On the Limitations and Capabilities of Position Embeddings for Length Generalization

Yang Chen, Yitao Liang, and Zhouchen Lin, *Fellow, IEEE*

***Abstract*—In Transformers, Position Embeddings (PEs) significantly influence Length Generalization (LG) performance, yet their fundamental role remains unclear. In this work, we identify two fundamental limitations of PEs in achieving LG: the inability to acquire new operators beyond training data, and the inability to deal with inconsistent computational roles of the same positions across scales. To formalize these limitations, we introduce *computational representation complexity* which measures the number of distinct unit operators required for a task, and the notion of *canonicality*, which characterizes operator-position consistency as the scale varies. We prove that PEs cannot achieve LG for mappings whose computational representation complexity increases with scale, nor for mappings that are non-canonical. We further show that PEs enable LG when these two failure modes are absent. In this case, LG can be achieved with a PE whose positional relation function *characterizes* the computational representation. To enhance LG for non-canonical mappings, we introduce *scale hint*, allowing flexible instance scaling. We also propose a *learning-based position embedding* framework that automatically learns positional relations. Our work provides theoretical insights and practical strategies for improving LG in Transformers.**

***Index Terms*—Length generalization, position embedding, transformer, reasoning.**

I. Introduction

LENGTH Generalization (LG) refers to the ability of a model to extrapolate from small-scale instances to larger ones in reasoning [1], [2], [3], [4]. In many tasks, the sample space grows exponentially with the problem scale, making exhaustive training infeasible. Thus, it is important to learn from limited training samples at small scales while generalizing to larger ones. Furthermore, learning to solve complex tasks from simple ones is a significant ability of human learning. LG is an essential aspect when building a model of human-level reasoning capability [5], [6], [7].

Received 10 July 2025; revised 18 March 2026; accepted 25 April 2026. Recommended for acceptance by R. Giryes. *(Corresponding authors: Zhouchen Lin; Yitao Liang.)*

Yang Chen is with the State Key Laboratory of General Artificial Intelligence, School of Intelligence Science and Technology, Peking University, Beijing 100871, China (e-mail: yangchen@pku.edu.cn).

Yitao Liang is with the Institute for Artificial Intelligence, Peking University, Beijing 100871, China, and also with the Beijing Institute for General Artificial Intelligence, Beijing 648408, China (e-mail: yitaol@pku.edu.cn).

Zhouchen Lin is with the State Key Laboratory of General Artificial Intelligence, School of Intelligence Science and Technology, Peking University, Beijing 100871, China, also with the Institute for Artificial Intelligence, Peking University, Beijing 100871, China, and also with the Pazhou Laboratory (Huangpu), Guangzhou 510555, China (e-mail: zlin@pku.edu.cn).

Digital Object Identifier 10.1109/TPAMI.2026.3688702

In general, LG is inherently difficult because the training data do not provide information on how to compute results for unseen large-scale instances. No single algorithm can guarantee length generalization across all tasks [8] (see Appendix A, (available online) for a further illustration). As a result, incorporating prior knowledge about the target concept is crucial for designing effective models. Current works have adopted various techniques to encode such prior knowledge. In Transformers, Position Embeddings (PEs) are found to play a significant role in LG performance [3], [9]. However, few works have theoretically investigated why and how PEs enable LG [4]. Moreover, certain tasks fail to generalize with existing PEs [2], [10], [11]. It is unclear whether this is due to suboptimal PE strategies or fundamental PE limitations. A fundamental question arises naturally:

What are the limitations and capabilities of PEs for LG?

A PE encodes positional relations between elements in a sequence. Intuitively, these relations define how the model interprets the positional structure of a sequence, specifying how positions interact and influence computations, enabling the model to distinguish between different positions and determine positional dependencies. The relations are determined by a Positional Relation Function (PRF), denoted as $\phi(i, j)$, which maps a query position i and a key position j to a value that represents their relationship. For example, in Relative Position Embedding (RPE) [12], the function is $\phi(i, j) = i - j$; in Absolute Position Embedding (APE) [13], [14], the PRF can be seen as $\phi(i, j) = i * K + j$ for some constant K such that $\phi(i_1, j_1) \neq \phi(i_2, j_2)$ for any $0 \leq j_1 \leq i_1 \leq N - 1$, $0 \leq j_2 \leq i_2 \leq N - 1, (i_1, j_1) \neq (i_2, j_2)$, where N is the maximum length considered. Some PEs have very different implementations but they share the same PRF and thus capture the same position relations. For instance, while learnable RPE [12] and RoPE [15] implement positional relations with learnable vectors and rotary matrix respectively, they have the same PRF $\phi(i, j) = i - j$. See Appendix B, (available online) for a more detailed illustration. This work focuses on the role of positional relations in LG and will mainly discuss the impact of PRFs.

A. Our Works

Limitations of PEs for LG: In Section IV, we identify two fundamental failure modes of PEs in achieving LG. In the first case (Example 6), solving larger-scale instances requires additional unit operators beyond those seen in training, so the training data provide no information about these operators, and

no PRF can encode the mapping consistently across scales. In the second case (Example 7), arising in autoregressive models, even with a fixed operator set, the same position may correspond to different operators across scales, preventing a fixed PRF from consistently identifying computational roles and leading to LG failure.

To formalize the two modes, we introduce the notions of Computational Representation (CR) and canonicality. CR provides a unified framework for describing how a mapping is computed and how positional relations correspond to computational roles. Canonicality characterizes whether the computation at the same position is invariant across scales.

We further define *Computational Representation Complexity* (CRC) to quantify the number of distinct unit operators required for a task. Based on this notion, we formalize the above limitations of PEs for LG in Section V-A. Theorem 1 shows that when CRC increases from the training scale to the testing scale, adapting PEs alone is insufficient for LG in general, capturing the first failure mode. Theorem 2 further shows that even when CRC remains invariant, the scale-invariant nature of PEs can prevent LG if the mapping lacks canonicality, capturing the second failure mode.

Capabilities of PEs for LG: In Section V-B, we study the complementary side of the above limitations. Theorem 3 shows that when a mapping admits a canonical CR with non-increasing CRC, LG can be achieved with a proper PE, or more specifically, the PE whose PRF characterizes the CR (Definition 6). This result indicates that CRC and canonicality together characterize the boundary of PE-based LG. In particular, the capability of PEs lies in aligning operators and identifying the computational roles of positions consistently across different scales. We also empirically validate the insight for the capabilities of PEs for LG in Section VI.

Scale Hint Technique: While increasing CRC and non-canonicality both limit LG with PEs, they are fundamentally different. Increasing CRC reflects an inherent scaling challenge, whereas non-canonicality arises from how positional roles are encoded and can be mitigated by modifying the data format. From the perspective of PEs, the latter stems from the scale-invariant nature of PRFs, which may be insufficient to consistently characterize computations across different scales.

To address this issue, we introduce the *Scale Hint* (SH) technique (Section VII-A), which augments the PRF with the instance scale as an additional input. This allows the PRF to vary across scales, requiring only a consistent structure within each scale rather than across all scales. As a result, SH enables more flexible data representations and reduces unnecessary computational overhead.

For example, in an n-digit Addition instance, it suffices to align each addend to length n, instead of padding to a global maximum length. The corresponding PRF takes the form

$$\phi(i,j,n) = K\lfloor\frac{i-j}{n}\rfloor + \min\left((i-j) \bmod n, K-1\right),$$

where i and j denote the query and key positions, n is the scale hint, and K is a constant.

Learning-Based Position Embeddings: In practice, it would be unrealistic to manually redesign the PRF task by task. It is desirable to use a single model across different tasks. To address this problem, we propose learning-based PE (Section VII-B), respectively, where the PRF ϕ is learned automatically. More concretely, we replace the handcrafted PRF $\phi(i,j)$ ($\phi(i,j,n)$ if scale hint is employed) with a learnable one $\phi_\theta(i,j)$ ($\phi_\theta(i,j,n)$, respectively). Empirically, we observe that the learning-based PEs achieve length generalization across a variety of tasks, eliminating the need for task-specific designs. This approach shows the potential to use one learnable PE to handle diverse tasks that would otherwise require different handcrafted designs.

II. Related Work

Length Generalization in Reasoning Tasks: The literature on length generalization can be broadly categorized into two strands. The first focuses on *processing extremely long sequences*, often referred to as *long-context modeling* in the context of LLMs [16], [17], [18], [19], [20]. This line of research primarily addresses the challenges of capturing long-range dependencies and mitigating the substantial memory and computational demands associated with long inputs.

The second line of work, to which the present study contributes, investigates *generalization from shorter training sequences to longer sequences at inference time* [4], [10]. The central question is how models can generalize by learning from length-limited data that provide only incomplete information in training. Addressing this question necessitates the introduction of suitable inductive biases [8], [21], which may be incorporated through data formatting [2], [22], architectural modifications [12], [23], or training strategies [24], [25]. This work specifically examines the role of PEs, a structural component of Transformers, as a means of facilitating LG.

