = \boldsymbol{\beta} \quad \textit{s.t.} \quad \boldsymbol{x} \in Q_{\boldsymbol{\beta}}. \tag{24}$$

In addition, we need to bijectively map the d-dimensional vector $\boldsymbol{\beta}$ to an integer *token ID*, denoted by z. We use a δ^{-1}-*base system*: we define the vector $\boldsymbol{u}_{(\delta^{-1})} \in \mathbb{R}^d$ as

$$\boldsymbol{u}_{(\delta^{-1})} \coloneqq (\delta^{-(d-1)}, \delta^{-(d-2)}, \dots, \delta^{-1}, 1)^\top, \tag{25}$$

and define z as

$$z \coloneqq \boldsymbol{u}_{(\delta^{-1})}^\top \boldsymbol{\beta} \quad \in \{0, 1, \dots, \delta^{-d} - 1\}. \tag{26}$$

To implement this, we define an affine linear map $\mathcal{L}_2^{(1)} : \mathbb{R}^{5d} \to \mathbb{R}$ via

$$\mathcal{L}_2^{(1)}(\boldsymbol{x}) = \boldsymbol{u}_{(\delta^{-1})}^\top \mathcal{L}_2'^{(1)}(\boldsymbol{x}). \tag{27}$$

Thus, we have

$$\left(\mathcal{L}_2^{(1)} \circ (\mathrm{FF}_1^{(1)})^{\circ(\delta^{-1}-1)} \circ \mathcal{L}_1^{(1)}(\boldsymbol{x})\right)_n = \boldsymbol{u}_{(\delta^{-1})}^\top \boldsymbol{\beta} = z \quad \textit{s.t.} \quad \boldsymbol{x} \in Q_{\boldsymbol{\beta}}. \tag{28}$$

We establish an upper bound on the maximum distance in input space between adjacent token IDs to derive the approximation error for the following steps. Define the input cubes corresponding to each token ID z as follows:

$$Q_z \coloneqq \left\{\boldsymbol{x} \in [0,1]^d : \boldsymbol{x}_i \in \left[\boldsymbol{\beta}_i\delta,\ (\boldsymbol{\beta}_i + 1)\delta\right) \quad \text{for } i = 1, 2, \dots, d \quad \textit{s.t.} \quad z = \boldsymbol{u}_{(\delta^{-1})}^\top \boldsymbol{\beta}\right\}. \tag{29}$$

Then we have

$$\max_{z, \boldsymbol{x} \in Q_z, \boldsymbol{x}' \in Q_{z-1}} \|\boldsymbol{x} - \boldsymbol{x}'\|_2 \leq \begin{cases} \delta\sqrt{d}, & \text{if } \boldsymbol{\beta}_d \in \{1, 2, \dots, \delta^{-1} - 1\}, \\ \sqrt{d}, & \text{if } \boldsymbol{\beta}_d = 0. \end{cases} \tag{30}$$

Informally, in this δ^{-1}-based representation, the least significant digit corresponds to the index of the d-th dimension, $\boldsymbol{\beta}_d$. As the token ID increments sequentially, the index in the d-th dimension increases as $0, 1, 2, \dots, \delta^{-1} - 1$, while the higher-order digits remain unchanged. Consequently, consecutive token IDs correspond to tokens that are "similar" in the d-dimensional space, with a maximum distance of $\delta\sqrt{d}$. However, when a carry occurs, the higher-order digits may change significantly, leading to cases where tokens that are not adjacent in input space become adjacent in their indices. In such cases, the distance is only bounded by $\sqrt{d}$.

Step 2. Contextual Mapping. The networks, with N-loops, map the list of N token IDs, denoted by $\boldsymbol{z} \in \{0, 1, \dots, \delta^{-d} - 1\}^N$, into sequence IDs bijectively. Combined with token IDs, the network computes contextual mapping.

We consider only the case where all N input tokens are distinct, disregarding other cases, as they can be treated as negligible when δ is small. The number of subsets in which at least one of the N tokens is duplicated is given by

$$(\delta^{-d})^N - \left(\delta^{-d} \cdot (\delta^{-d} - 1) \cdots (\delta^{-d} - N + 1)\right) < \frac{N(N-1)}{2}\delta^{-(N-1)d}, \tag{31}$$

when δ is sufficiently small. The volume of these subsets is bounded by

$$\frac{C\delta^{-(N-1)d}}{\delta^{-Nd}} = \mathcal{O}(N^2\delta^d). \tag{32}$$

Thus, the error with respect to the L^p norm is bounded by $\mathcal{O}(N^{2/p}\delta^{d/p})$.

Let $\mathbb{L}_\delta$ denote the set of N distinct token IDs, *i.e.*

$$\mathbb{L}_\delta \coloneqq \{\boldsymbol{z} \in \{0, 1, \dots, \delta^{-d} - 1\}^N \mid \boldsymbol{z}_i \neq \boldsymbol{z}_j \text{ for all } i \neq j\}. \tag{33}$$

Due to permutation equivariance, we can assume without loss of generality that elements of $\boldsymbol{z} \in \mathbb{L}_\delta$ are ordered, *i.e.*, $\boldsymbol{z}_1 > \boldsymbol{z}_2 > \cdots > \boldsymbol{z}_N$. Define $\boldsymbol{u}_{(\delta^{-d})} := (\delta^{-(N-1)d}, \ldots, \delta^{-d}, 1)^\top$, which satisfy

$$|\boldsymbol{u}_{(\delta^{-d})}^\top \boldsymbol{z} - \boldsymbol{u}_{(\delta^{-d})}^\top \boldsymbol{z}'| > 1, \quad \text{for any } \boldsymbol{z}, \boldsymbol{z}' \in \mathbb{L}_\delta \text{ with } \boldsymbol{z} \neq \boldsymbol{z}'. \tag{34}$$

This mapping, $\boldsymbol{u}_{(\delta^{-d})}^\top \boldsymbol{z}$, represents $\boldsymbol{z}$ in a δ^{-d}-base system. Then, we define sequence ID for $\boldsymbol{z} \in \mathbb{L}_\delta$ as:

$$\mathrm{s}(\boldsymbol{z}) := \boldsymbol{u}_{(\delta^{-d})}^\top \boldsymbol{z} = \sum_{n=1}^{N} \boldsymbol{z}_n \delta^{-(N-n)d}. \tag{35}$$

By Lemma A.6, there exists a Transformer block $\mathrm{TF}'^{(2)} : \mathbb{R}^{5\times N} \to \mathbb{R}^{5\times N}$ with single head, head size $s = 1$, and width size $q = 3$, and two affine linear maps $\mathcal{L}_1'^{(2)} : \mathbb{R} \to \mathbb{R}^5$ and $\mathcal{L}_2'^{(2)} : \mathbb{R}^5 \to \mathbb{R}$ such that

$$\boldsymbol{\mathcal{L}}_2'^{(2)} \circ \left(\mathrm{TF}'^{(2)}\right)^{\circ N} \circ \boldsymbol{\mathcal{L}}_1'^{(2)}(\boldsymbol{z}^\top) = \mathrm{s}(\boldsymbol{z}) \cdot \mathbf{1}_N^\top. \tag{36}$$

Furthermore, we have to add *dummy indices* to alleviate the approximation error caused by the looped architecture in Step 3. Recall that $\boldsymbol{\mathcal{B}} \in \{0, 1, \ldots, \delta^{-1} - 1\}^{d\times N}$ represents the coordinates of the inputs. Let $\boldsymbol{Z}_{\boldsymbol{\mathcal{B}}} \in \{0, 1, \ldots, \delta^{-1} - 1\}^{d\times N}$ denote the ordered coordinates of $\boldsymbol{\mathcal{B}}$ where the tokens are ordered by their token IDs, i.e., $(\boldsymbol{u}_{(\delta^{-1})}^\top \boldsymbol{Z}_{\boldsymbol{\mathcal{B}}})_1 > (\boldsymbol{u}_{(\delta^{-1})}^\top \boldsymbol{Z}_{\boldsymbol{\mathcal{B}}})_2 > \cdots > (\boldsymbol{u}_{(\delta^{-1})}^\top \boldsymbol{Z}_{\boldsymbol{\mathcal{B}}})_N$, in other words, $\boldsymbol{z} = \boldsymbol{u}_{(\delta^{-1})}^\top \boldsymbol{Z}_{\boldsymbol{\mathcal{B}}}$. Recall that we consider only the case where all N input tokens are distinct. By redefining the sequence ID of Eq. 35 for $\boldsymbol{\mathcal{B}}$ instead of $\boldsymbol{z}$, sequence IDs in δ^{-d}-base can be rewritten in the δ^{-1}-base system as follows:

$$\mathrm{s}(\boldsymbol{\mathcal{B}}) := \boldsymbol{u}_{(\delta^{-d})}^\top (\boldsymbol{u}_{(\delta^{-1})}^\top \boldsymbol{Z}_{\boldsymbol{\mathcal{B}}}) \tag{37}$$

$$= \sum_{i=1}^{d} \sum_{n=1}^{N} (\boldsymbol{Z}_{\boldsymbol{\mathcal{B}}})_{i,n} \delta^{-\left((N-n)d + (d-i)\right)}. \tag{38}$$

Then, we define *extended sequence IDs* as:

$$\mathrm{s}_{\text{valid}}(\boldsymbol{\mathcal{B}}) := 2\mathrm{s}(\boldsymbol{\mathcal{B}}) - (\boldsymbol{Z}_{\boldsymbol{\mathcal{B}}})_{d,N} \tag{39}$$

and the *dummy sequence IDs* as:

$$\mathrm{s}_{\text{dummy}}^{b}(\boldsymbol{\mathcal{B}}) := \mathrm{s}_{\text{valid}}(\boldsymbol{\mathcal{B}}) + b. \tag{40}$$

for $b \in \{\delta^{-1}, \delta^{-1} + 1, \ldots, 2\delta^{-1} - 1\}$. Then, define each set as follows:

$$\mathcal{S}_{\text{valid}} := \left\{\mathrm{s}_{\text{valid}}(\boldsymbol{\mathcal{B}}) : \boldsymbol{\mathcal{B}} \in \{0, 1, \ldots, \delta^{-1} - 1\}^{d\times N}\right\},$$
$$\mathcal{S}_{\text{dummy}} := \left\{\mathrm{s}_{\text{dummy}}^{b}(\boldsymbol{\mathcal{B}}) : \boldsymbol{\mathcal{B}} \in \{0, 1, \ldots, \delta^{-1} - 1\}^{d\times N},\ b \in \{\delta^{-1}, \delta^{-1} + 1, \ldots, 2\delta^{-1} - 1\}\right\}.$$

Recalling that $(\boldsymbol{Z}_{\boldsymbol{\mathcal{B}}})_{d,N} \in \{0, \ldots, \delta^{-1} - 1\}$, we observe that

$$\mathcal{S}_{\text{valid}} \cap \mathcal{S}_{\text{dummy}} = \emptyset, \tag{41}$$

$$\mathcal{S}_{\text{valid}} \cup \mathcal{S}_{\text{dummy}} = \{0, 1, \ldots, 2\delta^{-Nd} - 1\}. \tag{42}$$

We define the input cubes for each valid sequence ID $s \in \mathcal{S}_{\text{valid}}$ as follows:

$$\boldsymbol{Q}_s := \left\{\mathbf{X} \in [0, 1]^{d\times N} : \boldsymbol{X} \in \boldsymbol{Q}_{\boldsymbol{\mathcal{B}}}, \quad s.t. \quad s = \mathrm{s}'(\boldsymbol{\mathcal{B}})\right\}. \tag{43}$$

Analogous to Eq. 30, we have

$$\max_{s\in\mathcal{S}_{\text{valid}}, \boldsymbol{X}\in\boldsymbol{Q}_s, \boldsymbol{X}'\in\boldsymbol{Q}_{s-1}} \|\boldsymbol{X} - \boldsymbol{X}'\|_2 \leq \begin{cases} \delta\sqrt{Nd} & \text{if } \boldsymbol{\mathcal{B}}_{d,N} \in \{1, 2, \ldots, \delta^{-1} - 1\}, \\ \sqrt{Nd} & \text{if } \boldsymbol{\mathcal{B}}_{d,N} = 0. \end{cases} \tag{44}$$

To implement this, we slightly modified $\mathrm{TF}'^{(2)}$. By Corollary A.7, there exists a Transformer block $\mathrm{TF}^{(2)} : \mathbb{R}^{8\times N} \to \mathbb{R}^{8\times N}$ with two heads, head size $s = 1$, and width size $q = 5$, and two affine linear maps $\mathcal{L}_1^{(2)} : \mathbb{R}^2 \to \mathbb{R}^8$ and $\mathcal{L}_2^{(2)} : \mathbb{R}^8 \to \mathbb{R}$ *s.t.*

$$\boldsymbol{\mathcal{L}}_2^{(2)} \circ \left(\mathrm{TF}^{(2)}\right)^{\circ N} \circ \boldsymbol{\mathcal{L}}_1^{(2)}\left(\begin{bmatrix} \boldsymbol{z}^\top \\ \boldsymbol{Z}_{d,:} \end{bmatrix}\right) = (2\mathrm{s}(\boldsymbol{z}) - \boldsymbol{Z}_{d,N}) \cdot \mathbf{1}_N^\top. \tag{45}$$

Step 3. Token-wise Function Value Mapping. In Steps 1 and 2, the network receives a token ID and extended sequence ID as input for each token, collectively forming a *contextual token ID*. With $2\delta^{-(N+1)d}-1$ loops, the network approximately maps *contextual token IDs* to the output token embeddings of the target function.