Role of Position Embeddings in Transformers: PEs encode positional information that is otherwise absent in the standard Transformer architecture. They are critical for enabling Transformers to model sequences in a position-sensitive manner and have been shown to significantly enhance the model's expressive capacity. In particular, it has been demonstrated that Transformers are Turing-complete when equipped with APE, but not without them [26].

Recent empirical studies have highlighted the importance of PEs in length generalization [3], [9]. Replacing APEs with PEs that encode relative position relations has been shown to improve performance on tasks requiring extrapolation to longer sequences [12], [15]. Moreover, more sophisticated PEs that encode structured or hierarchical positional information can lead to further gains [27], [28], [29]. Despite these promising empirical findings, a comprehensive theoretical understanding of the role that PEs play in enabling length generalization remains limited. This work takes a step towards such understanding by providing a systematic theoretical investigation into how modifications to PEs influence generalization behavior across sequence lengths.

III. Preliminary

A. Notations

We use $[N]$ to denote the set $\{1,\ldots,N\}$. The cardinality of a set A is denoted by $|A|$. $\Sigma^{[N]}$ denotes the set of all strings over

Σ of length at most N, i.e., $\Sigma^{[N]} = \bigcup_{k\in[N]} \Sigma^k$. Let Σ^* denote the Kleene closure of the alphabet Σ, i.e., $\Sigma^* = \bigcup_{k=1}^{\infty} \Sigma^k$. We use $\mathbb{1}(\cdot)$ to denote the indicator function. We use $\mathcal{P}(\mathcal{S})$ to denote the power set of the set $\mathcal{S}$.

B. Length Generalization

We first formalize the notion of length generalization studied in this paper. The following definition specifies the training-testing setting in which a model is trained on smaller-scale instances and is required to generalize to larger ones.

Definition 1 ((N_0, N)-Length Generalization): Suppose that in an instance $x = c_1, \ldots, c_n \in \Sigma$ of length n, each element c_i is sampled i.i.d. from some distribution $\mathcal{P}$ on Σ, where Σ is the element domain and the support set of the distribution $\mathcal{P}$ is Σ, i.e., $\mathrm{supp}(\mathcal{P}) = \Sigma$. A learning algorithm $\mathscr{A}$ achieves (N_0, N)-Length Generalization ((N_0, N)-LG) for the concept f^* if there exists a distribution $\mathcal{P}_{N_0}$ on $[N_0]$ such that the model f learned by $\mathscr{A}$ on a sufficiently large $\mathcal{X}_{N_0} = \{x_k\}$, which is generated by $(n, x) \sim \mathcal{P}_{N_0}(n)\mathcal{P}(x \mid n)$ satisfies $f(x) = f^*(x)$ for all $x \in \Sigma^n$, $n = 1, \ldots, N$.

Definition 1 highlights the particular difficulty of LG compared with standard learning settings. It captures two core challenges in length generalization: 1) the exponential growth of the input space as the scale increases, and 2) the absence of information in shorter instances about components unique to longer inputs. Consequently, successful LG requires more than fitting the training distribution; it requires an inductive bias that transfers computation from small to larger scales. We study PEs as a key source of such bias and characterize their limitations and capabilities for LG.

C. Computational Representation

To analyze when LG is possible with PEs in autoregressive generative models like Transformers, we need a way to describe not only the input-output of a mapping, but also the procedure by which it is computed sequentially for instances of different scales. To this end, we introduce the notion of a computational representation, which formalizes a sequential computation in terms of unit operators and their dependencies.

Definition 2 (Computational Representation): Consider a sequence-to-sequence mapping $f : \Sigma^* \mapsto \Sigma^*$. For an input sequence $x = (x_1, \ldots, x_n) \in \Sigma^n$, the output is $f(x) = (y_1, \ldots, y_{K(n)}) \in \Sigma^{K(n)}$, where $K : \mathbb{N} \mapsto \mathbb{N}$ specifies the output length. Let $z = (z_1, \ldots, z_{n+K(n)})$ denote the concatenation of the input and output sequences, where

$$z_i = \begin{cases} x_i, & 1 \le i \le n, \\ y_{i-n}, & n+1 \le i \le n+K(n). \end{cases}$$

A **Computational Representation (CR)** $\mathcal{C}_N = \{C_n\}_{n\in[N]}$ of f up to input length N is a collection of sets of tuples

$$C_n = \{(i, g^{(i)}, I_i) \mid 1 \le i \le n + K(n)\},$$

where each triple $(i, g^{(i)}, I_i)$ specifies how the element z_i is computed. Here $g^{(i)}$ is an operator applied to a set of parent elements indexed by

$$I_i = (i_1, \ldots, i_{r_i}) \in [i-1]^{r_i}.$$

For input elements, we define $g^{(i)} = g_0$, representing an input operator, and set $I_i = \emptyset$ $(1 \le i \le n)$.

For output elements, the computation is defined by

$$z_i = g^{(i)}(z_{i_1}, \ldots, z_{i_{r_i}}), \qquad n+1 \le i \le n+K(n),$$

where the operator $g^{(i)}$ belongs to a finite set

$$\mathcal{G}(\mathcal{C}_N) = \{g_1, \ldots, g_R\}.$$

Each operator in $\mathcal{G}(\mathcal{C}_N)$ is a function $g_r : \Sigma^{d_r} \mapsto \Sigma$ with arity $d_r \le D$. We refer to each $g \in \mathcal{G}(\mathcal{C}_N)$ as a *unit operator*.

For notational simplicity, the input tuples $(i, g_0, \emptyset)$ for $1 \le i \le n$ may be omitted from C_n when no ambiguity arises. We may also omit the subscripts N and $n \in [N]$ when the maximum input length is clear from context.

Remark 1: For a mapping f such that $f(x) \in \Sigma^{K(n)}$ for $x \in \Sigma^n$ up to scale, we may assume without loss of generality that the PRF set is $\Phi_{N+K(N)}$, where

$$\Phi_m := \{\phi \mid \phi : [m] \times [m] \mapsto [m]\}.$$

Furthermore, to simplify notation, we treat $K(n)$ as a constant K. Our analysis extends directly to the general case where $K(n)$ varies with n.

The CR describes how the concatenated sequence z is computed using a finite set of unit operators, specifying the dependency structure and the operator applied at each position. The following examples illustrate how common sequence mappings can be expressed in this form.

Example 1 (CR of Parity (with CoT)): Consider the Parity (with CoT) task $f(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$ where $\Sigma = \{0, 1\}$, $y_1 = x_1$ and $y_i = x_i \oplus y_{i-1}$ for all $2 \le i \le n$. A CR $\mathcal{C} = \{C_n\}$ of this task can be given as:

$$C_n = \{(1, g_0, \emptyset), \ldots, (n, g_0, \emptyset), (n+1, g_{\mathrm{ID}}, (1)),$$
$$(n+2, g_{\oplus}, (2, n+1)), \ldots, (2n, g_{\oplus}, (n, 2n-1))\},$$

with the unit operator set $\mathcal{G}(\mathcal{C}_N) = \{g_{\mathrm{ID}}, g_{\oplus}\}$ where $g_{\mathrm{ID}} : \Sigma \mapsto \Sigma$ is the identity operator, i.e., $g_{ID}(u) = u$, and $g_{\oplus} : \Sigma^2 \mapsto \Sigma$ is the XOR operator, i.e., $g_{\oplus}(u, v) = u \oplus v$.

Example 1 shows a simple CR in which the same small set of operators is reused across positions and scales. The next example presents a more involved arithmetic task, illustrating that the same formalism also applies to computations with richer dependency structures.

*Example 2 (CR of Multiplication (1 * N)):* Consider the Multiplication (1 * N) task $f(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$ where $\Sigma = \{0\} \cup [9]$, $x_1, x_2 \ldots x_n$ are the multipliers, and $y_1 \ldots y_n$ is the product (the digits are reversed), satisfying:

$$x_1 * x_n \ldots x_2 = y_n \ldots y_1.$$

A CR $\mathcal{C} = \{C_n\}$ of this task is given as:

$$\begin{aligned} C_n = \{&(1, g_0, \emptyset), \ldots, (n, g_0, \emptyset), \\ &\times (n+1, g_1, (1, 2)), (n+2, g_2, (1, 2, 3, n+1)), \ldots, \\ &\times (2n, g_2, (1, n, n{+}1, 2n{-}1))\ (2n, g_3, (1, n{+}1, 2n))\}, \end{aligned}$$

with the unit operator set $\mathcal{G}(\mathcal{C}_N) = \{g_1, g_2, g_3\}$ where

$$\begin{aligned} g_1(u_1, u_2) &= u_1 * u_2 \bmod 10, \\ g_2(u_1, u_2, u_3, u_4) &= [u_1 * u_3 + \lfloor u_1 * u_2/10 \rfloor \\ &\quad + \mathbb{1}\left(u_4 < (u_1 * u_2 \bmod 10)\right)] \bmod 10, \\ g_3(u_1, u_2, u_3) &= \lfloor u_1 * u_2/10 \rfloor + \mathbb{1}\left(u_3 < (u_1 * u_2 \bmod 10)\right). \end{aligned}$$

Importantly, a mapping may admit multiple CRs, corresponding to different procedures to compute the mapping. The following examples present alternative CRs for the Parity and Multiplication tasks.

The previous examples present one CR for each mapping. However, a mapping is generally not associated with a unique CR: the same input-output mapping may admit multiple computational procedures. This non-uniqueness is important for our later analysis, since different CRs that can lead to very different LG performance. The next two examples illustrate this point for the Parity and Multiplication tasks.

Example 3 (Another CR of Parity (with CoT)): Suppose that the Parity (with CoT) task f is defined the same as that in Example 1. Another CR $\mathcal{C}' = \{C'_n\}$ of this mapping is:

$$\begin{aligned} C'_n = \{&(1, g_0, \emptyset), \ldots, (n, g_0, \emptyset), \\ &(n+1, g_{\text{ID}}, (1)), (n+2, g_{\oplus,2}, (1,2)), \\ &(n+3, g_{\oplus,3}, (1,2,3)), \ldots, (2n, g_{\oplus,n}, (1,2,\ldots,n))\}, \end{aligned}$$

with the unit operator set $\mathcal{G}(\mathcal{C}'_N) = \{g_{\text{ID}}, g_{\oplus,2}, \ldots, g_{\oplus,N}\}$ where $g_{\text{ID}} : \Sigma \mapsto \Sigma$ is the identity operator, i.e., $g_{ID}(u) = u$, and $g_{\oplus,k} : \Sigma^k \mapsto \Sigma$ is the XOR operator, i.e., $g_{\oplus,k}(u_1, \ldots, u_k) = \bigoplus_{i=1}^{k} u_i$ for all $k = 2, \ldots, N$.

Example 3 shows that even a simple task such as Parity can admit an alternative CR with a substantially larger operator set. The next example shows that the same phenomenon also appears in the more complex Multiplication task.

*Example 4 (Another CR of Multiplication (1 * N)):* Suppose that the Mulitplication (1 * N) task f is defined the same as that in Example 2. Another CR $\mathcal{C}' = \{C'_n\}$ of this mapping can be given as:

$$\begin{aligned} C'_n = \{&(1, g_0, \emptyset), \ldots, (n, g_0, \emptyset), \\ &(n+1, g'_1, (1,2)), \ldots (2n-1, g'_{n-1}, (1,2,\ldots,n)), \\ &(2n, g''_{n-1}, (1,2,\ldots,n))\}, \end{aligned}$$

with the unit operator set $\mathcal{G}(\mathcal{C}'_N) = \{g'_1, \ldots, g'_N, g''_1, \ldots, g''_N\}$ where

$$\begin{aligned} &g'_k(u_1, \ldots, u_{k+1}) \\ &= \left(u_1 * u_{k+1} + \lfloor \& \; sum_{i=2}^{k} \frac{u_1 * u_i}{10^{k+1-i}} \rfloor\right) \bmod 10, \\ &g''_k(u_1, \ldots, u_{k+1}) = \lfloor \sum_{i=2}^{k+1} \frac{u_1 * u_i}{10^{k+2-i}} \rfloor, \end{aligned}$$

for all $k = 1, \ldots, N-1$.

The above examples show that one mapping may have multiple CRs with different unit operator sets. The non-uniqueness can persist even when the operator set is fixed. Different computational structures using the same unit operators may still compute the same mapping.

Example 5: Consider the mapping $f(x_1, \ldots, x_n) = ((x_1 + 1) \bmod 10, (x_1 + 2) \bmod 10, \ldots, (x_1 + n) \bmod 10)$, and the operator set $\mathcal{G} = g_1, g_2$ where $g_1(u) = (u + 1) \bmod 10$ and $g_2(u) = (u + 2) \bmod 10$. The following two CRs $\mathcal{C} = \{C_n\}$ and $\mathcal{C}' = \{C'_n\}$ sharing the same unit operator set represent two different procedures that compute the same function f:

$$\begin{aligned} C_n = \{&(n+1, g_1, (1)), (n+2, g_1, (n+1)), \ldots, \\ &(2n-1, g_1, (2n-2)), (2n, g_2, (2n-2))\}, \\ C'_n = \{&(n+1, g_1, (1)), (n+2, g_2, (1)), \\ &(n+3, g_2, (n+1)), \ldots, (2n, g_2, (2n-2))\}. \end{aligned}$$

A mapping may admit multiple CRs, and these CRs may require different numbers of unit operators. As we will elaborate later, LG is closely related to whether the required computation can be reused across scales, we introduce computational representation complexity to measure the size of the smallest unit operator set needed to compute a mapping.

Definition 3 (Computational Representation Complexity): Suppose that $f : \Sigma^* \mapsto \Sigma^*$ is a sequence-to-sequence mapping. The **Computational Representation Complexity (CRC)** of f up to input length n is the minimal cardinality of the unit operator set among all CRs of f up to length N. Formally,

$$\text{CRC}(f, N) := \min_{\mathcal{C}_N \in \mathfrak{C}_N(f)} |\mathcal{G}(\mathcal{C}_N)|,$$

where the minimization is taken over the set of all computation representations, i.e.,

$$\mathfrak{C}_N(f) := \{\mathcal{C}_N \mid \mathcal{C}_N \text{ is a CR of } f\},$$

and $\mathcal{G}(\mathcal{C}_N)$ denotes the set of unit operators used in $\mathcal{C}_N$.

In simple terms, CRC measures the size of the smallest unit operator set among all CRs of the mapping. We will establish a fundamental limitation of PEs for LG using the notion of CRC in Section V.

D. Canonical CR and Canonical Mapping

CRC captures whether additional operators are required at larger scales. However, even when the same operator set suffices across scales, LG may still depend on whether the computational procedure is represented consistently. To formalize this aspect, we introduce the notions of canonical CR and canonical mapping.

Definition 4 (Canonical CR): A CR $\mathcal{C} = \{C_n\}_{n \in [N]}$ is canonical if for any $C_m, C_n \in \mathcal{C}$, and $(i, g^{(i)}, I_i) \in C_m, (i, \tilde{g}^{(i)}, \tilde{I}_i) \in C_n$, it holds that

$$g^{(i)} = \tilde{g}^{(i)} \quad \text{and} \quad I_i = \tilde{I}_i.$$

Intuitively, a canonical CR enforces consistency of the computation across scales: the same position in the computation follows the same operator and dependency pattern regardless of the input length.

Definition 5 (Canonical Mapping): A sequence-to-sequence mapping $f : \Sigma^* \mapsto \Sigma^*$ is canonical if there exists a canonical CR of f.

This notion will be used later to distinguish tasks for which the computational procedure can be aligned consistently across scales from those for which such alignment is impossible. This distinction will play a key role in understanding when LG fails due to ambiguity in the underlying computation.

IV. Two Barriers to LG With PEs

In this section, we present two examples to illustrate when PEs fail to achieve LG. These examples highlight two qualitatively different limitations.

A. Increasing Operator Requirements

Intuitively, suppose there exists a relatively simple set of unit operators that can be fully observed from small-scale instances, together with a procedure that computes the target mapping using only these operators. In this case, LG may be achievable if the model learns these unit operators during training and receives signals indicating how they should be applied to inputs beyond the training scale.

Conversely, if every procedure capable of solving the task at larger scales requires additional unit operators beyond those needed for the training instances, LG becomes unlikely. This is because the training data provide no information for learning these additional operators. Adapting PEs may influence which computational procedure the model tends to learn, but it does not provide information about new operators that are not present in the training data.

The following example illustrates this intuition.