To construct contextual token IDs, we define a bijective affine linear mapping $\mathcal{L}_0^{(3)}:\mathbb{N}^2\to\mathbb{N}$ as follows:

$$\mathcal{L}_0^{(3)}(z,s) \coloneqq 2z\delta^{-Nd}+s, \tag{46}$$

where z represents a token ID, defined in Eq. 26, and s represents an sequence ID. Recall that $z\in\{0,1,\dots,\delta^{-d}-1\}$ and sequence IDs are less than $2\delta^{-Nd}$, so informally, it's as if we are adding another digit, z, as the most significant digit in a δ^{-d}-based system. Define the set of contextual token IDs as:

$$\mathcal{K}_{\text{valid}} \coloneqq \left\{\mathcal{L}_0^{(3)}(z,s) : z\in\{0,1,2,\dots,\delta^{-d}-1\}, s\in\mathcal{S}_{\text{valid}}\right\}. \tag{47}$$

and dummy contextual token ID as

$$\mathcal{K}_{\text{dummy}} \coloneqq \left\{\mathcal{L}_0^{(3)}(z,s) : z\in\{0,1,2,\dots,\delta^{-d}-1\}, s\in\mathcal{S}_{\text{dummy}}\right\}. \tag{48}$$

From Eq. 41 and Eq. 42, the following holds:

$$\mathcal{K}_{\text{valid}}\cap\mathcal{K}_{\text{dummy}}=\emptyset, \tag{49}$$

$$\mathcal{K}\coloneqq\mathcal{K}_{\text{valid}}\cup\mathcal{K}_{\text{dummy}}=\{0,1,\dots,2\delta^{-(N+1)d}-1\}. \tag{50}$$

We now define the target output embedding for each ID. Let $\boldsymbol{y}_k\in\mathbb{R}^d$ denote the contextual token embedding corresponding to each contextual token ID, defined as follows:

$$\boldsymbol{y}_k \coloneqq \begin{cases} \bar{f}(\hat{\boldsymbol{X}}_{\boldsymbol{\mathcal{B}}})_{:,n} \quad \textit{s.t.} \quad \mathcal{L}_0^{(3)}\left(\boldsymbol{z}_n, \mathrm{s}'(\boldsymbol{\mathcal{B}})\right)=k & \text{for } k\in\mathcal{K}_{\text{valid}},\\ \mathrm{lin\_interp}\left(\boldsymbol{y}_{\mathrm{near}^-(k,\mathcal{K}_{\text{valid}})}, \boldsymbol{y}_{\mathrm{near}^+(k,\mathcal{K}_{\text{valid}})}, k-\mathrm{near}^-(k,\mathcal{K}_{\text{valid}}), \delta^{-1}\right) & \text{for } k\in\mathcal{K}_{\text{dummy}}, \end{cases} \tag{51}$$

where the nearest functions are defined as

$$\mathrm{near}^+(a,\mathcal{S}) \coloneqq \underset{b\in\mathcal{S},b>a}{\arg\min}\,|a-b|\,, \quad \mathrm{near}^-(a,\mathcal{S}) \coloneqq \underset{b\in\mathcal{S},b<a}{\arg\min}\,|a-b|\,, \tag{52}$$

and a function $\mathrm{lin\_interp}$ is defined by

$$\mathrm{lin\_interp}(a,b,t,n)\coloneqq a+\frac{t}{n}(b-a). \tag{53}$$

The illustration of $\boldsymbol{y}$ is shown in Fig. 2.

Thanks to our design of $\mathcal{K}$, the error between neighboring contextual token embeddings can be bounded as follows. There are two types of error: the variation induced by contextual perturbation and the variation induced by token perturbation, or both. The examples of each pattern are shown in Fig. 2, such that

(1) I <u>draft</u> papers. ; I <u>write</u> papers. (perturbation of token)

(2) He <u>writes</u> papers. ; Mozart <u>writes</u> music. (perturbation of context)

(3) He <u>drinks</u> coffee. ; He <u>drinks</u> coffee. (perturbation of both token and context)

Recall that there are two types of adjacency that generally have small errors, with a few instances causing large errors when 'carryover' occurs, as stated in Eq. 30 and Eq. 44. The key point of our design of $\mathcal{K}$ is that when a large variation occurs, linear interpolation inevitably takes place to smooth out the steep changes between adjacent indices.

Thus for token ID in Eq. 30, if a small variation of token ID, with same context, in input space, $\delta\sqrt{d}$, occurs, the error $e_k = \|\boldsymbol{y}_k - \boldsymbol{y}_{k-1}\|_p$ in the output contextual token embedding can be bounded by the modulus of token continuity as

$$e_k \le \omega_f^{\text{tok}}(\delta\sqrt{d}). \tag{54}$$

In contrast, if a large variation in token input space, $\sqrt{d}$, occurs in the token input space, the error can be bounded using linear interpolation with δ^{-1} intermediate points as:

$$e_k \le \delta\omega_f^{\text{tok}}(\sqrt{d}). \tag{55}$$

The same holds for sequence IDs in Eq. 44. That is, since the variation in context is bounded by sequence variation, the difference in adjacent contextual token IDs caused by perturbations in context is bounded as

$$e_k \le \omega_f^{\text{cont}}(\delta\sqrt{Nd}), \quad \text{or} \quad e_k \le \delta\omega_f^{\text{cont}}(\sqrt{Nd}). \tag{56}$$

for $k = 0, 1, \dots, K-1$.

Since that

$$\omega_f^{\text{cont, tok}}(n \cdot t) \le n \cdot \omega_f^{\text{cont, tok}}(t) \tag{57}$$

for any $n \in \mathbb{N}$ and $t \in [0, \infty)$, with $\delta < 1$, it follows that (note that it holds with opposite inequality due to $\delta < 1$)

$$\delta\omega_f^{\text{cont}}(\sqrt{Nd}) \le \omega_f^{\text{cont}}(\delta\sqrt{Nd}) \quad \text{and} \quad \delta\omega_f^{\text{cont}}(\sqrt{Nd}) \le \omega_f^{\text{cont}}(\delta\sqrt{Nd}). \tag{58}$$

Considering the maximum difference when both token and context perturbations occur, we have, with the triangle inequality,

$$\max_{k' \in \mathcal{K}} \|\boldsymbol{y}_{k'} - \boldsymbol{y}_{k'-1}\|_p \le \delta\left(\omega_f^{\text{tok}}(\sqrt{d}) + \omega_f^{\text{cont}}(\sqrt{Nd})\right) \le \omega_f^{\text{tok}}(\delta\sqrt{d}) + \omega_f^{\text{cont}}(\delta\sqrt{Nd}). \tag{59}$$

Generally, the following inequality holds, for any vector $\boldsymbol{x} \in \mathbb{R}^d$,

$$\max_i |\boldsymbol{x}_i| \le \|\boldsymbol{x}\|_p \le d^{\frac{1}{p}} \max_i |\boldsymbol{x}_i|. \tag{60}$$

Substituting $\boldsymbol{x} = \boldsymbol{y}_k - \boldsymbol{y}_{k-1}$ into the above inequality results in

$$\max_i |(\boldsymbol{y}_k - \boldsymbol{y}_{k-1})_i| \le \|\boldsymbol{y}_k - \boldsymbol{y}_{k-1}\|_p. \tag{61}$$

By Lemma 4.1, there exists a feed-forward layer $\mathrm{FF}^{(3)} : \mathbb{R}^{12d} \to \mathbb{R}^{12d}$ of width size $18d$ and two affine linear maps $\mathcal{L}_1^{(3)} : \mathbb{R} \to \mathbb{R}^{12d}$ and $\mathcal{L}_2^{(3)} : \mathbb{R}^{12d} \to \mathbb{R}^d$ such that

$$|\left(\mathcal{L}_2^{(3)} \circ \left(\mathrm{id} + \mathrm{FF}^{(3)}\right)^{(K-1)} \circ \mathcal{L}_1^{(3)}(k) - \boldsymbol{y}_k\right)_i| \le \max_{k' \in \mathcal{K}} |(\boldsymbol{y}_{k'} - \boldsymbol{y}_{k'-1})_i|, \tag{62}$$

for $i = 1, 2, \dots, d$ and $k = 0, 1, \dots, K-1$, where $K = 2\delta^{-(N+1)d}$.

Let $\tilde{\boldsymbol{y}}_k \in \mathbb{R}^d$ be defined as $\tilde{\boldsymbol{y}}_k \coloneqq \mathcal{L}_2^{(3)} \circ \left(\mathrm{id} + \mathrm{FF}^{(3)}\right)^{(K-1)} \circ \mathcal{L}^{(3)}(k)$. Then we have

$$\begin{aligned} |(\tilde{\boldsymbol{y}}_k - \boldsymbol{y}_k)_i| &\le \max_{k' \in \mathcal{K}} |(\boldsymbol{y}_{k'} - \boldsymbol{y}_{k'-1})_i| & (63)\\ &\le \max_{k' \in \mathcal{K}} \|\boldsymbol{y}_{k'} - \boldsymbol{y}_{k'-1}\|_p \quad \text{because of Eq. 61} & (64)\\ &\le \omega_f^{\text{tok}}(\delta\sqrt{d}) + \omega_f^{\text{cont}}(\delta\sqrt{Nd}) \quad \text{because of Eq. 59}, & (65) \end{aligned}$$

for $i = 1, 2, \dots, d$ and $k = 0, 1, \dots, K-1$.

Concatenated into a Single Transformer Layer Define the input space for each step as:

$$\boldsymbol{X}^{(0)} \in \mathbb{R}^{1\times N}, \quad \boldsymbol{X}^{(1)} \in \mathbb{R}^{5d\times N}, \quad \boldsymbol{X}^{(2)} \in \mathbb{R}^{8\times N}, \quad \boldsymbol{X}^{(3)} \in \mathbb{R}^{12d\times N}, \tag{66}$$

where $\boldsymbol{X}^{(0)}$ act as counter. Define $\mathrm{Attn} : \mathbb{R}^{(17d+9)\times N} \to \mathbb{R}^{(17d+9)\times N}$ with two heads, head size $s = 1$, and width size $q = 5$ via

$$\mathrm{Attn}\left(\begin{bmatrix} \boldsymbol{X}^{(0)} \\ \boldsymbol{X}^{(1)} \\ \boldsymbol{X}^{(2)} \\ \boldsymbol{X}^{(3)} \end{bmatrix}\right) = \begin{bmatrix} \mathbf{0}_{1\times N} \\ \mathbf{0}_{5d\times N} \\ \mathrm{Attn}^{(2)}(\boldsymbol{X}^{(2)}) \\ \mathbf{0}_{12d\times N} \end{bmatrix}, \tag{67}$$

where $\mathrm{Attn}^{(2)}$ denote the self-attention layer of $\mathrm{TF}^{(2)}$ and $\mathbf{0}_{m\times N}$ denote $m \times N$ zero matrix. Let

$$x_0 \in \mathbb{R}, \quad \boldsymbol{x}_1 \in \mathbb{R}^{5d}, \quad \boldsymbol{x}_2 \in \mathbb{R}^{8}, \quad \boldsymbol{x}_3 \in \mathbb{R}^{12d}$$

denote the token-wise input space. Define $\mathrm{FF} : \mathbb{R}^{17d+9} \to \mathbb{R}^{17d+9}$, with the impulse function in Proposition A.4, as:

$$\mathrm{FF}\left(\begin{bmatrix} x_0 \\ \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \boldsymbol{x}_3 \end{bmatrix}\right) = \left(\begin{bmatrix} 1 \\ \mathrm{FF}^{(1)}(\boldsymbol{x}_1) \\ \mathrm{FF}^{(2)}(\boldsymbol{x}_2) + \mathbf{impulse}_{(\delta^{-1}-1)}\big(\mathcal{L}'(\boldsymbol{x}_1), x_0\big) \\ \mathrm{FF}^{(3)}(\boldsymbol{x}_3) + \mathbf{impulse}_{(\delta^{-1}+N)}\big(\mathcal{L}''(\boldsymbol{x}_2), x_0\big) \end{bmatrix}\right) \tag{68}$$

where $\mathrm{FF}^{(2)}$ denotes the feed-forward layer of $\mathrm{TF}^{(2)}$, two linear maps $\mathcal{L}' : \mathbb{R}^{5d} \to \mathbb{R}^{8}$ and $\mathcal{L}'' : \mathbb{R}^{8} \to \mathbb{R}^{12d}$ are defined respectively as follows:

$$\mathcal{L}'(\boldsymbol{x}) \coloneqq \mathcal{L}_1^{(2)}\left(\begin{bmatrix} \mathcal{L}_2^{(1)}(\boldsymbol{x}) \\ \boldsymbol{x}_{5d} \end{bmatrix}\right) \tag{69}$$

$$\mathcal{L}''(\boldsymbol{x}) \coloneqq \mathcal{L}_1^{(3)}\Big(\mathcal{L}_0^{(3)}\big((\boldsymbol{x}_2)_1, \mathcal{L}_2^{(2)}(\boldsymbol{x}_2)\big)\Big) \tag{70}$$

and $\mathbf{impulse}$ refers to the dimension-wise application of $\mathrm{impulse}$ function. Note that x_0 serves the role of a counter.

Each step should always be zero or set to an appropriate initial value at the beginning of the corresponding loop iteration. If the bias term causes it to deviate from this value before the iteration starts, the offset can be subtracted in advance to compensate.

As shown in Proposition A.4, the impulse function requires 2 ReLU functions per dimension and 2 ReLU functions for the corresponding loop iteration. Since the total dimension of $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ is $12d + 8$, this results in an additional width size of $24d + 16 + 4 = 24d + 20$. Since the width size is $7d$ for $\mathrm{FF}^{(1)}$, 5 for $\mathrm{TF}^{(2)}$, and $18d$ for $\mathrm{FF}^{(3)}$, resulting in a total width size of $49d + 25$.