Example 6: Consider the hypothesis:

$$\mathcal{F}_1 = \{f \mid f : \{0,1\}^{[N]} \mapsto \{0,1\}\}.$$

Suppose that the training data consist of all instances up to scale $N_0 = N - 1$. If the computation of f^* on $\{0,1\}^N$ is unlimited, the PRFs are unlikely to even uniquely encode all possible functions that interpolates the training data. More specifically, the set of all interpolators is

$$\tilde{\mathcal{F}}_1 := \{f \in \mathcal{F}_1 \mid f(x) = f^*(x) \text{ for all } x \in \{0,1\}^{[N_0]}\}.$$

Since

$$\left|\tilde{\mathcal{F}}_1\right| = 2^{2^N},$$

$$|\Phi_N| = \left((N+1)^2\right)^{(N+1)^2} = (N+1)^{2(N+1)^2},$$

then we have

$$\lim_{N\to\infty} \frac{|\Phi_N|}{\left|\tilde{\mathcal{F}}_1\right|} = 0.$$

This means for any algorithms, almost all interpolators cannot identify by PRFs as the set of the interpolators is much "larger" than the set of the PRFs. On the other hand, if the computation of f^* on $\{0,1\}^N$ is the same as one of those on $\{0,1\}^1, \ldots, \{0,1\}^{N_0}$, namely,

$$f^*(x_1, \ldots, x_N) = f_m(x_{i_1}, \ldots, x_{i_m}) \text{ for some } m \le N_0,$$

then we can uniquely identify the mapping by the PRF

$$\phi(i,j) = \begin{cases} (i-1) * N + j, & i < N, \\ (m-1) * N + k, & i = N, j = i_k, \\ m * N, & \text{otherwise.} \end{cases}$$

Example 6 reveals that, when only training data from smaller scales are available, even in the extreme case where $N_0 = N - 1$, PRFs alone are insufficient to identify the target function. This is because PRFs cannot encode the many mappings that are consistent with the observed small-scale data. Consequently, almost all such mappings remain indistinguishable under any choice of PRFs. This limitation is fundamental to PEs for LG and holds regardless of the learning algorithm or model architecture.

We now turn to a different type of limitation, which arises even when the required operator set does not increase.

B. Ambiguity Across Scales

Unlike the previous case, the required operator set may remain the same across scales. The difficulty instead lies in how the computation is extended. In particular, the same position may require different computational behaviors at different scales, even when the observed prefixes are identical. We refer to this phenomenon as ambiguity across scales.

Because PRFs assign positional roles in a scale-invariant manner, a model with PEs must produce the same output at a given position whenever the prefix is the same. As a result, it cannot distinguish between computations that should behave differently across scales, leading to a failure of LG. The following example illustrates this limitation.

Example 7: Consider the copy mapping:

$$f_2(x_1, \ldots, x_n, y_1, \ldots, y_m) = x_{m+1},$$

where $y_i = x_i$ for all $i = 1, \ldots, n$ and $m = 1, \ldots, n-1$. Assume a model with a PRF ϕ (denoted as f_ϕ) can express the copy mapping. Then when $n = 2$, for $x = 10$, we have $f_\phi(1,0,1) = 0$. However, when $n = 3$, for $x = 100$, we have $f_\phi(1,0,1) = 1$, which is a contradiction.

The failure in Example 7 comes from the scale-invariance of PRFs, meaning that the roles of positions are treated the same across different scales. As a result, once a prefix is fixed, the output at a given position is also fixed and must remain the same across scales. However, some procedures, in particular non-canonical CRs, require different behaviors at the same position when the scale changes. This mismatch prevents PEs from achieving LG.

This type of failure can often be addressed by changing how the task is represented. For example, in the copy mapping above, if all inputs are padded to a fixed length $\tilde{N}$, then there exists a procedure where the computation at each position is consistent across scales, removing the ambiguity. In Section VII, we augment the PEs with the scale hint to handle this limitation without modifying the task representation.

These two examples highlight the key challenges for LG with PEs. In the next section, we formalize these limitations and characterize when LG is impossible or achievable using the notions of CRC and canonicality introduced in Section III.

V. Main Results

The examples in Section IV highlight two distinct barriers to LG with PEs: (i) the need for additional unit operators at larger scales, and (ii) ambiguity in how the computation should be extended across scales. In this section, we formalize these two limitations using the notions of CRC and canonicality. We further show that when neither barrier is present, LG can be achieved with PEs. We provide proof sketches for the theorems in this section, with full proofs deferred to Appendix C, (available online).

A. Limitations of PEs

We begin with the limitation arising from increasing operator requirements. Consider the scenario where a model equipped with adaptive PEs aims to learn a Boolean function up to scale N, while being trained only on smaller-scale instances up to scale N_0. We prove that even with an optimal choice of PEs selected with precise knowledge of the target function, the model cannot achieve LG for nearly all functions whose CRC increases from the training scale to the testing scale. Formally, we have the following theorem.

Theorem 1 (Limitation of PEs for Mappings with Increasing CRC): Let $\mathcal{F}_N$ be the set of all Boolean functions $f : \{0,1\}^{[N]} \mapsto \{0,1\}^K$ up to scale N. Suppose that $N_0 : \mathbb{N} \mapsto \mathbb{N}$ satisfies $N_0(n) < n$ for all $n \in \mathbb{N}$, and with slight abuse of notation write N_0 for $N_0(N)$.

Let Φ_N be a family of admissible PRFs up to scale N. For $f \in \mathcal{F}_N$, let

$$f|_{[n]} := \{(x, f(x)) \mid x \in \{0,1\}^{[n]}\}.$$

Define

$$\tilde{\mathcal{F}}_{N_0,N} := \{f \in \mathcal{F}_N \mid \mathrm{CRC}(f, N_0) < \mathrm{CRC}(f, N)\}.$$

Assume that

$$\lim_{N\to\infty} \frac{|\mathcal{F}_{N_0}|\,|\Phi_N|}{|\tilde{\mathcal{F}}_{N_0,N}|} = 0.$$

For any algorithm

$$\mathcal{A} : \bigcup_{n\geq 1} \mathcal{P}\left(\{0,1\}^{[n]} \times \{0,1\}^K\right) \times \Phi \mapsto \mathcal{F},$$

define $\mathcal{F}_{N_0,N}$ be the set of functions whose LG can be achieved by the algorithm $\mathcal{A}$, i.e.,

$$\mathcal{F}_{N_0,N} = \left\{ f \in \tilde{\mathcal{F}}_{N_0,N} \mid \exists \phi \in \Phi_N, \text{ s.t. } \mathcal{A}\left(f|_{[N_0]}, \phi\right) = f \right\}.$$

Then it holds that

$$\lim_{N\to\infty} \frac{\left|\tilde{\mathcal{F}}_{N_0,N} \setminus \mathcal{F}_{N_0,N}\right|}{|\tilde{\mathcal{F}}_{N_0,N}|} = 1.$$

Proof. Sketch: The proof is based on a counting argument. For a given algorithm, a model learned from training instances up to scale N_0 is determined by the restriction of the target function to $\{0,1\}^{[N_0]}$ together with the PRF implemented by the PE. Therefore, the number of functions that can achieve (N_0, N)-LG is upper bounded by the number of such pairs, which grows as $2^{K(2^{N_0+1}-1)} S^{N^2}$.

In contrast, the total number of Boolean functions up to length N is $2^{K(2^{N+1}-1)}$, which grows doubly exponentially with N. Hence the functions that can be learned from scale N_0 data form a negligible subset. Consequently, among the functions whose CRC strictly increases from N_0 to N, almost all cannot achieve (N_0, N)-LG with any PRFs. □

Theorem 1 states that even with full knowledge of the target function, adapting PEs alone is insufficient to achieve LG for almost all functions whose CRC grows from the smaller training scale to the larger testing scale. As the proof of Theorem 1 shows, the underlying reason is that the hypothesis space consistent with the small-scale training data remains extremely large, while PEs alone cannot provide enough information to identify the target mapping.

The limitation in Theorem 1 is inherent to the expressive power of PRFs and therefore applies to all models using PEs. This limitation captures the case where the required operator set grows with scale. However, as illustrated in Example 7, even when the required operator set does not grow, LG may still fail due to ambiguity in how the computation should be extended across scales. We now formalize this second limitation in the following Theorem 2.

Theorem 2 (Limitation of PEs for Non-Canonical Mappings): Let $\mathcal{F}_N$ be the set of all Boolean functions $f : \{0,1\}^{[N]} \mapsto \{0,1\}^K$ up to scale N. Consider an arbitrary but fixed autoregressive model sufficiently expressive to represent any function in $\mathcal{F}_N$ given an appropriate PE. Suppose that $N_0 : \mathbb{N} \mapsto \mathbb{N}$ is a function such that $N_0(n) < n$ for all $n \in \mathbb{N}$ (with slight abuse of notation, we also write N_0 to denote $N_0(N)$ in the theorem). Define $\tilde{\tilde{\mathcal{F}}}_{N_0,N} \subseteq \mathcal{F}_N$ be the set of all non-canonical mappings with non-increasing CRC from N_0 to N. For any learning algorithm

$$\mathcal{A} : \bigcup_{n\geq 1} \mathcal{P}\left(\{0,1\}^{[n]} \times \{0,1\}^K\right) \times \Phi \mapsto \mathcal{F},$$

any mapping $f \in \tilde{\tilde{\mathcal{F}}}_{N_0,N}$, and any PRF $\phi \in \Phi_N$, it holds that

$$\mathcal{A}\left(f|_{[N_0]}, \phi\right) \neq f.$$

Proof. Sketch: Assume that some PRF can achieve LG for a non-canonical mapping. Let F be the autoregressive model learned by the algorithm with the PRF. Then the model F naturally induces a canonical CR of f, i.e., $\mathcal{C} = \{C_n\}_{n\in[N]}$ where

$$C_n = \{(n+i, g_{n+i-1}, (1, \ldots, n+i-1))\}_{1\leq i\leq K},$$
$$g_k(z_1, \ldots, z_k) = F(z_1, \ldots, z_k),$$

for all $n = 1, \ldots, N$, and $k = 1, \ldots, N+K-1$. This contradicts the non-canonicality of f. □

Theorem 2 shows that, for non-canonical mappings, PEs cannot resolve the ambiguity in how the computation should be extended across scales, and thus LG cannot be achieved. This limitation stems from the scale-invariance of PRFs together with the autoregressive nature of the model: the output at a given position is determined by the prefix and therefore cannot adapt when the required computation differs across scales.