Define two affine linear maps $\mathcal{L}_1 : \mathbb{R}^{d} \to \mathbb{R}^{17d+9}$ and $\mathcal{L}_2 : \mathbb{R}^{17d+9} \to \mathbb{R}^{d}$ via

$$\mathcal{L}_1(\boldsymbol{x}) = \left(0, \mathcal{L}_1^{(1)}(\boldsymbol{x}), \mathbf{0}_{12d+8}{}^{\top}\right)^{\top}, \quad \mathcal{L}_2\big((x_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)^{\top}\big) = \mathcal{L}_2^{(3)}(\boldsymbol{x}_3). \tag{71}$$

Thus, the network, consisting of three steps, is defined as:

$$\tilde{f}(\boldsymbol{X}) \coloneqq \boldsymbol{\mathcal{L}}_2 \circ \mathrm{TF}^{\circ r} \circ \boldsymbol{\mathcal{L}}_1(\boldsymbol{X}) \tag{72}$$

where $r = \delta^{-1} + N + 2\delta^{-(N+1)d}$ and $\mathrm{TF} : \mathbb{R}^{(17d+9)\times N} \to \mathbb{R}^{(17d+9)\times N}$ consists of Attn and FF.

Deriving Approximation Error Generally, the following inequality holds.

$$\sum_{i=1}^{n} x_i^p \le \left(\sum_{i=1}^{n} x_i\right)^p \quad \text{for } x_i \ge 0 \text{ and } p \ge 1. \tag{73}$$

Substituting $x_i = \|\bar{f}(\boldsymbol{X}) - f(\boldsymbol{X})\|_p$ into the above inequality results in

$$\|\bar{f} - f\|_{L^p} = \left(\int \|\bar{f}(\boldsymbol{X}) - f(\boldsymbol{X})\|_p^p \, d\boldsymbol{X}\right)^{1/p} \le \int \|\bar{f}(\boldsymbol{X}) - f(\boldsymbol{X})\|_p d\boldsymbol{X}. \tag{74}$$

Also, generally, the following inequality holds, for $\boldsymbol{x} \in \mathbb{R}^m$,

$$\max_i |\boldsymbol{x}_i| \leq \|\boldsymbol{x}\|_p \leq m^{\frac{1}{p}} \max_i |\boldsymbol{x}_i|. \tag{75}$$

With the triangle inequality, we can bound the approximation error as

$$\begin{aligned}
\|\tilde{f} - f\|_{L^p} &\leq \int \|\tilde{f}(\boldsymbol{X}) - f(\boldsymbol{X})\|_p d\boldsymbol{X} \quad \text{because of Eq. 74} && (76)\\
&\leq \int \|\tilde{f}(\boldsymbol{X}) - \bar{f}(\boldsymbol{X})\|_p d\boldsymbol{X} + \int \|\bar{f}(\boldsymbol{X}) - f(\boldsymbol{X})\|_p d\boldsymbol{X} && (77)\\
&\leq (Nd)^{\frac{1}{p}} \max_{k' \in \mathcal{K}} |\tilde{\boldsymbol{y}}_{k'} - \boldsymbol{y}_{k'}| + \int \|\bar{f}(\boldsymbol{X}) - f(\boldsymbol{X})\|_p d\boldsymbol{X} + \mathcal{O}(N^{2/p}\delta^{d/p}) \quad \text{because of Eq. 75} && (78)\\
&\leq (Nd)^{\frac{1}{p}} \max_{k' \in \mathcal{K}} |\tilde{\boldsymbol{y}}_{k'} - \boldsymbol{y}_{k'}| + \omega_f(\delta\sqrt{Nd}) + \mathcal{O}(N^{2/p}\delta^{d/p}) \quad \text{because of Eq. 22} && (79)\\
&\leq (Nd)^{\frac{1}{p}} \left(\omega_f^{\text{tok}}(\delta\sqrt{d}) + \omega_f^{\text{cont}}(\delta\sqrt{Nd})\right) + \omega_f(\delta\sqrt{Nd}) + \mathcal{O}(N^{2/p}\delta^{d/p}) \quad \text{because of Eq. 65.} && (80)
\end{aligned}$$

Then, δ is expressed in terms of the number of loops as:

$$\begin{aligned}
r = \delta^{-1} + N + 2\delta^{-(N+1)d} &\Leftrightarrow \delta^{-1} + 2\delta^{-(N+1)d} = r - N && (81)\\
&\Rightarrow \delta^{-1} \cdot 2\delta^{-(N+1)d} \geq r - N && (82)\\
&\Leftrightarrow \delta \leq \left(\frac{r-N}{2}\right)^{-1/\left((N+1)d+1\right)}. && (83)
\end{aligned}$$

Thus, we have completed the proof of Theorem A.1. □

A.2. Proof of Theorem 3.6

Extending the construction in Theorem A.1, we then account for the boundedness of the weights and bit complexity.

Proof. Due to the use of bounded weights to approximate discontinuous functions, there inevitably exist regions where quantization errors arise in Step 1 of our construction. We define these regions, for $0 < \epsilon < \delta$, as

$$\Omega([0,1]^{d\times N}, \delta^{-1}, \epsilon) := \left\{ \boldsymbol{X} \in [0,1]^{d\times N} : \exists\, i, n \quad \text{such that} \quad \boldsymbol{X}_{i,n} \in \bigcup_{k=1}^{\delta^{-1}} (k\delta - \epsilon, k\delta) \right\}. \tag{84}$$

That is, Ω consists of all inputs for which at least one coordinate $\boldsymbol{X}_{i,n}$ lies within an ϵ-neighborhood of a quantization discontinuity point $k\delta$ for some $k \in \{1, \ldots, \delta^{-1}\}$.

Outside the region Ω, the quantization function is piecewise constant and can be precisely approximated using bounded weights. By the construction in Lemma A.5, the maximum magnitude of weights required is proportional to $1/\epsilon$. We now estimate the Lebesgue measure of Ω. For each coordinate (i, n), the set

$$\bigcup_{k=1}^{\delta^{-1}} (k\delta - \epsilon, k\delta)$$

is a union of δ^{-1} intervals, each of length ϵ, so the total measure of this set is $\delta^{-1}\epsilon$. Since the input $\boldsymbol{X}$ has $d \times N$ coordinates, and since we consider the event that at least one coordinate lies in a bad region, we apply the union bound for measures to obtain $\mathrm{meas}(\Omega) \leq dN \times \delta^{-1}\epsilon$. Substituting the bound on $\mathrm{meas}(\Omega)$, and using the fact that the maximum magnitude of the weights M satisfies $M = 1/\epsilon$, we have $\mathrm{meas}(\Omega) \leq dN \times \delta^{-1}M^{-1}$. Consequently, the L^p-norm of the quantization error is bounded by $\mathcal{O}\left(\left((M\delta)^{-1}dN\right)^{1/p}\right)$.

When replacing hardmax with softmax, it is required that the error in step 2 remains sufficiently small so that it does not affect step 3. Specifically, a step function is used, in Lemma 4.1 for step 3, to map the index, defined for $\epsilon > 0$ as

$$\mathrm{step}_\epsilon(x) = \begin{cases} 1 & \text{if } x \geq 0, \\ 0 & \text{if } x < 0 - \epsilon, \end{cases} \tag{85}$$

The construction of Lemma 4.1 use this function for indices $k \in \mathcal{K}$ obtained from step 2 as $\mathrm{step}_\epsilon(k-i)$ for $i = 0, 1 \cdots K-1$. Thus, the error in step 2 does not affect step 3 if the perturbed indices, denoted by $\tilde{k}$, satisfies

$$\mathrm{step}_\epsilon(\tilde{k} + 1 - i) = \mathrm{step}_\epsilon(k - i) \quad \text{for all } i = 0, 1 \cdots K-1. \tag{86}$$

To estimate the error introduced by replacing hardmax with softmax in step 2, we revisit the construction of Lemma A.6. Specifically, we extract and reformulate the key components necessary for this estimation. In particular, we consider a simplified form of the attention computation in Eq. 135, denoted by $g_H : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$, which is defined as

$$g_H(\boldsymbol{v}, \boldsymbol{a}) \coloneqq \boldsymbol{v}_{\arg\max_i \boldsymbol{a}_i} \tag{87}$$

When hardmax in Eq. 135 is replaced with softmax, the function can be expressed as

$$g_S(\boldsymbol{v}, \boldsymbol{a}, \lambda) \coloneqq \sum_{i=1}^{N} \frac{\exp(\lambda \boldsymbol{a}_i)}{\sum_{j=1}^N \exp(\lambda \boldsymbol{a}_j)} \boldsymbol{v}_i, \tag{88}$$

where $\lambda > 0$. Note that $\lim_{\lambda\to\infty} g_S(\boldsymbol{v}, \boldsymbol{a}, \lambda) = g_H(\boldsymbol{v}, \boldsymbol{a})$. According to the construction of Lemma A.6, such as Eq. 147, the domains, denoted by $\boldsymbol{v} \in \mathbb{L}_\delta$ and $\boldsymbol{a} \in \mathbb{A}_\delta$, are restricted on

$$\mathbb{L}_\delta \coloneqq \{\boldsymbol{z} \in \{0, 1, \ldots, \delta^{-d} - 1\}^N \mid \boldsymbol{z}_i \neq \boldsymbol{z}_j \text{ for all } i \neq j\}, \tag{89}$$

$$\mathbb{A}_\delta \coloneqq \left\{\boldsymbol{a} \in \mathbb{R}^N \;\middle|\; \boldsymbol{a}_i \in \{0, 1, \ldots, \delta^{-d} - 1\} \text{ or } \boldsymbol{a}_i < 0, \quad \text{if } \boldsymbol{a}_i, \boldsymbol{a}_j \geq 0, \text{ then } \boldsymbol{a}_i \neq \boldsymbol{a}_j \text{ for all } i \neq j, \quad \exists i \text{ s.t. } a_i > 0\right\}. \tag{90}$$

We impose the following two additional assumptions on $\boldsymbol{v} \in \mathbb{L}_\delta$ and $\boldsymbol{a} \in \mathbb{A}_\delta$

$$n \;=\; \arg\max_{1\leq i\leq N} \boldsymbol{a}_i, \qquad v_n \;=\; \max_{1\leq i\leq N} v_i. \tag{91}$$

Under these assumptions, for any finite $\lambda > 0$ we have

$$0 < g_S(\boldsymbol{v}, \boldsymbol{a}, \lambda) < g_H(\boldsymbol{v}, \boldsymbol{a}) \tag{92}$$

and

$$g_S(\boldsymbol{v}, \boldsymbol{a}, \lambda) \geq \frac{\exp(\lambda \boldsymbol{a}_n)}{\sum_{j=1}^N \exp(\lambda \boldsymbol{a}_j)} \boldsymbol{v}_n, \tag{93}$$

Thus we have

$$\begin{aligned}
g_H(\boldsymbol{v}, \boldsymbol{a}) - g_S(\boldsymbol{v}, \boldsymbol{a}, \lambda) &= \boldsymbol{v}_n - g_S(\boldsymbol{v}, \boldsymbol{a}, \lambda) && (94)\\
&\leq \frac{\sum_{j=1}^N \exp(\lambda \boldsymbol{a}_j) - \exp(\lambda \boldsymbol{a}_n)}{\sum_{j=1}^N \exp(\lambda \boldsymbol{a}_j)} \boldsymbol{v}_n && (95)\\
&= \frac{\sum_{j\neq n} \exp(\lambda \boldsymbol{a}_j)}{\sum_{j\neq n} \exp(\lambda \boldsymbol{a}_j) + \exp(\lambda \boldsymbol{a}_n)} \boldsymbol{v}_n && (96)\\
&\leq \frac{\sum_{j\neq n} \exp(\lambda \boldsymbol{a}_j)}{\exp(\lambda \boldsymbol{a}_n)} \boldsymbol{v}_n && (97)\\
&= \boldsymbol{v}_n \sum_{j\neq n} \exp\big(\lambda(\boldsymbol{a}_j - \boldsymbol{a}_n)\big) && (98)\\
&\leq \boldsymbol{v}_n N \exp(-\lambda) \quad \text{because } \boldsymbol{a}_j - \boldsymbol{a}_n \leq -1 && (99)\\
&\leq \delta^{-d} N \exp(-\lambda). && (100)
\end{aligned}$$

Thus, if we aim to bound the error within $\epsilon > 0$, λ must satisfies

$$\delta^{-d} N \exp(-\lambda) < \epsilon \Leftrightarrow \lambda > \log\left(\frac{\delta^{-d} N}{\epsilon}\right). \tag{101}$$

From Eq. 46, the error of contextual ID $k - \tilde{k}$ can be bounded in terms of the error of $\epsilon = g_H(\boldsymbol{v}, \boldsymbol{a}) - g_S(\boldsymbol{v}, \boldsymbol{a}, \lambda)$ as:

$$k - \tilde{k} \leq (\boldsymbol{u}_{(\delta^{-d})})^\top (\epsilon \mathbf{1}_N) \leq \epsilon N \delta^{-(N-1)d} \tag{102}$$

where $\boldsymbol{u}_{(\delta^{-d})} \coloneqq (\delta^{-(N-1)d}, \ldots, \delta^{-d}, 1)^\top$ and $\mathbf{1}_N \in \mathbb{R}^N$ denotes the all-ones vector.

Since Eq. 86 holds if $0 < k - \tilde{k} < 1$, it follows that

$$\lambda > \log\Big((\delta^{-d} N) \cdot (N \delta^{-(N-1)d}) \Big) \Rightarrow \epsilon N \delta^{-(N-1)d} < 1, \tag{103}$$

and Eq. 86 holds. Thus, the bit complexity of λ can be bounded by

$$\mathcal{O}\big(\log\log(\delta^{-Nd} N^2) \big), \tag{104}$$

while ensuring that no error occurs in step 3 when using the softmax function instead of the hardmax function.

The bit complexity at each step of the computation can be analyzed as follows. In Step 1 and Step 2, the bit complexity is bounded by $\mathcal{O}(\log \delta^{-1})$, reflecting the cost of maintaining precision within error δ. In contrast, Step 3 incurs a significantly higher cost, with a bit complexity bounded by $\mathcal{O}(2\delta^{-(N+1)d})$, due to the need to evaluate higher-order terms accurately.

As a result, the overall bit complexity of the Looped Transformer is dominated by Step 3 and can be bounded by

$$\mathcal{O}(\max\{\log\log(\delta^{-Nd} N^2), 2\delta^{-(N+1)d}\}) = \mathcal{O}(\delta^{-(N+1)d}) \qquad (\delta \to 0). \tag{105}$$

With Theorem A.1, the proof of Theorem 3.6 is completed. □

A.3. Approximating Discontinuous Piecewise Functions

We define three utility functions using ReLU activations. Since the target function is discontinuous, there are negligible 'trifling regions' introduced by the bounded weights of the networks.

Proposition A.2 (Rectangular function). *Given $t_1, t_2 \in \mathbb{R}$, there exist four* ReLU *functions that can approximate the following function, denote by* $\mathrm{rect}_t : \mathbb{R} \to \mathbb{R}$, *for* $0 < \epsilon < t_2 - t_1$, *such that:*

$$\mathrm{rect}_{(t_1, t_2, \epsilon)}(x) = \begin{cases} 1 & \text{if } x \in [t_1, t_2 - \epsilon], \\ 0 & \text{if } x \in (-\infty, t_1 - \epsilon] \cup [t_2, \infty), \end{cases} \tag{106}$$

which is represented by

$$\mathrm{rect}_{(t_1, t_2, \epsilon)}(x) = \sigma_R\big(\tfrac{x - t_1 + \epsilon}{\epsilon}\big) - \sigma_R\big(\tfrac{x - t_1}{\epsilon}\big) + \sigma_R\big(\tfrac{-x + t_2}{\epsilon}\big) - \sigma_R\big(\tfrac{-x + t_2 - \epsilon}{\epsilon}\big) - 1. \tag{107}$$

Note that maximum magnitude of the weights is $1/\epsilon$ and 'trifling regions' are $(t - \epsilon, t) \cup (t + 1 - \epsilon, t - 1)$.

Proposition A.3 (Step function). *There exist four* ReLU *functions that can approximate the following function, denote by* $\mathrm{step} : \mathbb{R} \to \mathbb{R}$, *for* $\epsilon > 0$, *such that:*

$$\mathrm{step}_\epsilon(x) = \begin{cases} 1 & \text{if } x \geq 0, \\ 0 & \text{if } x < 0 - \epsilon, \end{cases} \tag{108}$$

which is represented via

$$\mathrm{step}_\epsilon(x) = \sigma_R\big(\tfrac{x}{\epsilon} + 1\big) - \sigma_R\big(\tfrac{x}{\epsilon}\big). \tag{109}$$

Proposition A.4 (Impulse function). *Given $\theta \in \mathbb{N}$, there exist four* ReLU *functions that can approximate the following function, denote by* $\mathrm{impulse}_\theta : \mathbb{R} \times \mathbb{N} \to \mathbb{R}$ *for* $x \in [-M, M]$ *and* $t \in \mathbb{N}$, *such that:*

$$\mathrm{impulse}_\theta(x, t) = \begin{cases} x & \text{if } t = \theta, \\ 0 & \text{otherwise.} \end{cases} \tag{110}$$

which is represented via

$$\begin{aligned} \mathrm{impulse}_\theta(x, t) \coloneqq & \sigma_R\big(x + 2M(t - \theta + 1/2)\big) - 2M\sigma_R(t - \theta + 1/2) \\ & - \sigma_R\big(x + 2M(t - \theta - 1/2)\big) + 2M\sigma_R(t - \theta - 1/2). \end{aligned} \tag{111}$$

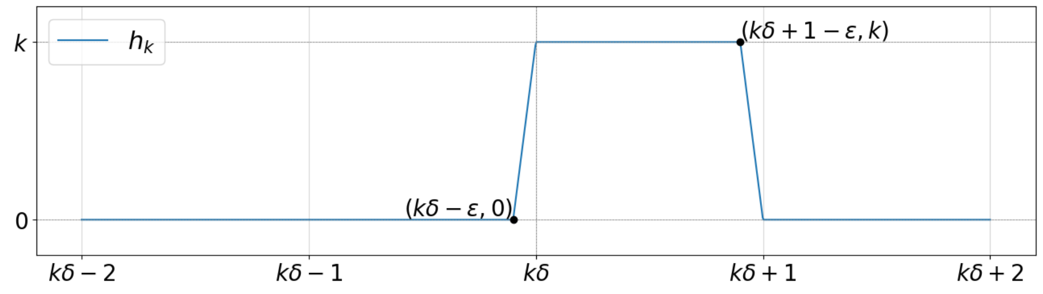


Figure 3. An illustration of $h_k(x)$.

A.4. Step 1. Token-wise Quantization

We aim to construct quantization function $\mathbf{g} : [0,1]^d \to \{0,1,\dots,\delta^{-1}\}^d$, for $\epsilon < \delta$, for each dimension as

$$\mathbf{g}(\boldsymbol{x}) = \left(g(\boldsymbol{x}_1), g(\boldsymbol{x}_2), \dots, g(\boldsymbol{x}_d)\right)^\top, \quad \text{where } g(x) = k \quad \text{if } x \in [k\delta, (k+1)\delta - \epsilon] \text{ for } k = 0, \dots, \delta^{-1} - 1. \tag{112}$$

This function $g : \mathbb{R} \to \mathbb{R}$ can be expressed as

$$g(x) = \sum_{i=0}^{n-1} i \cdot \mathrm{rect}_{(i\delta,(i+1)\delta),\epsilon)}(x) \tag{113}$$

for any $n \in \mathbb{N}$ and $x \in \mathbb{R}$. The illustration of $h_k(x) \coloneqq k \cdot \mathrm{rect}_{(k\delta,(k+1)\delta))}(x)$ is shown in Fig 3. The key idea is that $h_k(x)$ can be represented with a single function h in the form of $h\big(kx, k^2, k\big)$. Lemma A.5 implement $h\big(kx, k^2, k\big)$ with a feed-forward layer and perform the summation through a skip connection.

Lemma A.5. *Given any $\delta^{-1} \in \mathbb{N}$ and $\boldsymbol{x} \in \mathbb{R}^d$, there exists a feed-forward layer $\mathrm{FF} : \mathbb{R}^{5d} \to \mathbb{R}^{5d}$ of width size $q = 7d$ with the maximum magnitude of the weights $1/\epsilon$, and two affine linear maps $\mathcal{L}_1 : \mathbb{R}^d \to \mathbb{R}^{5d}$ and $\mathcal{L}_2 : \mathbb{R}^{5d} \to \mathbb{R}^d$ s.t.*

$$\Big(\mathcal{L}_2 \circ \big(\mathrm{id} + \mathrm{FF}\big)^{\circ(\delta^{-1}-1)} \circ \mathcal{L}_1(\boldsymbol{x})\Big)_i = k \quad \text{if } x \in [k\delta, (k+1)\delta - \epsilon], \quad \text{for } k = 0, \dots, \delta^{-1} - 1 \tag{114}$$

for any $i = 1, 2, \dots, d$.

Proof. On the basis of Proposition A.2, define function $h_k(x) = k \cdot \mathrm{rect}_k(x)$ via

$$\begin{aligned} h_k(x) \coloneqq \sigma_R\big(\tfrac{k}{\epsilon}(x - k\delta + \epsilon)\big) - \sigma_R\big(\tfrac{k}{\epsilon}(x - k\delta)\big) + \sigma_R\big(\tfrac{k}{\epsilon}(-x + k\delta + 1)\big) \\ - \sigma_R\big(\tfrac{k}{\epsilon}(-x + k\delta + 1 - \epsilon)\big) - k, \end{aligned} \tag{115}$$

which satisfies

$$h_k(x) = \begin{cases} k & \text{if } x \in [k\delta, (k+1)\delta - \epsilon], \\ 0 & \text{if } x \in (-\infty, k\delta - \epsilon] \cup [(k+1)\delta, \infty), \end{cases} \tag{116}$$

For any $x \in [k\delta, (k+1)\delta - \epsilon]$ for $k = 0, 1, \dots, \delta^{-1} - 1$, we have

$$\sum_{i=0}^{\delta^{-1}-1} h_i(x) = h_k(x) = k. \tag{117}$$

Define function $h : \mathbb{R}^3 \to \mathbb{R}$ to represent h_k via

$$\begin{aligned} h(kx, k^2, k) \coloneqq \sigma_R\big(\tfrac{kx}{\epsilon} - \tfrac{k^2\delta}{\epsilon} + k\big) - \sigma_R\big(\tfrac{kx}{\epsilon} - \tfrac{k^2\delta}{\epsilon}\big) + \sigma_R\big(-\tfrac{kx}{\epsilon} + \tfrac{k^2\delta}{\epsilon} + \tfrac{k}{\epsilon}\big) \\ - \sigma_R\big(-\tfrac{kx}{\epsilon} + \tfrac{k^2\delta}{\epsilon} + k\tfrac{1-\epsilon}{\epsilon}\big) - \sigma_R(k) = h_k(x). \end{aligned} \tag{118}$$

Define $\boldsymbol{\xi}_k$ as

$$\boldsymbol{\xi}_k = \Big(kx,\ k^2,\ k,\ x,\ \sum_{i=0}^{k-1} h_i(x)\Big)^\top. \tag{119}$$

Then, construct a feed-forward layer $\mathrm{FF} : \mathbb{R}^5 \to \mathbb{R}^5$ with a skip connection such that

$$\begin{aligned}\big(\mathrm{id} + \mathrm{FF}\big)(\boldsymbol{\xi}_k) &= \big(\mathrm{id} + \mathrm{FF}\big)\Big((kx,\ k^2,\ k,\ x,\ \sum_{i=0}^{k-1} h_i(x))^\top\Big) &(120)\\ &= \Big((k+1)x,\ (k+1)^2,\ k+1,\ x,\ \sum_{i=0}^{k} h_i(x)\Big)^\top &(121)\\ &= \boldsymbol{\xi}_{k+1}. &(122)\end{aligned}$$

via

$$\begin{aligned}\big(\mathrm{id} + \mathrm{FF}\big)\left(\begin{bmatrix} kx \\ k^2 \\ k \\ x \\ \sum_{i=0}^{k-1} h_i(x)\end{bmatrix}\right) &= \begin{bmatrix} kx \\ k^2 \\ k \\ x \\ \sum_{i=0}^{k-1} h_i(x)\end{bmatrix} + \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 1 & -1 & -1 \end{bmatrix} \sigma_R & \left(\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ \frac{1}{\epsilon} & -\frac{\delta}{\epsilon} & 1 & 0 & 0 \\ \frac{1}{\epsilon} & -\frac{\delta}{\epsilon} & 0 & 0 & 0 \\ -\frac{1}{\epsilon} & \frac{\delta}{\epsilon} & \frac{1}{\epsilon} & 0 & 0 \\ -\frac{1}{\epsilon} & \frac{\delta}{\epsilon} & \frac{1-\epsilon}{\epsilon} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} kx \\ k^2 \\ k \\ x \\ \sum_{i=0}^{k-1} h_i(x)\end{bmatrix}\right) + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} &(123)\\ &= \begin{bmatrix} kx \\ k^2 \\ k \\ x \\ \sum_{i=0}^{k-1} h_i(x)\end{bmatrix} + \begin{bmatrix} x \\ 2k+1 \\ 1 \\ 0 \\ h_k(x)\end{bmatrix} &(124)\\ &= \begin{bmatrix} kx + x \\ k^2 + 2k + 1 \\ k+1 \\ x \\ \sum_{i=0}^{k-1} h_i(x) + h_k(x)\end{bmatrix} &(125)\\ &= \begin{bmatrix} (k+1)x \\ (k+1)^2 \\ k+1 \\ x \\ \sum_{i=0}^{k} h_i(x)\end{bmatrix}. &(126)\end{aligned}$$

Then, define two affine linear maps $\mathcal{L}_1 : \mathbb{R}^1 \to \mathbb{R}^5$ and $\mathcal{L}_2 : \mathbb{R}^5 \to \mathbb{R}^1$ by

$$\mathcal{L}_1(x) \coloneqq (0, 0, 0, x, 0), \quad \mathcal{L}_2(x_1, x_2, x_3, x_4, x_5) \coloneqq x_5. \tag{127}$$

Thus, we have

$$\begin{aligned}\mathcal{L}_2 \circ \big(\mathrm{id} + \mathrm{FF}\big)^{\circ(\delta^{-1}-1)} \circ \mathcal{L}_1(x) &= \mathcal{L}_2 \circ \big(\mathrm{id} + \mathrm{FF}\big)^{\circ(\delta^{-1}-1)}(\boldsymbol{\xi}_0) &(128)\\ &= \mathcal{L}_2(\boldsymbol{\xi}_{\delta^{-1}}) &(129)\\ &= \sum_{i=0}^{\delta^{-1}-1} h_i(x). &(130)\end{aligned}$$