Theorems 1 and 2 together identify two limitations of PEs for LG: (i) mappings with increasing CRC, and (ii) non-canonical mappings. We next show that these limitations are tight: when neither occurs, LG can be achieved with PEs.

B. Capabilities of PEs

We now consider the case where neither limitation identified above applies, i.e., the CRC remains invariant across scales and the mapping is canonical. The following theorem shows that these conditions are sufficient for achieving LG with PEs.

Theorem 3 (Capability of PEs): Let $\mathcal{F}_N$ be the set of all Boolean functions $f : \{0,1\}^{[N]} \mapsto \{0,1\}^K$ up to scale N. Suppose that $N_0 : \mathbb{N} \mapsto \mathbb{N}$ satisfies $N_0(n) < n$ for all $n \in \mathbb{N}$, and with slight abuse of notation write N_0 for $N_0(N)$. Let Φ be the set of all PRFs, Φ_N be the set of all PRFs up to N, and $f|_{[n]}$ represent the restricted mapping of f up to n, i.e.,

$$f|_{[n]} := \{(x, f(x)) \mid x \in \{0,1\}^{[n]}\}.$$

Define

$$\hat{\mathcal{F}}_{N_0,N} := \{f \in \mathcal{F}_N \mid \mathrm{CRC}(f, N_0) = \mathrm{CRC}(f, N) \text{ and } f \text{ is canonical}\}.$$

Then there exists a reconstruction map

$$\mathcal{A}_{\text{rec}} : \bigcup_{n \geq 1} \mathcal{P}\left(\{0,1\}^{[n]} \times \{0,1\}^K\right) \times \Phi \mapsto \mathcal{F}$$

such that for all N and $f \in \hat{\mathcal{F}}_{N_0,N}$, with some $\phi_f \in \Phi_N$, it holds that

$$\mathcal{A}_{\text{rec}}\left(f|_{[N_0]}, \phi_f\right) = f.$$

Proof. Sketch: The proof is constructive. For each canonical mapping with non-increasing CRC, we construct a PRF together with a reconstruction algorithm that recovers the mapping from its restriction to scale N_0.

For all N and $f \in \hat{\mathcal{F}}_{N_0,N}$, there exists a canonical CR $\mathcal{C}_f$ of f such that

$$\mathcal{G}(\mathcal{C}, N_0) = \mathcal{G}(\mathcal{C}, N).$$

This ensures that all unit operators are already observable from the training data, and it is possible to align them across different scales using a PRF.

Now we are to construct a concrete PRF for the $\mathcal{C}_f$. To formalize the construction for general mappings, we introduce the notion of a characterizing PRF, which serves as the key mechanism for aligning the computation across scales.

Definition 6 (Characterizing PRF for CR): Suppose that $\mathcal{C} = \{C_n\}$ is a CR of a mapping $f : \Sigma^* \mapsto \Sigma^*$. Let $\phi : \mathbb{N} \times \mathbb{N} \mapsto \mathbb{N}$ be a PRF and $\phi^* \in \phi(\mathbb{N} \times \mathbb{N})$ be a constant. We say that ϕ is a characterizing PRF for the CR $\mathcal{C}$ if it satisfies the following two conditions:

1) (Consistency) For any m, n, and two tuples $(i, g^{(i)}, I_i) \in C_m$ and $(i', g^{(i')}, I_{i'}) \in C_n$

$$\phi(i-1, j) = \phi(i'-1, j') \neq \phi^*,$$

then it must hold that

$$g^{(i)} = g^{(i')} \quad \text{and} \quad I_i^{-1}(j) = I_{i'}^{-1}(j'),$$

where

$$I_\alpha^{-1}(\beta) := \begin{cases} k, & \text{if } \beta \text{ is the k-th element in } I_\alpha, \\ +\infty, & \text{otherwise.} \end{cases}$$

2) (Distinctness) For all $C_n \in \mathcal{C}$, if $(i, g^{(i)}, I_i) \in C_n$, then $j \notin I_i$; if and only if

$$\phi(i-1, j) = \phi^*.$$

Algorithm 1: Reconstruction algorithm $\mathcal{A}_{\text{rec}}$.

Input: A restricted mapping $f|_{[N_0]}$, and a PRF ϕ

Output: A reconstructed CR $\hat{\mathcal{C}}$

1: $\hat{\mathcal{G}}, \hat{\mathcal{H}} \leftarrow$ UnitOperatorSetRecovery$(f|_{[N_0]}, \phi)$, where $\hat{\mathcal{G}}$ is the unit operator set and $\hat{\mathcal{H}}$ is the lookup table

2: $\hat{\mathcal{C}} \leftarrow$ CRReconstruction$(\hat{\mathcal{G}}, \hat{\mathcal{H}})$

3: **return**the reconstructed CR $\hat{\mathcal{C}}$

Intuitively, a characterizing PRF assigns identifiers to position pairs such that unit operators and their input roles can be consistently aligned across different scales. The consistency condition ensures that identical identifiers correspond to the same operator and input role, while the distinctness condition guarantees that irrelevant positions remain distinguishable.

We first show the existence of characterizing PRF for the CR $\mathcal{C}_f$. We then construct an algorithm $\mathcal{A}_{\text{rec}}$ (see Algorithm 1) that can identify the target f with its restricted mapping up to scale N_0 and the characterizing PRF for the CR f.

The algorithm $\mathcal{A}_{\text{rec}}$ recovers the set of unit operators $\mathcal{G}(\mathcal{C}_f, N_0)$ and a lookup table $\mathcal{H}_f$ that identifies the corresponding inputs for the unit operators across different scales (i.e., UnitOperatorSetRecovery; see Algorithm 2 in Appendix C, available online). With the set of unit operators $\mathcal{G}(\mathcal{C}_f, N_0)$ and the lookup table $\mathcal{H}_f$, the algorithm $\mathcal{A}_{\text{rec}}$ can reconstruct the complete CR $\mathcal{C}_f$ (i.e., CRReconstruction; see Algorithm 3 in Appendix C, available online). The consistency and distinctness properties of the characterizing PRF ensure that the reconstruction algorithm produces a CR equal to $\mathcal{C}_f$. □

Theorem 3 shows that, for canonical mappings with non-increasing CRC, LG can be achieved with PEs. The constructive proof also suggests a practical design principle for PEs: selecting a characterizing PRF corresponding to a canonical CR of the mapping. While we do not explicitly implement Algorithm 1 to reconstruct the CR, we demonstrate empirically in the next section that PEs based on such characterizing PRFs can effectively promote LG.

VI. Experiments

We have shown that PEs can achieve LG for mappings that are canonical and have non-increasing CRC. In particular, the constructive proof of Theorem 3 suggests that a characterizing PRF is sufficient for identifying the target mapping.

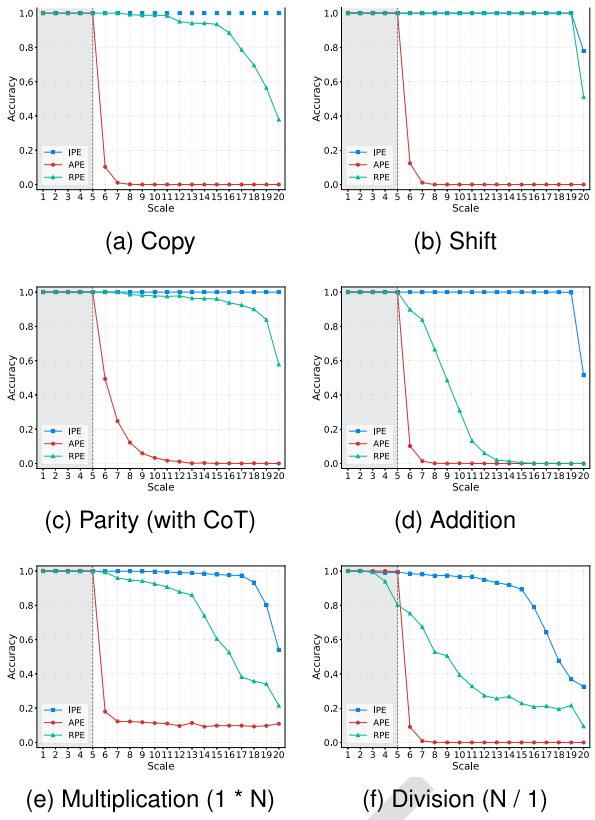

Fig. 1. Evaluation results of models using different PEs across six tasks. Each model is trained on 10,000 samples of scales 1–5 for 300 epochs, with evaluation performed on 1,000 samples at each scale (1–20). Checkpoints are saved every 30 epochs. For each configuration, the plotted curve corresponds to the checkpoint that achieves the best average performance across all scales.