For d-dimensional inputs, we need d-times more parameters. □

A.5. Step 2. Contextual Mapping

The network takes token IDs as inputs, denoted by $\boldsymbol{z} \in \{0, 1, \dots, \delta^{-d} - 1\}^N$. We only consider cases where all token IDs are distinct. The network maps token IDs into a sequence ID using the inner product with the vector $\boldsymbol{u} \in \mathbb{R}^N$ defined as $\boldsymbol{u} := (\delta^{-(N-1)d}, \delta^{-(N-2)d}, \dots, \delta^{-d}, 1)^\top$ *i.e.*

$$\mathrm{s}(\boldsymbol{z}) := \boldsymbol{u}^\top \boldsymbol{z}. \tag{131}$$

Due to permutation equivariance, we can assume without loss of generality that elements of $\boldsymbol{z} \in \mathbb{L}_\delta$ are ordered, *i.e.*, $\boldsymbol{z}_1 > \boldsymbol{z}_2 > \cdots > \boldsymbol{z}_N$. Then the map s satisfies

$$|\boldsymbol{u}^\top \boldsymbol{z} - \boldsymbol{u}^\top \boldsymbol{z}'| > 1, \quad \text{if } \boldsymbol{z} \neq \boldsymbol{z}'. \tag{132}$$

In other words, s represent $\boldsymbol{z}$ in δ^{-d}-base system. The network computes $\boldsymbol{u}^\top \boldsymbol{z}$ in the form of $\sum_{i=1}^N \delta^{-(N-i)d} \boldsymbol{z}_i$. The network computes $\boldsymbol{s}^{(k)} := \sum_{i=1}^k \delta^{-(k-i)d} \boldsymbol{z}_i$ in each loop, and after N-loops, it outputs $\boldsymbol{s}^{(N)} = \boldsymbol{u}^\top \boldsymbol{z}$. To implement this, the self-attention layer selects $\boldsymbol{z}_k$ in the k-th loop iteration. We design the key, query, and value weights to select the maximum token ID. The feed-forward layer post-processes the token ID in such a way that if it is selected, then it is replaced with a negative value to prevent selection in subsequent iterations, *i.e.*, the post-processed token IDs for the k-th loop are

$$\boldsymbol{z}_i^{(k)} = z \quad \textit{s.t.} \quad \begin{cases} z < 0 & \text{if} \quad i \leq k, \\ z = \boldsymbol{z}_i & \text{otherwise.} \end{cases}$$

We focus on self-attention layers that employ the hardmax function.

Lemma A.6. *Consider the set of distinct indices corresponding to the d-dimensional δ-discretized regions of N tokens, i.e.*

$$\mathbb{L}_\delta := \{\boldsymbol{z} \in \{0, 1, \dots, \delta^{-d} - 1\}^N \mid \boldsymbol{z}_i \neq \boldsymbol{z}_j \textit{ for all } i \neq j\}. \tag{133}$$

There exists a function $\mathrm{s} : \mathbb{R}^N \to \mathbb{R}$ composed of Transformer block $\mathrm{TF} : \mathbb{R}^{5 \times N} \to \mathbb{R}^{5 \times N}$ with the hardmax function, single head, head size $s = 1$, and width size $q = 3$, and two affine linear maps $\mathcal{L}_1 : \mathbb{R} \to \mathbb{R}^5$ and $\mathcal{L}_2 : \mathbb{R}^5 \to \mathbb{R}$, such that

$$\boldsymbol{\mathcal{L}}_2 \circ \mathrm{TF}^{\circ N} \circ \boldsymbol{\mathcal{L}}_1(\boldsymbol{z}^\top) = (\boldsymbol{u}^\top \boldsymbol{z}) \cdot \boldsymbol{1}_N^\top,$$

for any $\boldsymbol{z} \in \mathbb{L}_\delta$, where $\boldsymbol{u} := (\delta^{-(N-1)d}, \delta^{-(N-2)d}, \dots, \delta^{-d}, 1)^\top$.

Proof. Due to permutation equivariance, we can assume without loss of generality that elements of $\boldsymbol{z} \in \mathbb{L}_\delta$ are ordered, *i.e.*, $\boldsymbol{z}_1 > \boldsymbol{z}_2 > \cdots > \boldsymbol{z}_N$. Define $\boldsymbol{u} \in \mathbb{R}^N$ as $\boldsymbol{u} := (\delta^{-(N-1)d}, \dots, \delta^{-d}, 1)^\top$, which satisfy

$$|\boldsymbol{u}^\top \boldsymbol{z} - \boldsymbol{u}^\top \boldsymbol{z}'| > 1, \quad \text{if } \boldsymbol{z} \neq \boldsymbol{z}' \text{ for any } \boldsymbol{z}, \boldsymbol{z}' \in \mathbb{L}_\delta. \tag{134}$$

We construct Transformer block $\mathrm{TF} : \mathbb{R}^{5 \times N} \to \mathbb{R}^{5 \times N}$ with single head and head size $s = 1$ such that, for any $\boldsymbol{z} \in \mathbb{L}_\delta$,

$$\mathrm{TF}^{\circ N}\left(\begin{bmatrix} \boldsymbol{z}^\top \\ \boldsymbol{z}^\top \\ \boldsymbol{1}_N^\top \\ \boldsymbol{0}_N^\top \\ \boldsymbol{0}_N^\top \end{bmatrix}\right) = \begin{bmatrix} \boldsymbol{0}_N^\top \\ \boldsymbol{0}_N^\top \\ \boldsymbol{1}_N^\top \\ \boldsymbol{0}_N^\top \\ (\boldsymbol{u}^\top \boldsymbol{z}) \cdot \boldsymbol{1}_N^\top \end{bmatrix}. \tag{135}$$

where $\boldsymbol{0}_N, \boldsymbol{1}_N \in \mathbb{R}^N$ denote the all-zero and all-one vectors, respectively. For $\boldsymbol{z} \in \mathbb{L}_\delta$, define two series for $k = 0, \cdots N$ as:

$$\boldsymbol{z}_i^{(k)} := z \quad \textit{s.t.} \quad \begin{cases} z < 0 & \text{if} \quad i \leq k, \\ z = \boldsymbol{z}_i & \text{otherwise,} \end{cases} \quad \text{for } i = 1, \dots, N, \quad \in \mathbb{R}^N, \tag{136}$$

$$\boldsymbol{s}^{(k)} := \sum_{i=0}^{k} \delta^{-(k-i)d} \boldsymbol{z}_i \quad \in \mathbb{R}. \tag{137}$$

While $\boldsymbol{z}^{(k)}$ is not uniquely determined, any vector that satisfies the conditions is accepted as $\boldsymbol{z}^{(k)}$. The series $\boldsymbol{s}^{(k)}$ satisfies

$$\begin{aligned} \boldsymbol{s}^{(k+1)} &= \sum_{i=1}^{k+1} \delta^{-(k+1-i)d} \boldsymbol{z}_i && (138) \\ &= \Big(\sum_{i=1}^{k+1-1} \delta^{-(k+1-i)d} \boldsymbol{z}_i \Big) + \boldsymbol{z}_k && (139) \\ &= \Big(\sum_{i=0}^{k} \delta^{-d} \cdot \delta^{-(k-i)d} \boldsymbol{z}_i \Big) + \boldsymbol{z}_{k+1} && (140) \\ &= \delta^{-d} \cdot \boldsymbol{s}^{k} + \boldsymbol{z}_{k+1}, && (141) \end{aligned}$$

for $k = 0, \dots, N-1$. Note that $\boldsymbol{s}^{(N)} = \boldsymbol{u}^\top \boldsymbol{z}$. Define a single-head self-attention $\mathrm{Attn} : \mathbb{R}^{5\times N} \to \mathbb{R}^{5\times N}$ such that

$$\mathrm{Attn}\left(\begin{bmatrix} \boldsymbol{v}^\top \\ \boldsymbol{a}^\top \\ \boldsymbol{1}_N^\top \\ *^\top \\ *^\top \end{bmatrix} \right) = \begin{bmatrix} \boldsymbol{0}_N^\top \\ \boldsymbol{0}_N^\top \\ \boldsymbol{0}_N^\top \\ (\boldsymbol{v}_{\arg\max_j \boldsymbol{a}_j}) \cdot \boldsymbol{1}_N^\top \\ \boldsymbol{0}_N^\top \end{bmatrix}, \tag{142}$$

where $\boldsymbol{v}, \boldsymbol{a} \in \mathbb{R}^N$, and $* \in \mathbb{R}^N$ denotes arbitrary vectors, via the weight parameters

$$\boldsymbol{W}_O = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \boldsymbol{W}_V = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{W}_K = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{W}_Q = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{143}$$

Define $\mathrm{FF} : \mathbb{R}^5 \to \mathbb{R}^5$ of width size $q = 3$ via:

$$\begin{aligned} \mathrm{FF}\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \right) &= \begin{bmatrix} 0 & 0 & 0 \\ -M & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & \delta^{-d} - 1 \end{bmatrix} \sigma_R \left(\begin{bmatrix} 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} 0 \\ \epsilon \\ 0 \\ 0 \\ 0 \end{bmatrix} \right) && (144) \\ &= \begin{bmatrix} 0 \\ -M\sigma_R(x_2 - x_4 + \epsilon) \\ 0 \\ -\sigma_R(x_4) \\ (\delta^{-d} - 1)\sigma_R(x_5) + \sigma_R(x_4) \end{bmatrix}, && (145) \end{aligned}$$

where $0 < \epsilon < 1$ and $M > \frac{\delta^{-d}-1}{\epsilon}$.

For $x_2 \in \{0, 1, \dots, \delta^{-d} - 1\} \cup \{x \mid x \leq 0\}$ and $x_4 \in \{0, 1, \dots, \delta^{-d} - 1\}$ with $x_4 \geq x_2$, we have

$$x_2 - M\sigma_R(x_2 - x_4 + \epsilon) = z, \quad \textit{s.t.} \quad \begin{cases} z = x_1 & \text{if} \quad x_4 > x_2, \\ z < 0 & \text{if} \quad x_4 = x_2. \end{cases} \tag{146}$$

This post-processes the token ID in such a way that if it is selected, then it is replaced with a negative value *i.e.*

$$\begin{aligned} \boldsymbol{z}_i^{(k)} - M\sigma_R(\boldsymbol{z}_i^{(k)} - \boldsymbol{z}_k + \epsilon) = z \quad \textit{s.t.} \quad & \begin{cases} z < 0 & \text{if} \quad i \leq k+1, \\ z = \boldsymbol{z}_i & \text{otherwise}, \end{cases} \\ = \boldsymbol{z}_i^{(k+1)} \quad & \text{for } i = 1, \dots, N. \end{aligned} \tag{147}$$

We confirm that the Transformer block $\mathrm{TF} : \mathbb{R}^{5\times N} \to \mathbb{R}^{5\times N}$, composed of Attn and FF, satisfies, for $k = 0, \dots, N-1$,

$$\mathrm{TF}\left(\begin{bmatrix} \boldsymbol{z}^\top \\ (\boldsymbol{z}^{(k)})^\top \\ \mathbf{1}_n^\top \\ \mathbf{0}_n^\top \\ (\boldsymbol{s}^{(k)})^\top \end{bmatrix}\right) = (\mathrm{id} + \mathbf{FF}) \circ (\mathrm{id} + \mathrm{Attn}) \left(\begin{bmatrix} \boldsymbol{z}^\top \\ (\boldsymbol{z}^{(k)})^\top \\ \mathbf{1}_n^\top \\ \mathbf{0}_n^\top \\ (\boldsymbol{s}^{(k)})^\top \end{bmatrix}\right) \tag{148}$$

$$= (\mathrm{id} + \mathbf{FF}) \left(\begin{bmatrix} \boldsymbol{z}^\top \\ (\boldsymbol{z}^{(k)})^\top \\ \mathbf{1}_n^\top \\ \boldsymbol{z}_{k+1} \cdot \mathbf{1}_N^\top \\ (\boldsymbol{s}^{(k)})^\top \end{bmatrix}\right) \tag{149}$$

$$= \begin{bmatrix} \boldsymbol{z}^\top \\ (\boldsymbol{z}^{(k)})^\top \\ \mathbf{1}_n^\top \\ \boldsymbol{z}_{k+1} \cdot \mathbf{1}_N^\top \\ (\boldsymbol{s}^{(k)})\mathbf{1}_N^\top \end{bmatrix} + \begin{bmatrix} \mathbf{0}_n^\top \\ -M\boldsymbol{\sigma}_R\big((\boldsymbol{z}^{(k)})^\top - \boldsymbol{z}_k \cdot \mathbf{1}_N^\top + \epsilon \mathbf{1}_N^\top\big) \\ \mathbf{0}_n^\top \\ -\boldsymbol{z}_{k+1} \cdot \mathbf{1}_N^\top \\ (\delta^{-d} - 1)(\boldsymbol{s}^{(k)})\mathbf{1}_N^\top + \boldsymbol{\sigma}_R(\boldsymbol{z}_{k+1}\mathbf{1}_N^\top) \end{bmatrix} \tag{150}$$

$$= \begin{bmatrix} \boldsymbol{z}^\top \\ (\boldsymbol{z}^{(k+1)})^\top \\ \mathbf{1}_n^\top \\ \mathbf{0}_n^\top \\ \delta^{-d}(\boldsymbol{s}^{(k)})\mathbf{1}_N^\top + \boldsymbol{z}_{k+1}\mathbf{1}_N^\top \end{bmatrix} \quad \text{because of Eq. 147} \tag{151}$$

$$= \begin{bmatrix} \boldsymbol{z}^\top \\ (\boldsymbol{z}^{(k+1)})^\top \\ \mathbf{1}_n^\top \\ \mathbf{0}_n^\top \\ (\boldsymbol{s}^{(k+1)})\mathbf{1}_N^\top \end{bmatrix} \quad \text{because of Eq. 141.} \tag{152}$$

Define two affine linear maps $\mathcal{L}_1 : \mathbb{R} \to \mathbb{R}^5$ and $\mathcal{L}_2 : \mathbb{R}^5 \to \mathbb{R}$ via $\mathcal{L}_1(x) \coloneqq (x, x, 1, 0, 0)$ and $\mathcal{L}_2(x_1, x_2, x_3, x_4, x_5) \coloneqq x_5$ respectively. Thus, we have

$$\boldsymbol{\mathcal{L}}_2 \circ \mathrm{TF}^{\circ N} \circ \boldsymbol{\mathcal{L}}_1(\boldsymbol{z}^\top) = (\boldsymbol{u}^\top \boldsymbol{z}) \cdot \mathbf{1}_N^\top.$$

□

Combining the sequence ID with token ID, the network computes contextual mapping.