While Algorithm 1 is not directly applicable to practical settings, this result provides a guiding principle for designing PEs: a charactering PRF for the canonical CR of non-increasing CRC should facilitate LG.

To empirically validate this insight, we conduct experiments comparing three types of PEs across various reasoning tasks: Ideal PE (IPE), APE, and RPE. IPE is the PE whose PRF faithfully characterizes the canonical CR of the minimum unit operator set. APE and RPE are the most common PEs as the baselines in the experiments.

We consider six tasks. For each task, the instances are all aligned to a target length to guarantee the existence of a characterizing PRF. Experimental results shown in Fig. 1 demonstrate that when the operators required at larger scales are already present within the training domain, IPE outperforms both APE and RPE, especially in the three relatively complicated tasks: Addition, Multiplication, and Division. More details regarding experimental setups and implementations are provided in Appendix D.

VII. Extensions

The results in Sections V-B and VI establish that PEs can enable LG when the target mapping is canonical and has non-increasing CRC, provided that the underlying PRF characterizes a suitable CR.

However, directly applying this insight in practice faces two challenges. First, for non-canonical mappings, it may be impossible to construct a characterizing PRF, even when the CRC does not increase. Second, designing task-specific PEs requires substantial prior knowledge and is often impractical.

We introduce two extensions of PEs to address the two challenges respectively: the *scale hint* technique and *learning-based position embeddings*.

A. Scale Hint Technique

In Section V-B, we established that LG is achievable when the task has a canonical CR whose CRC does not increase from training to testing scales, provided a suitable PRF characterizing the corresponding CR. On the one hand, the canonicality is essential for LG with PEs. On the other hand, the canonicality does not seem inherent for the scaling challenge of LG as a simple modification of the representation format can change the canonicality. See the next example for an illustration.

Example 8: Consider the Addition task where the numbers are only aligned scale-wisely, i.e., an instance of scale n is

$$\mathtt{x_1 \ldots x_n + y_1 \ldots y_n = z_1 \ldots z_n z_{n+1}},$$

for $n = 1, \ldots, N$. For such an unaligned Addition task, no CR $\mathcal{C} = \{C_n\}$ is canonical and thus the unaligned Addition mapping is non-canonical. However, if the numbers are aligned to some constant $\tilde{N}$ by padding with 0, then the mapping becomes canonical.

Example 8 shows that a non-canonical mapping can be transformed into a canonical one by modifying the representation format, suggesting that the canonicality barrier is not intrinsic to the task itself.

However, modifying the input representation is not always feasible. Instead, we address this issue at the level of PEs by allowing the PRF to vary across scales.

In practice, the scale of an instance is often known or can be reasonably estimated. The core idea of the Scale Hint (SH) technique is to leverage this information by augmenting the PRF with the instance scale as an additional input. Formally, we define the PRF with Scale Hint (PRF-SH) as a mapping that explicitly incorporates the instance scale, i.e., $\phi : [N] \times [N] \times [N] \mapsto [S]$ that maps the query position i, the key position j, and the instance scale n to some value $\phi(i, j, n)$. The resulting position embedding is denoted as PE-SH. Analogous to Definition 6, we can define when a PRF-SH characterizes a CR.

Incorporating scale hints makes the PRF strictly more expressive, enabling the characterization of a broader class of CRs. In fact, the following Theorem 4 shows that PRF-SH is complete for characterizing CRs of non-increasing CRC.

Theorem 4: For any CR $\mathcal{C} = \{C_n\}$ of non-increasing CRC, there exists a corresponding PRF-SH that characterizes it.

Using PE-SH not only broadens the applicability but also enables more compact and efficient representations, thus facilitating more effective learning. For example, consider the Addition task. If we do not employ the scale hint, we need to

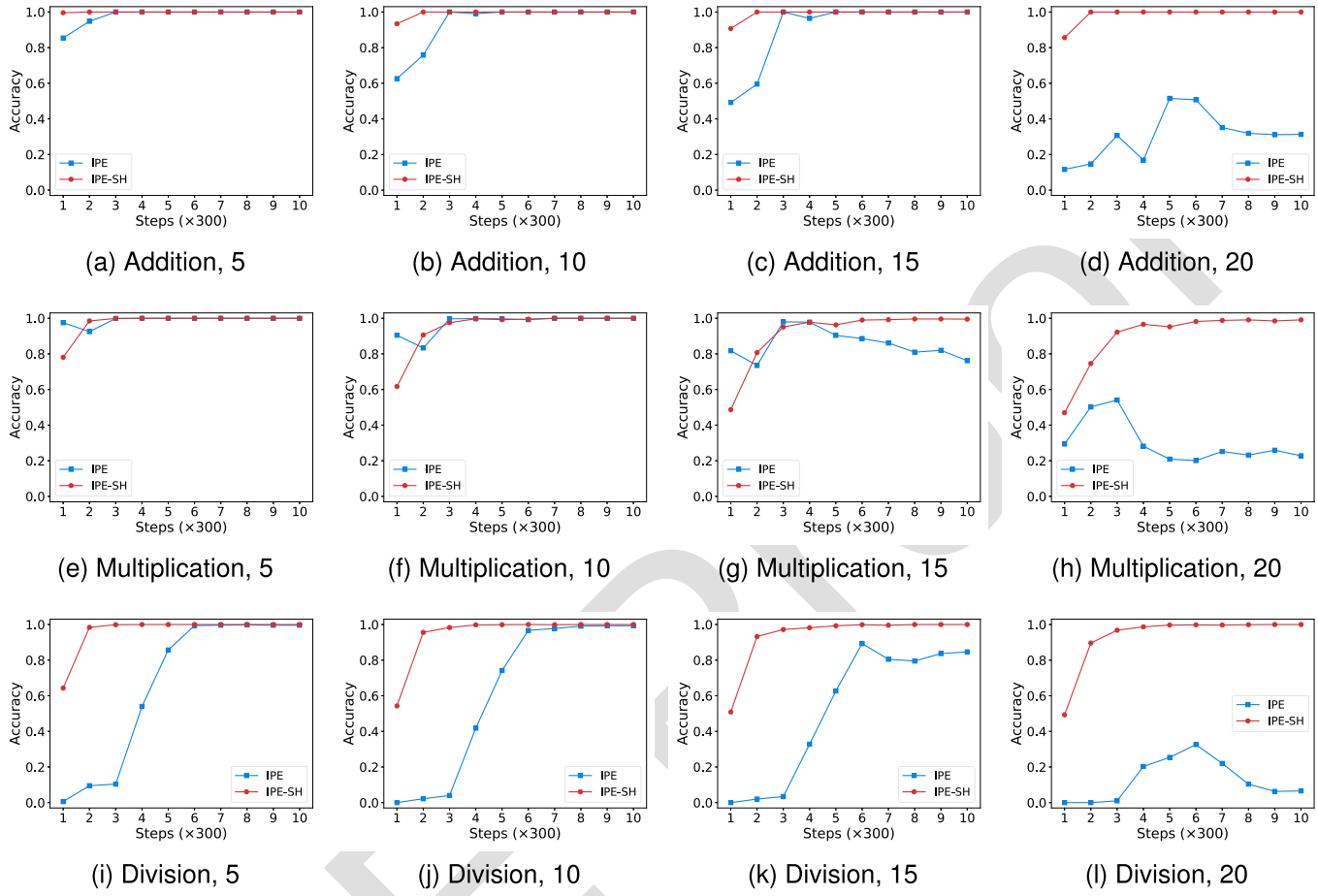

Fig. 2. Comparison between IPE and IPE-SH in Addition, Multiplication (1 * N), and Division (N / 1). For IPE, we align input samples to scale 20, whereas IPE-SH operates without scale alignment. Both models are trained on samples of scales 1–5 and evaluated on scales 5, 10, 15, 20 (the numbers in the subpations mean the evaluation scales). For clarity, we present only the evaluation results on scales 16–20.