Corollary A.7. *There exists a Transformer block* $\mathrm{TF}_2 : \mathbb{R}^{8\times N} \to \mathbb{R}^{8\times N}$ *with the hardmax function, two heads, head size* $s = 1$*, and width size* $q = 5$*, and two affine linear maps* $\mathcal{L}_1 : \mathbb{R}^2 \to \mathbb{R}^8$ *and* $\mathcal{L}_2 : \mathbb{R}^8 \to \mathbb{R}$ *s.t.*

$$\boldsymbol{\mathcal{L}}_2 \circ \mathrm{TF}^{(2)^{\circ N}} \circ \boldsymbol{\mathcal{L}}_1 \left(\begin{bmatrix} \boldsymbol{z}^\top \\ \boldsymbol{Z}_{d,:} \end{bmatrix}\right) = \big(2\boldsymbol{u}^\top \boldsymbol{z} - \boldsymbol{Z}_{d,N}\big) \cdot \mathbf{1}_N^\top \quad \text{for any } \boldsymbol{z} \in \mathbb{L}_\delta.$$

Proof. Define a self-attention $\mathrm{Attn} : \mathbb{R}^{8\times N} \to \mathbb{R}^{8\times N}$ such that

$$\mathrm{Attn}\left(\begin{bmatrix} \boldsymbol{v}^\top \\ \boldsymbol{a}^\top \\ \mathbf{1}_N^\top \\ *^\top \\ *^\top \\ \boldsymbol{Z}_{d,:} \\ *^\top \\ *^\top \end{bmatrix}\right) = \begin{bmatrix} \mathbf{0}_N^\top \\ \mathbf{0}_N^\top \\ \mathbf{0}_N^\top \\ (\boldsymbol{v}_{\arg\max_j \boldsymbol{a}_j}) \cdot \mathbf{1}_N^\top \\ \mathbf{0}_N^\top \\ \mathbf{0}_N^\top \\ \boldsymbol{Z}_{d,N} \cdot \mathbf{1}_N^\top \\ \mathbf{0}_N^\top \end{bmatrix}, \tag{153}$$

where $\boldsymbol{v}, \boldsymbol{a} \in \mathbb{R}^N$, and $* \in \mathbb{R}^N$ denotes arbitrary vectors, via the weight parameters

$$\boldsymbol{W}_O^1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{W}_V^1 = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}, \quad \boldsymbol{W}_K^1 = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \end{bmatrix}, \quad \boldsymbol{W}_Q^1 = \begin{bmatrix} 0 & 0 & 1 & 0 & \dots & 0 \end{bmatrix} \tag{154}$$

and

$$\boldsymbol{W}_O^2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \boldsymbol{W}_V^2, \boldsymbol{W}_K^2 = \begin{bmatrix} 0 & \dots & 0 & 1 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{W}_Q^2 = \begin{bmatrix} 0 & 0 & 1 & 0 & \dots & 0 \end{bmatrix}, \tag{155}$$

where ... denotes a sequence of zeros.

Define $\mathrm{FF} : \mathbb{R}^8 \to \mathbb{R}^8$ of width size $q = 5$ via:

$$\mathrm{FF}\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}\right) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -M & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & \delta^{-d}-1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix} \sigma_R\left(\begin{bmatrix} 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} + \begin{bmatrix} 0 \\ \epsilon \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}\right) \tag{156}$$

$$= \begin{bmatrix} 0 \\ -M\sigma_R(x_2 - x_4 + \epsilon) \\ 0 \\ -\sigma_R(x_4) \\ (\delta^{-d}-1)\sigma_R(x_5) + \sigma_R(x_4) \\ 0 \\ 0 \\ -\sigma_R(x_7) \\ \sigma_R(x_7) - \sigma_R(x_8) \end{bmatrix}, \tag{157}$$

where $0 < \epsilon < 1$ and $M > \frac{\delta^{-d}-1}{\epsilon}$. Then, we define two affine linear maps $\mathcal{L}_1 : \mathbb{R}^2 \to \mathbb{R}^8$ and $\mathcal{L}_2 : \mathbb{R}^8 \to \mathbb{R}$ respectively:

$$\mathcal{L}_1(x_1, x_2) := (x_1, x_1, 1, 0, 0, x_2, 0, 0), \quad \mathcal{L}_2(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) := 2x_5 - x_6 \tag{158}$$

From Lemma A.6, the corollary holds for this construction. □

A.6. Step 3. Function Value Mapping with Bit Extraction

We employ a bit extraction technique (Bartlett et al., 1998) , as used (Zhang et al., 2023) for weight-tied ReLU networks, to approximately memorize the label set. Given $K \in \mathbb{N}$ input indices $k = 1, 2, \dots, K$ with associated values $y_1, y_2, \dots, y_K \in \mathbb{R}$, the network approximately memorizes the differences $y_i - y_{i-1}$ using their base-2 representation. Since the binary representation is limited to $\{0,1\}$, the differences $y_i - y_{i-1}$ must be rescaled by a factor $\epsilon \coloneqq \max_i |y_i - y_{i-1}|$ as

$$a_i = \lfloor \frac{y_i}{\epsilon} \rfloor, \tag{159}$$

where $\lfloor x \rfloor \coloneqq \max\{n : n \leq x,\ n \in \mathbb{Z}\}$. Then, the difference $b_i = a_i - a_{i-1}$ satisfies $b_i \in \{-1, 0, 1\}$ and can be represented using two binary values $c_i, d_i \in \{0,1\}$ as follows:

$$b_i = c_i - d_i, \tag{160}$$

and we have

$$a_k = a_0 + \sum_{i=0}^{k} b_i = a_0 + \sum_{i=0}^{k} c_i - \sum_{i=0}^{k} d_i \quad \text{for} \quad k = 0, 1, \dots, n-1. \tag{161}$$

Lemma A.9 and Lemma 4.1 show that $\sum_{i=0}^{k} c_i$ and $\sum_{i=0}^{k} d_i$ can be realized by composition of single feed-forward layer. Thus, the network can approximate y_i using ϵa_i, denoted as $\tilde{y}_i$, with the following accuracy:

$$|\tilde{y}_i - y_i| = |\epsilon \lfloor \frac{y_i}{\epsilon} \rfloor - \epsilon \frac{y_i}{\epsilon}| = \epsilon |\lfloor \frac{y_i}{\epsilon} \rfloor - \frac{y_i}{\epsilon}| \leq \epsilon. \tag{162}$$

For a d-dimensional input-output pair, we construct the networks for each dimension *i.e*.

$$\tilde{\boldsymbol{y}} = (\tilde{y}^1, \tilde{y}^2, \dots, \tilde{y}^d) \tag{163}$$

The basic strategy of our lemma and proof follows Lemma D.1 from Zhang et al. (2023), as shown below and Proposition 3.2. However, their result cannot be directly applied here, as it requires depth-2 networks.

Proposition A.8 (Lemma D.1 in Zhang et al. (2023))**.** *Given any* $r \in \mathbb{N}^+$*, there exists a function* $\mathrm{FF} : \mathbb{R}^{3d} \to \mathbb{R}^{3d}$ *with width* 8 *and depth* 2*, utilizing two affine linear maps* $\mathcal{L}_1 : \mathbb{R}^2 \to \mathbb{R}^5$ *and* $\mathcal{L}_2 : \mathbb{R}^5 \to \mathbb{R}$*, such that for any* $\theta_1, \theta_2, \dots, \theta_r \in \{0,1\}$*, the following holds:*

$$\mathcal{L}_2 \circ \mathrm{FF}^{\circ r} \circ \mathcal{L}_1\big(k,\ \mathrm{bin}\, 0.\theta_1\theta_2\cdots\theta_r\big) = \sum_{\ell=1}^{k} \theta_\ell \quad \text{for } k = 1, 2, \dots, r, \tag{164}$$

where $\mathrm{bin}\, 0.\theta_1\theta_2\cdots\theta_r$ *denote the binary representation of* $\theta = \sum_{l=1}^{r} \theta_l 2^{-l}$*.*

We found that the *loop unrolling* technique allows us to reduce the number of layers from 2 to 1 by replacing $x^{k+1} = \mathrm{ReLU}(\mathrm{ReLU}(x'^k))$ with $(x^{k+1}, x'^k) = \mathrm{ReLU}(x'^k, x^k)$. Although our method makes the weights dependent on $\theta_1, \theta_2, \dots, \theta_r$, this does not present an issue for our construction in function approximation. Specifically, $\theta_1, \theta_2, \dots, \theta_r$ is fixed for each target function, and the role of the network is to learn the weights tailored to that single function.

Lemma A.9. *Given* $\theta_1, \theta_2, \dots, \theta_r \in \{0,1\}$ *for some* $r \in \mathbb{N}^+$*, there exists a feed-forward layer* $\mathrm{FF} : \mathbb{R}^6 \to \mathbb{R}^6$ *of width size* 9 *and two affine linear maps* $\mathcal{L}_1 : \mathbb{R} \to \mathbb{R}^6$ *and* $\mathcal{L}_2 : \mathbb{R}^6 \to \mathbb{R}$ *s.t.*

$$\mathcal{L}_2 \circ (\mathrm{id} + \mathrm{FF})^{\circ r} \circ \mathcal{L}_1\big(k\big) = \sum_{l=1}^{k} \theta_l \quad \text{for } k = 1, 2, \dots, r, \tag{165}$$

where the bit complexity is bounded by $\mathcal{O}(r)$*.*

Proof. From Proposition A.3, we have a function $\mathrm{step}_\epsilon(x)$, for $\epsilon > 0$, defined by

$$\mathrm{step}_\epsilon(x) = \sigma_R\big(\tfrac{x}{\epsilon} + 1\big) - \sigma_R\big(\tfrac{x}{\epsilon}\big), \tag{166}$$

as shown in Figure 4, and it satisfies

$$\mathrm{step}_\epsilon(x) = \begin{cases} 1 & \text{if } x \geq 0, \\ 0 & \text{if } x \leq 0 - \epsilon. \end{cases} \tag{167}$$

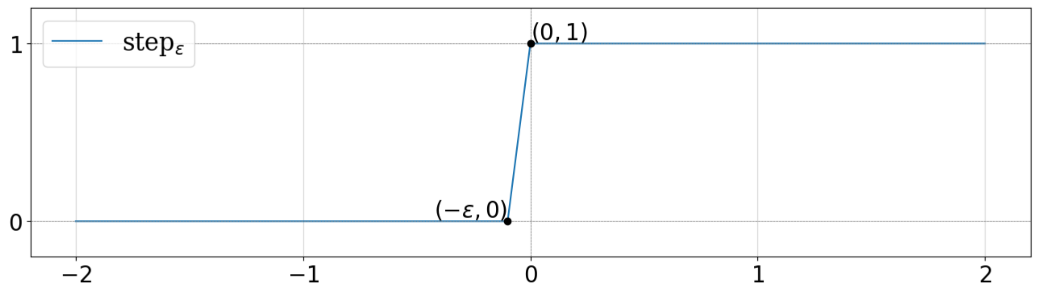


Figure 4. An illustration of $\mathrm{step}_\epsilon(x)$.

Define β_l for $l = 0, 1, \dots, r$ as

$$\beta_l = \text{bin}\, 0.\theta_l \cdots \theta_r, \tag{168}$$

where bin $0.\theta_l \cdots \theta_r$ denote the binary representation of $\theta = \sum_{i=l}^r \theta_i 2^{-i}$ and $\beta_0 := 0$. If we set $\epsilon < 2^{-r}$, it follows that

$$\begin{aligned} \theta_l &= \mathrm{step}_\epsilon(\text{bin}\, 0.\theta_l \cdots \theta_r - \tfrac{1}{2}) & (169)\\ &= \mathrm{step}_\epsilon(\beta_l - \tfrac{1}{2}), & (170)\end{aligned}$$

implying, for $l = 0, 1, \dots, r-1$,

$$\begin{aligned} \beta_{l+1} &= 2\beta_l - \theta_l & (171)\\ &= 2\beta_l - \mathrm{step}_\epsilon\big(\beta_l - \tfrac{1}{2}\big). & (172)\end{aligned}$$

For all $l = 1, \dots, r$, since the product xy satisfies

$$\begin{aligned} xy &= \max\{0, x+y-1\} & (173)\\ &= \sigma_R(x+y-1), & (174)\end{aligned}$$

for $x, y \in \{0,1\}$, it follows that

$$\begin{aligned} \sum_{l=1}^k \theta_l &= \sum_{l=1}^k \theta_l + \sum_{l=k+1}^r 0 & (175)\\ &= \sum_{l=1}^r \theta_l \cdot \mathrm{step}_\epsilon(k-l) & (176)\\ &= \sum_{l=1}^r \sigma_R\Big(\theta_l + \mathrm{step}_\epsilon(k-l) - 1\Big) & (177)\\ &= \sum_{l=1}^r \sigma_R\Big(\mathrm{step}_\epsilon(\beta_l - \tfrac{1}{2}) + \mathrm{step}_\epsilon(k-l) - 1\Big). & (178)\end{aligned}$$

To compute the right-hand side, we require two nested ReLU functions. By employing loop unrolling, we precompute $\mathcal{T}(\beta_l - \frac{1}{2})$ and $\mathcal{T}(k-l)$ in the previous iterations, reducing the requirement to a single layer.

Define $\boldsymbol{\xi}_l$ for $l = 0, 1, \dots, r-1$ as

$$\begin{aligned} \boldsymbol{\xi}_l &= \Big(k-l,\ \beta_l,\ \beta_{l+1},\ \mathrm{step}_\epsilon(\beta_l - \frac{1}{2}),\ \mathrm{step}_\epsilon(k-l),\ \mathrm{sum}(l)\Big)^\top, \\ \text{where} \quad \mathrm{sum}(l) &:= \sum_{i=1}^l \sigma_R\Big(\mathrm{step}_\epsilon(\beta_i - \tfrac{1}{2}) + \mathrm{step}_\epsilon(k-i) - 1\Big). \end{aligned} \tag{179}$$

Note that we have β_{l+1} in the l-th loop to precompute $\mathrm{step}_\epsilon(\beta_{l+1} - \frac{1}{2})$ and $\mathrm{step}_\epsilon\big((k-(l+1)\big)$ for the $(l+1)$-th loop.

Define $\mathrm{FF} : \mathbb{R}^6 \to \mathbb{R}^6$ with a width size of 9 such that

$$\big(\mathrm{id} + \mathrm{FF}\big)(\boldsymbol{\xi}_l) = \big(\mathrm{id} + \mathrm{FF}\big)\left(\begin{bmatrix} k-l \\ \beta_l \\ \beta_{l+1} \\ \mathrm{step}_\epsilon(\beta_l - \frac{1}{2}) \\ \mathrm{step}_\epsilon(k-l) \\ \mathrm{sum}(l) \end{bmatrix}\right) \tag{180}$$

$$= \begin{bmatrix} k-l \\ \beta_l \\ \beta_{l+1} \\ \mathrm{step}_\epsilon(\beta_l - \frac{1}{2}) \\ \mathrm{step}_\epsilon(k-l) \\ \mathrm{sum}(l) \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\sigma_R\left(\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1/\epsilon & 0 & 0 & 0 \\ 0 & 0 & 1/\epsilon & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1/\epsilon & 0 & 0 & 0 & 0 & 0 \\ 1/\epsilon & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} k-l \\ \beta_l \\ \beta_{l+1} \\ \mathrm{step}_\epsilon(\beta_l - \frac{1}{2}) \\ \mathrm{step}_\epsilon(k-l) \\ \mathrm{sum}(l) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1/(2\epsilon)+1 \\ -1/(2\epsilon) \\ 0 \\ -1/\epsilon+1 \\ -1/\epsilon \\ -1 \end{bmatrix}\right) + \begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{181}$$

$$= \begin{bmatrix} k-l \\ \beta_l \\ \beta_{l+1} \\ \mathrm{step}_\epsilon(\beta_l - \frac{1}{2}) \\ \mathrm{step}_\epsilon(k-l) \\ \mathrm{sum}(l) \end{bmatrix} + \begin{bmatrix} -1 \\ \sigma_R(\beta_l) - \sigma_R\big(\mathrm{step}_\epsilon(\beta_l - \frac{1}{2})\big) \\ \sigma_R(\beta_{l+1}) - \big(\sigma_R(\frac{\beta_{l+1}-1/2}{\epsilon}+1) - \sigma_R(\frac{\beta_{l+1}-1/2}{\epsilon})\big) \\ -\sigma_R\big(\mathrm{step}_\epsilon(\beta_l - \frac{1}{2})\big) + \sigma_R(\frac{\beta_{l+1}-1/2}{\epsilon}+1) - \sigma_R(\frac{\beta_{l+1}-1/2}{\epsilon}) \\ -\sigma_R\big(\mathrm{step}_\epsilon(k-l)\big) + \sigma_R(\frac{k-(l+1)}{\epsilon}+1) - \sigma_R(\frac{k-(l+1)}{\epsilon}) \\ \sigma_R\Big(\mathrm{step}_\epsilon(k-l) + \mathrm{step}_\epsilon(\beta_l - \frac{1}{2}) - 1\Big) \end{bmatrix} \tag{182}$$

$$= \begin{bmatrix} k-(l+1) \\ 2\beta_l - \mathrm{step}_\epsilon(\beta_l - \frac{1}{2}) \\ 2\beta_{l+1} - \mathrm{step}_\epsilon(\beta_{l+1} - \frac{1}{2}) \\ \mathrm{step}_\epsilon(\beta_{l+1} - \frac{1}{2}) \\ \mathrm{step}_\epsilon\big((k-(l+1)\big) \\ \mathrm{sum}(l+1) \end{bmatrix} = \begin{bmatrix} k-(l+1) \\ \beta_{l+1} \\ \beta_{l+2} \\ \mathrm{step}_\epsilon(\beta_{l+1} - \frac{1}{2}) \\ \mathrm{step}_\epsilon\big((k-(l+1)\big) \\ \mathrm{sum}(l+1) \end{bmatrix} = \boldsymbol{\xi}_{l+1}, \tag{183}$$

Define $\mathcal{L}_1 : \mathbb{R} \to \mathbb{R}^6$ and $\mathcal{L}_2 : \mathbb{R}^6 \to \mathbb{R}$ via

$$\mathcal{L}_1(k) := (k, \beta_0, \beta_1, 0, 0, 0)^\top = \boldsymbol{\xi}_0, \quad \mathcal{L}_2(x_1, x_2, x_3, x_4, x_5, x_6) := x_6, \tag{184}$$

respectively. The lemma holds for this construction. $\square$

Then, we prove Lemma 4.1 with Lemma A.9.

Lemma 4.1. *Given $\boldsymbol{y}_k \in \mathbb{R}^d$ for $k = 0, 1, \ldots, K-1$ with*

$$|(\boldsymbol{y}_k - \boldsymbol{y}_{k-1})_i| \leq \varepsilon_i \quad \textit{for } i = 1, \ldots, d,$$

there exists a feed-forward layer $\mathrm{FF} : \mathbb{R}^{12d} \to \mathbb{R}^{12d}$ with a width size of $18d$, and two affine linear maps $\mathcal{L}_1 : \mathbb{R} \to \mathbb{R}^{12d}$ and $\mathcal{L}_2 : \mathbb{R}^{12d} \to \mathbb{R}^d$ s.t.

$$\left|\big(\mathcal{L}_2 \circ (\mathrm{id} + \mathrm{FF})^{\circ(K-1)} \circ \mathcal{L}_1(k) - \boldsymbol{y}_k\big)_i\right| \leq \varepsilon_i, \tag{18}$$

for $i = 1, \ldots, d$ and $k = 0, 1, \ldots, K-1$.

Proof. We prove this for the case where $d = 1$, considering $y_k \in \mathbb{R}$ for $k = 0, 1, \dots, K-1$. Define

$$a_k = \left\lfloor \tfrac{y_k}{\varepsilon} \right\rfloor \quad \text{for } k = 0, 1, \dots, K-1, \tag{185}$$

where $\lfloor x \rfloor = \max\{n : n \le x,\ n \in \mathbb{Z}\}$ and set

$$b_k = a_k - a_{k-1} \quad \text{for } k = 1, 2, \dots, K-1. \tag{186}$$

Since $b_k \in \{-1, 0, 1\}$, there exist $c_k \in \{0, 1\}$ and $d_k \in \{0, 1\}$ such that

$$b_k = c_k - d_k \quad \text{for } k = 1, 2, \dots, K-1. \tag{187}$$

Thus, we have

$$a_k = a_0 + \sum_{i=1}^{k} c_i - \sum_{i=1}^{k} d_i \quad \text{for any } k \in \{1, 2, \dots, K-1\} \tag{188}$$

From Lemma A.9, there exist $\mathrm{FF}^{(c)}, \mathrm{FF}^{(d)} : \mathbb{R}^6 \to \mathbb{R}^6$ of width size 9 and affine linear maps $\mathcal{L}_2' : \mathbb{R}^6 \to \mathbb{R}$ and $\mathcal{L}_1^{(c)}, \mathcal{L}_1^{(d)} : \mathbb{R} \to \mathbb{R}^6$ *s.t.*

$$\mathcal{L}_2' \circ (\mathrm{id} + \mathrm{FF}^{(c)})^{\circ(m-1)} \circ \mathcal{L}_1^{(c)}(k) = \sum_{i=1}^{k} c_i, \quad \mathcal{L}_2' \circ (\mathrm{id} + \mathrm{FF}^{(d)})^{\circ(m-1)} \circ \mathcal{L}_1^{(d)}(k) = \sum_{i=1}^{k} d_i, \tag{189}$$

for $k = 0, 1, \dots, K-1$. Then, define $\mathrm{FF} : \mathbb{R}^{12} \to \mathbb{R}^{12}$ of width size 18, for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^6$,

$$\mathrm{FF}\left(\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}\right) := \begin{bmatrix} \mathrm{FF}^{(c)}(\boldsymbol{x}) \\ \mathrm{FF}^{(d)}(\boldsymbol{y}) \end{bmatrix}. \tag{190}$$

Define $\mathcal{L}_1 : \mathbb{R} \to \mathbb{R}^{12}$ and $\mathcal{L}_2 : \mathbb{R}^{12} \to \mathbb{R}$ as

$$\mathcal{L}_1(x) := \begin{bmatrix} \mathcal{L}_1^{(c)}(x) \\ \mathcal{L}_1^{(d)}(x) \end{bmatrix}, \quad \mathcal{L}_2\left(\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}\right) := \epsilon\big(a_0 + \mathcal{L}_2'(\boldsymbol{x}) - \mathcal{L}_2'(\boldsymbol{y})\big). \tag{191}$$

We can confirm that

$$\begin{aligned}
&\mathcal{L}_2 \circ (\mathrm{id} + \mathrm{FF})^{\circ(K-1)} \circ \mathcal{L}_1(k) && (192)\\
&= \mathcal{L}_2 \circ (\mathrm{id} + \mathrm{FF})^{\circ(K-1)} \left(\begin{bmatrix} \mathcal{L}_1^{(c)}(k) \\ \mathcal{L}_1^{(d)}(k) \end{bmatrix}\right) && (193)\\
&= \mathcal{L}_2 \left(\begin{bmatrix} (\mathrm{id} + \mathrm{FF}^{(c)})^{\circ(K-1)} \circ \mathcal{L}_1^{(c)}(k) \\ (\mathrm{id} + \mathrm{FF}^{(d)})^{\circ(K-1)} \circ \mathcal{L}_1^{(d)}(k) \end{bmatrix}\right) && (194)\\
&= \epsilon\big(a_0 + \sum_{i=1}^{k} c_i - \sum_{i=1}^{k} d_i\big) = \epsilon a_k. && (195)
\end{aligned}$$

Thus we have

$$|\mathcal{L}_2 \circ (\mathrm{id} + \mathrm{FF})^{\circ(K-1)} \circ \mathcal{L}_1(k) - y_k| = |\epsilon a_k - y_k| \le \varepsilon. \tag{196}$$

For d-dimensional inputs, we need d-times more parameters. □

B. Role of Time-dependent Scaling Parameters

We show that time-dependent scaling parameters overcome the limitations inherent to the looped architecture and eliminate the dependence of the modulus of continuity. We use the architecture defined in Section 4 as:

$$\mathrm{FF}(\boldsymbol{x}) \to \boldsymbol{\eta}(t) \odot \mathrm{FF}(\boldsymbol{x}) \quad \text{for the } t\text{-th loops}, \tag{197}$$

The following lemma demonstrates that time-dependent scaling parameters can exactly map indices to output vectors.

Theorem 4.2. *Given $\boldsymbol{y}_k \in \mathbb{R}^d$ for $k = 0, 1, \ldots, K-1$, there exists a feed-forward layer $\mathrm{FF} : \mathbb{R}^{4d} \to \mathbb{R}^{4d}$ with a width size of $6d$, $\boldsymbol{\eta}(t) \in \mathbb{R}^{4d}$ for $t = 1, \ldots, K-1$, and two affine linear maps $\mathcal{L}_1 : \mathbb{R} \to \mathbb{R}^{4d}$ and $\mathcal{L}_2 : \mathbb{R}^{4d} \to \mathbb{R}^d$ s.t.*

$$|\big(\mathcal{L}_2 \circ (\mathrm{id} + \boldsymbol{\eta} \odot \mathrm{FF})^{\circ(K-1)} \circ \mathcal{L}_1(k) - \boldsymbol{y}_k\big)_i| = 0,$$

for $i = 1, \ldots, d$ and $k = 0, 1, \ldots, K-1$.

Proof. We consider the case when $d = 1$, where $y_k \in \mathbb{R}$ for $k = 0, 1, \ldots, K-1$. We update y_k as follows:

$$y_k \to y_k + \epsilon, \tag{198}$$

where ϵ is chosen such that none of the y_l values are zero.