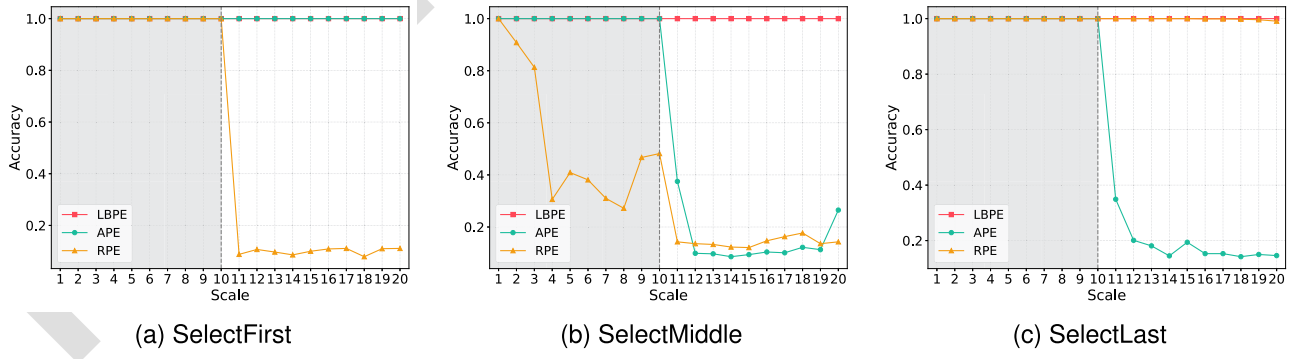

Fig. 3. Evaluation results of models with LBPE across three different Select tasks (SelectFirst, SelectMiddle, and SelectLast). Each model is trained on 1,000 samples of scales 1–10 for 2,000 epochs and evaluated on 1,000 samples at each scale 1–20. We save checkpoints every 20 epochs. For each configuration, we plot the curve for the checkpoint of the best average performance across all scales. Q3

align all the instances to the maximum target length:

$$\mathtt{x_1 \ldots x_n} \underbrace{\mathtt{0 \ldots 0}}_{N-n} + \mathtt{y_1 \ldots y_n} \underbrace{\mathtt{0 \ldots 0}}_{N-n} = \mathtt{z_1 \ldots z_n z_{n+1}} \underbrace{\mathtt{0 \ldots 0}}_{N-n}.$$

These redundant padding zeros increase computation and memory usage and may confound the model's ability to identify the correct positions and operators. By contrast, if we apply the scale hint, we can represent the instance as:

$$\mathtt{x_1 \ldots x_n + y_1 \ldots y_n = z_1 \ldots z_n z_{n+1}},$$

which significantly reduces overhead when $n \ll N$. Fig. 2 is a comparison between IPE and IPE-SH in Multiplication and Division, respectively. The results demonstrate that incorporating

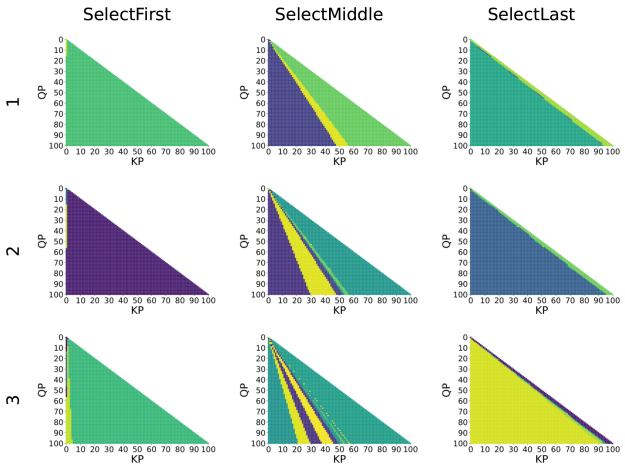

Fig. 4. Visualization of the learned PRFs in the three Select tasks. For each task, we show the predicted PRF values corresponding to the top three prediction weights (ranked 1–3 from top to bottom) for each query position i and key position j, where $i \leq j$. "QP" and "KP" mean "query position" and "key position", respectively.

a scale hint accelerates convergence and improves LG performance. Moreover, since no target length is fixed, PE-SH enables more flexible length generalization, allowing extrapolation to larger-scale instances beyond the limits imposed by fixed-length alignment.

B. *Learning-Based Position Embeddings*

Thus far, we have demonstrated that handcrafting appropriate PEs (or PE-SH) enables LG whenever the CRC of the task does not increase from training to testing scales. However, practically speaking, perfect prior knowledge regarding the positional relationships within a task is usually unavailable. Moreover, designing new handcrafted PEs for each individual task is prohibitively expensive.

To address these limitations, we propose Learning-Based Position Embeddings (LBPE), in which the PRF (or PRF-SH) itself is made into a learnable component. Importantly, the proposed LBPE differs fundamentally from existing learnable PEs in prior literature. Specifically, the conventional learnable PE has a fixed PRF and learns embedding vectors, whereas our LBPE method explicitly learns the PRF function itself.

LBPE can be implemented either with or without learnable embedding vectors. Here, we present one implementation of LBPE utilizing learnable embedding vectors for simplicity. Let $P \in \mathbb{R}^{S\times d}$ be the learnable embedding vectors and $\phi(i,j;\theta) : [N]\times[N] \mapsto \Delta^{[S]}$ be the Learning-Based PRF (LBPRF), where i,j are the query and key positions respectively, S is an upper bound of the PRF value, and θ is the learnable parameter of the LBPRF block. Then the LBPE with the learnable parameters θ and P is

$$\mathrm{LBPE}(i,j;\theta,P) = P^{\intercal}\phi(i,j;\theta).$$

For notation simplicity, we write $\mathrm{LBPE}(i,j;\theta,P)$ as $\mathrm{LBPE}(i,j)$ when this does not lead to misunderstanding.

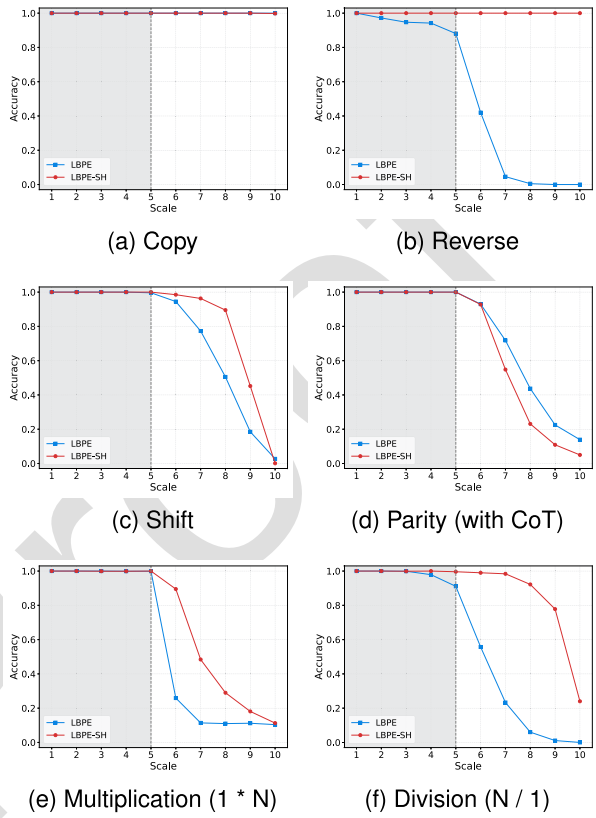

Fig. 5. Evaluation results of models with LBPE and LBPE-SH on various tasks. Each model is trained on 10,000 samples from scales 1–5 for 10,000 epochs (equivalent to 100,000 steps under our hyperparameter setting). Checkpoints are saved every 10,00 steps and evaluated on 1,000 samples at each scale from 1 to 10. The curves are the evaluation results of the checkpoints achieving the best average accuracies (over all scales).

We can simply replace any learnable PE with the above LBPE. For instance, we can adapt a Transformer with key-only RPE (i.e., the RPE is only added to the key embedding) to use LBPE by replacing RPE_{i-j} with $\mathrm{LBPE}(i,j)$. Denote the learnable RPE at layer l by $\mathrm{RPE}^{(l)}_{i-j}$. The query-key weight $\alpha^{(l)}_{i,j}$ at layer l from

$$\alpha^{(l)}_{i,j} = \left(h^{(l-1)}_j + \mathrm{RPE}^{(l)}_{i-j}\right)^{\intercal} W^{(l)\intercal}_K W^{(l)}_Q h^{(l-1)}_i,$$

becomes

$$\alpha^{(l)}_{i,j} = \left(h^{(l-1)}_j + \mathrm{LBPE}(i,j;\theta^{(l)},P^{(l)})\right)^{\intercal} W^{(l)\intercal}_K W^{(l)}_Q h^{(l-1)}_i.$$

However, we emphasize that LBPE is not restricted to this particular approach; other forms of PE such as RoPE can also be integrated into LBPE frameworks (see Appendix B, available online).