Next, we define $\eta(l) \in \mathbb{R}^4$ as:

$$\eta(l) \coloneqq (0, 1, \frac{y_l}{y_{l-1}} - 1, \frac{y_l}{y_{l-1}})^\top \quad \text{for } l = 1, 2, \ldots, K-1. \tag{199}$$

By Proposition A.4, we have, $x \in [-M, M]$ and $t \in \mathbb{N}$,

$$\begin{aligned}\mathrm{impulse}_0(x,t) &= \sigma_R\big(x + 2M(t+1/2)\big) - 2M\sigma_R(t+1/2) \\ &\quad - \sigma_R\big(x + 2M(t-1/2)\big) + 2M\sigma_R(t-1/2) \qquad (200)\\ &= \begin{cases} x & \text{if } t = 0, \\ 0 & \text{otherwise,}\end{cases} \qquad (201)\end{aligned}$$

where $M > \max_{k \in \{0,1,\ldots,K-1\}} y_k$.

Let $k \in \{0, 1, \ldots, K-1\}$ be the input index that specifies which y_k to extract. Define

$$s(l) \coloneqq \sum_{i=0}^{l} \mathrm{impulse}_0\big(y_i, k-i\big), \tag{202}$$

for $l = 0, 1, \ldots, K-1$, which satisfies

$$s(K-1) = y_k. \tag{203}$$

Define $\boldsymbol{\xi}_l \in \mathbb{R}^4$ via

$$\boldsymbol{\xi}_l \coloneqq \Big(k,\ k-l-1,\ y_l,\ s(l)\big)\Big)^\top. \tag{204}$$

for $l = 0, 1, \ldots, K-1$.

Then, define FF : $\mathbb{R}^4 \to \mathbb{R}^4$ of width size $q = 6$ via:

$$(\mathrm{id} + \eta(l) \odot \mathrm{FF})(\boldsymbol{\xi}_{l-1}) = \boldsymbol{\xi}_{l-1} +$$
$$\eta(l) \odot \left(\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & -2M & 2M \end{bmatrix} \sigma_R \left(\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 2M & 1 & 0 \\ 0 & 2M & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \boldsymbol{\xi}_{l-1} + \begin{bmatrix} 0 \\ -1 \\ M \\ -M \\ 1/2 \\ -1/2 \end{bmatrix} \right) + \begin{bmatrix} 0 \\ -1 \\ 0 \\ 0 \end{bmatrix} \right) \tag{205}$$

$$= \boldsymbol{\xi}_{l-1} + \begin{bmatrix} 0 \\ 1 \\ \frac{y_l}{y_{l-1}} - 1 \\ \frac{y_l}{y_{l-1}} \end{bmatrix} \odot \begin{bmatrix} 0 \\ -1 \\ \sigma_R(y_{l-1}) - \sigma_R(-y_{l-1}) \\ \Big(\sigma_R\big(y_{l-1} + 2M((k-l) + 1/2)\big) \\ -2M\sigma_R((k-l)+1/2) \\ -\sigma_R\big(y_{l-1} + 2M(k-l-1/2)\big) + 2M\sigma_R(k-l-1/2) \Big) \end{bmatrix} \tag{206}$$

$$= \begin{bmatrix} k \\ k-l \\ y_{l-1} \\ s(l-1) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ \frac{y_l}{y_{l-1}} - 1 \\ \frac{y_l}{y_{l-1}} \end{bmatrix} \odot \begin{bmatrix} 0 \\ -1 \\ y_{l-1} \\ \mathrm{impulse}_0\big(y_{(l-1)}, k-l)\big) \end{bmatrix} \tag{207}$$

$$= \begin{bmatrix} k \\ k-l-1 \\ y_l \\ s(l) \end{bmatrix} = \boldsymbol{\xi}_l. \tag{208}$$

for $l = 1, 2, \ldots, K-1$. Thus we have

$$(\mathrm{id} + \eta(K-1) \odot \mathrm{FF}) \circ \cdots \circ (\mathrm{id} + \eta(1) \odot \mathrm{FF})(\boldsymbol{\xi}_0) = \boldsymbol{\xi}_{K-1} \tag{209}$$

Then, define two affine linear maps $\mathcal{L}_1 : \mathbb{R} \to \mathbb{R}^4$ and $\mathcal{L}_2 : \mathbb{R}^4 \to \mathbb{R}$ by

$$\mathcal{L}_1(x) := (k, k, y_0, 0), \quad \mathcal{L}_2(x_1, x_2, x_3, x_4) := x_4 - \epsilon. \tag{210}$$

We can extend this to d-dimensional input by using d times more parameters, by applying the above to each dimension □

C. Details of Experiments

This appendix section provides additional details on the experiments for each task.

C.1. Reasoning Tasks

C.1.1. Problem Settings

Longest Common Subsequence (LCS) is the longest common to a given set of sequences. We use problems with input lengths of 60 and 100. Two sequences are sampled uniformly from the alphabet.

Edit Distance (ED) problem, also known as Levenshtein distance, is to find the minimum cost required to change one sequence into the other. We adopted the problem setting and data generation approach from Feng et al. (2023), but applied larger input lengths. The costs for insertion, deletion, and replacement were set to 2, 2, and 3, respectively. They generate instances of the edit distance problem as shown in Algorithm 1. The first string is randomly selected, while the second is generated in two ways: (1) a random string yielding a large edit distance, and (2) a corrupted copy of the first string, resulting in a small edit distance.

Algorithm 1 ED Data Generation from Feng et al. (2023)

```
1: Input: Length of the First String $n$
2: Input: Alphabet $V = \{a, b...z\}$
3: Output: Sequence $s_1$, $s_2$ Sample $t$ uniformly from $\{3, 4...10\}$   $T \leftarrow$ Sample $t$ letters from $V$   $s_1 \leftarrow$ Sample $n$ letters uniformly from $T$   Sample $p$ uniformly from $[0, 1]$
4: if $p < 0.4$ then
5:     Sample $l$ uniformly from $\{n-3, n-2, ..., n+2\}$
6:     $s_2 \leftarrow$ Sample $l$ letters uniformly from $T$
7: else
8:     while $len(s_2)$ not in $[n-3, n+2]$ do
9:         $s_2 \leftarrow s_1$
10:        for $i \leftarrow 1$ to $n$ do
11:            Sample $p$ uniformly from $\{0, 1...len(s_2) - 1\}$
12:            Sample $l$ uniformly from $T$
13:            Randomly conduct one of the followings: pop $s_2[p]$, substitute $s_2[p]$ with $l$, insert $l$ into $s_2[p]$
14:        end for
15:    end while
16: end if
```

Sudoku We use the dataset from Radcliffe (2020), which contains over 3 million Sudoku puzzles. The puzzles are flattened, with 0 representing blank grids. The input sequence is formatted as:

```
100503700603008090000009800010000000876100000000006000000000007080907604700060312.
```

Countdown To generate the dataset, we randomly sampled the target and input numbers for each instance. Pairs that have no solution were excluded. For tokenization, we assigned unique tokens to each number and symbol. The target sequence is represented as:

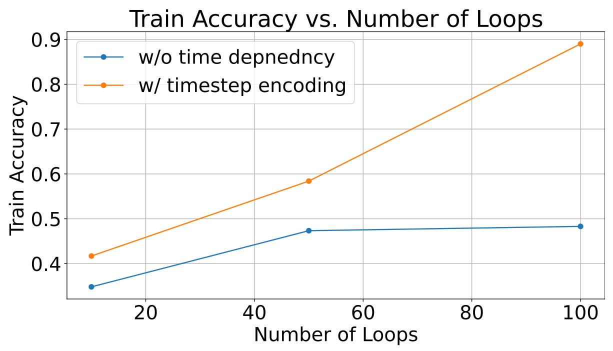
```
58 84 48 62 96 62 - 58 = 4 48 / 4 = 12 84 + 12 = 96 .
```

The model learns to predict the target sequence from the input:

```
58 84 48 62 96 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 .
```

C.1.2. Training and Test Accuracy Correlation for Edit Distance

Given that our study focuses on function approximation capabilities, one might question whether it is appropriate to rely on test evaluations, which are influenced by generalization. Here, we confirm that there is a strong correlation between training and test results, validating this approach. Figure 5 demonstrates a strong positive correlation between training and test accuracy, enabling the evaluation of approximation power through test accuracy.



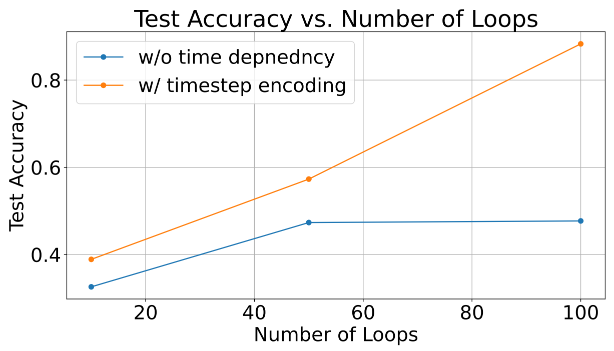


Figure 5. Training and test accuracy for the edit distance task with a sequence length of 60.

C.1.3. Model and Training Configuration

We used Looped Transformers with 4 attention heads and a 256-dimensional embedding. The AdamW optimizer (Loshchilov & Hutter, 2018) was used with $\beta_1 = 0.9$, $\beta_2 = 0.999$, a weight decay of 0.01, and a linear learning rate decay scheduler starting at $\mathrm{lr} = 10^{-4}$ and ending at 0, with 5 warm-up steps. Training consisted of 50 epochs for reasoning, 200K steps for in-context learning, and 100K iterations for language modeling, using a batch size of 64. For time-dependent models, $\boldsymbol{\gamma}(t)$ and $\boldsymbol{\alpha}(t)$ were initialized as zero and one vectors, respectively, following Peebles & Xie (2023). The input embeddings are added at each loop iteration. Furthermore, for Sudoku and in-context learning tasks, the output of each intermediate loop is incorporated into the loss as (Yang et al., 2023).

C.2. In-Context Learning

We followed the setting of Yang et al. (2024). The problem is to learn the function class from a given sequence composed of the pairs of input $\boldsymbol{x}_i$ and output values $f(\boldsymbol{x}_i)$. The input for model is $(\boldsymbol{x}_1, f(\boldsymbol{x}_1), \ldots, \boldsymbol{x}_k, f(\boldsymbol{x}_k), \boldsymbol{x}_{\text{test}})$, and model learns to predict $f(\boldsymbol{x}_{\text{test}})$. The model is trained on $f(\boldsymbol{x}_k)$ and its performance is evaluated on $f(\boldsymbol{x}_{\text{test}})$ using the squared error.

We use depth-4 decision trees with 20-dimensional inputs. Each function in this class is represented by a full binary tree with 16 leaf nodes. Non-leaf nodes are associated with specific input coordinates, while leaf nodes are assigned target values. To evaluate f($\boldsymbol{x}$), the tree is traversed from the root, moving to the right if the coordinate value is positive and to the left otherwise. Inputs and leaf node values are sampled from N(0,$\boldsymbol{I}$), and the coordinates for non-leaf nodes are drawn uniformly at random. Our training setup follows the approach of Yang et al. (2024). Following the curriculum training approach of Garg et al. (2022); Yang et al. (2024), we progressively increase the task dimensionality from 5 to 20 in steps of 1 every 5000 steps, while the sequence length increases from 26 to 101 in increments of 5 over the same interval.

C.3. Language Modeling

Tokenization is performed using byte-pair encoding, following GPT-2 (Radford et al., 2019). The Transformer model is based on the GPT-2 decoder architecture (Radford et al., 2019). The baseline standard Transformer model consists of 6 layers, 8 attention heads, and an embedding size of 512. The Looped Transformer has a 1 layer, 12 attention heads, and a hidden dimension of 768, which were chosen to match the parameter size of the baseline. We initialize $\boldsymbol{\gamma}(t)$ as zero vectors and $\boldsymbol{\alpha}(t)$ as one vector for time-dependent models. The AdamW optimizer (Loshchilov & Hutter, 2018) is used with $\beta_1 = 0.9$, $\beta_2 = 0.95$, a weight decay of 1×10^{-4}, and a learning rate schedule with 2000 warmup steps. The maximum learning rate is set to 2×10^{-4} and decays to 6×10^{-5} using a cosine schedule. Training is conducted for 100k iterations with a batch size of 48 and a block size of 1024.

D. Disccusion

Multiple Layers A natural question is whether our analysis, which focuses on single-layer Looped Transformers, can be extended to multi-layer architectures. We restricted our analysis to a single layer in order to highlight a key strength of Looped Transformers—namely, their universality as function approximators even with just one layer. A more specific question is whether multi-layer Looped Transformers can overcome potential limitations inherent to the single-layer design. While it is conceivable that deeper architectures but with fixed-depth feedforward layers may achieve better approximation accuracy, this remains an open question. The difficulty lies in the fact that such improvements are not captured in terms of asymptotic order but rather in constants, which are harder to analyze theoretically. For instance, if we allow a logarithmic number of layers depending on the desired approximation precision, then even our current construction may overcome the limitations of the looped architecture. However, this deviates from our main objective, which is to characterize the approximation rate solely in terms of the number of loops.

Additional Experiments To assess the model's sensitivity to input continuity, we designed a perturbed version of the WikiText-103 dataset, where 10% of the tokens were randomly replaced. We trained models with and without timestep encoding and evaluated both their memorization performance and continuity behavior. Continuity was measured by applying small perturbations to the input and quantifying the change in output embeddings. The model with timestep encoding showed improved memorization (cross-entropy loss reduced from 4.32 to 4.18) and a significant reduction in continuity coefficients (from 130.6 to 21.5). These results suggest that timestep encoding not only enhances stability under perturbations but also enables more faithful input-output mappings, thereby improving both robustness and learning efficiency.