We evaluate LBPE, implemented with learnable positional encodings, on three Select tasks that involve identifying tokens at specific positions. Specifically, we consider SelectFirst, SelectMiddle, and SelectLast, which correspond to selecting x_1, $x_{\lfloor n/2\rfloor+1}$, and x_n, respectively, given an input of the form "$x_1 \ldots x_n =$." The results, presented in Fig. 4, show that APE and RPE achieve LG only in SELECTFIRST and SELECTLAST, respectively, whereas LBPE succeeds in all three tasks. Fig. 4 also visualizes the learned PRFs, revealing that LBPE adaptively

captures task-specific positional relationships that closely align with the characterizing PRFs of each task.

Similarly, we can directly combine LBPE with the SH technique, resulting in LBPE-SH:

$$\text{LBPE-SH}(i, j, n; \theta, P) = P^{\intercal}\phi(i, j, n; \theta).$$

For learning-based PEs, incorporating the SH technique can also enhance LG, similar to the effect observed with handcrafted PEs. As shown in Fig. 5, while both LBPE and LBPE-SH achieve LG in the Copy task, LBPE-SH outperforms LBPE in all other tasks except Parity.

Although LBPE offers the flexibility to automatically learn diverse positional relationships, it is unrealistic to expect a single LBPE model to achieve LG universally across all tasks. Task-specific prior knowledge remains essential for guiding the design of learning modules, as different architectural choices inherently bias the model toward capturing particular types of positional relationships. We leave a systematic investigation of how different architectures induce distinct biases in LBPEs as an important direction for future work.

VIII. Conclusion

We analyze the role of PEs in LG. On the negative side, PEs have two fundamental limitations: they cannot achieve LG for mappings with increasing CRC or non-canonicality. On the positive side, PEs can enable LG when these conditions are absent. To effectively leverage PEs for LG, one should select a PE with a characterizing PRF. We further propose the scale hint technique, which overcomes the limitation for non-canonical mappings and extends the applicability of PEs to a wider range of tasks, as well as LBPE, which alleviates the need for handcrafted PE design for each task.

Future Work: We establish only the theoretical feasibility of PEs for LG from a capability perspective. A rigorous theoretical characterization of their capabilities in practical models with learning algorithms remains an important direction for future work. While our analysis focuses on PRFs, the impact of different PE implementations warrants further investigation. Extending the scale hint technique to natural language tasks, where scales are less explicit, and understanding how model architectures influence the PRFs learned in LBPE are also promising directions for future research.

References

[1] C. Anil et al., "Exploring length generalization in large language models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 3546–38556.

[2] H. Zhou et al., "What algorithms can transformers learn? A study in length generalization," 2023, *arXiv:2310.16028*.

[3] Y. Zhou, U. Alon, X. Chen, X. Wang, R. Agarwal, and D. Zhou, "Transformers can achieve length generalization but not robustly," 2024, *arXiv:2402.09371*.

[4] X. Huang et al., "A formal framework for understanding length generalization in transformers," in *Proc. 13th Int. Conf. Learn. Representations*, 2025.

[5] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behav. Brain Sci.*, vol. 40, 2017, Art. no. e253.

[6] D. Bahdanau, S. Murty, M. Noukhovitch, T. H. Nguyen, H. de Vries, and A. Courville, "Systematic generalization: What is required and can it be learned?," in *Proc. Int. Conf. Learn. Representations*, 2019.

[7] B. M. Lake and M. Baroni, "Human-like systematic generalization through a meta-learning neural network," *Nature*, vol. 623, no. 7985, pp. 115–121, 2023.

[8] Y. Chen, L. Yang, Y. Liang, and Z. Lin, "Low-dimension-to-high-dimension generalization and its implications for length generalization," in *Proc. Int. Conf. Mach. Learn.*, 2025.

[9] A. Kazemnejad, I. Padhi, K. N. Ramamurthy, P. Das, and S. Reddy, "The impact of positional encoding on length generalization in transformers," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, vol. 36.

[10] S. Jelassi, S. d'Ascoli, C. Domingo-Enrich, Y. Wu, Y. Li, and F. Charton, "Length generalization in arithmetic transformers," 2023, *arXiv:2306.15400*.

[11] M. Hahn and M. Rofin, "Why are sensitive functions hard for transformers?," 2024, *arXiv:2402.09963*.

[12] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," 2018, *arXiv:1803.02155*.

[13] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist.: Hum. Lang. Technol.*, 2019, vol. 1, pp. 4171–4186.

[15] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "RoFormer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, 2024, Art. no. 127063.

[16] C. Han et al., "LM-Infinite: Zero-shot extreme length generalization for large language models," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist.: Hum. Lang. Technol.*, 2024, vol. 1, pp. 3991–4008.

[17] Z. Hu et al., "LongRecipe: Recipe for efficient long context generalization in large language models," 2024, *arXiv:2409.00509*.

[18] S. Li et al., "Functional interpolation for relative positions improves long context transformers," in *Proc. 12th Int. Conf. Learn. Representations*, 2024.

[19] L. Fang et al., "What is wrong with perplexity for long-context language modeling?," in *Proc. 13th Int. Conf. Learn. Representations*, 2025.

[20] J. Yuan et al., "Native sparse attention: Hardware-aligned and natively trainable sparse attention," 2025, *arXiv:2502.11089*.

[21] E. Abbe, S. Bengio, A. Lotfi, and K. Rizk, "Generalization on the unseen, logic reasoning and degree curriculum," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 31–60.

[22] C. Xiao and B. Liu, "Generalizing reasoning problems to longer lengths," in *Proc. 13th Int. Conf. Learn. Representations*, 2025.

[23] D. Teney, A. M. Nicolicioiu, V. Hartmann, and E. Abbasnejad, "Neural redshift: Random networks are not random functions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 4786–4796.

[24] Y. Li, T. Ma, and H. Zhang, "Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations," in *Proc. Conf. Learn. Theory*, 2018, pp. 2–47.

[25] K. Lyu, J. Jin, Z. Li, S. S. Du, J. D. Lee, and W. Hu, "Dichotomy of early and late phase implicit biases can provably induce grokking," in *Proc. 12th Int. Conf. Learn. Representations*, 2024.

[26] J. Pérez, P. Barceló, and J. Marinkovic, "Attention is turing-complete," *J. Mach. Learn. Res.*, vol. 22, no. 75, pp. 1–35, 2021.

[27] Z. He et al., "Two stones hit one bird: Bilevel positional encoding for better length extrapolation," in *Proc. 41st Int. Conf. Mach. Learn.*, 2024, pp. 17858–17876.

[28] C. Finn, K. Xu, and S. Levine, "Position coupling: Improving length generalization of arithmetic transformers using task structure," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, vol. 37, pp. 22233–22315.

[29] S. McLeish et al., "Transformers can do arithmetic with the right embeddings," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, vol. 37, pp. 108012–108041.

Yang Chen received the BS degree in computer science from Peking University. He is currently working toward the PhD degree with Peking University. His work has been authored or coauthored at top-tier conferences, such as ICML and NeurIPS. His research interests include reasoning, learning theory, and optimization.

Yitao Liang received the PhD degree in computer science from the University of California, Los Angeles, advised by prof. Guy Van den Broeck. He is currently an assistant professor with Peking University. His research interests include knowledge reasoning, machine learning, and AI agents. His work has received recognition from top AI conferences such as the best-paper honorable mention from AAMAS 2016, the best paper from RL for Real Life workshop in ICML 2019, a best paper runner-up from the LLD workshop in NeurIPS 2017, and best paper from the TEACH workshop in ICML2023. He is an area chairs and senior area chairs in top venues, such as NeurIPS and ICML. He is a Boya fellow with Peking University

Zhouchen Lin (Fellow, IEEE) received the PhD degree in applied mathematics from Peking University in 2000. He is currently a Boya Special professor with the State Key Laboratory of General Artificial Intelligence, School of Intelligence Science and Technology, Peking University. He has authored or coauthored more than 350 technical papers, collecting more than 40,000 Google Scholar citations. His research interests include machine learning and numerical optimization. He has been a program co-chair of ICPR 2022, area chairs of ACML, ACCV, CVPR, ICCV, NIPS, NeurIPS, AAAI, IJCAI, ICLR, and ICML and senior area chairs of ICML, NeurIPS, CVPR, and ICLR for many times. He was an associate editor for *IEEE Transactions on Pattern Analysis and Machine Intelligence.* He is an associate editor for *International Journal of Computer Vision* and *Optimization Methods and Software*. He is a fellow of IAPR, AAIA, and CSIG.