MIND THE GAPS: LEARNING POSITIONS THAT GENERALISE

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformers can solve algorithmic tasks nearly perfectly within the domain; however, they typically fail on out-of-distribution evaluation, specifically on longer input sequences than those seen during training. This lack of generalisation has been partially addressed in the literature by assigning token positions according to their computational roles. Removing this guidance is challenging: the decoder can adapt to learned positional assignments that support accurate predictions within the training range but fail to generalise to longer sequences. Here, we show that models can learn positional assignments that support length generalisation from next-token prediction alone. To make these assignments easier to identify, we train across randomly gapped coordinate systems. These systems preserve exact positional matches, whereas mismatches induce varying displacements that are harder for the decoder to compensate for consistently. Building on this principle, we introduce **Re**ference-relative **RoPE** (ReRoPE), which learns a mixture of reference-relative RoPE transformations. For each token, ReRoPE scores the K nearest visible occurrences of every token type as candidate references, with each candidate defining a transformation in the mixture. Across addition, copy, reverse, and parity tasks, ReRoPE generalises to lengths several times the training maximum, matching supervised position coupling. Further experiments on addition show that ReRoPE retains this performance in deeper models and exceeds position coupling in terms of the maximum length to which the model generalises. Code: Upon Acceptance

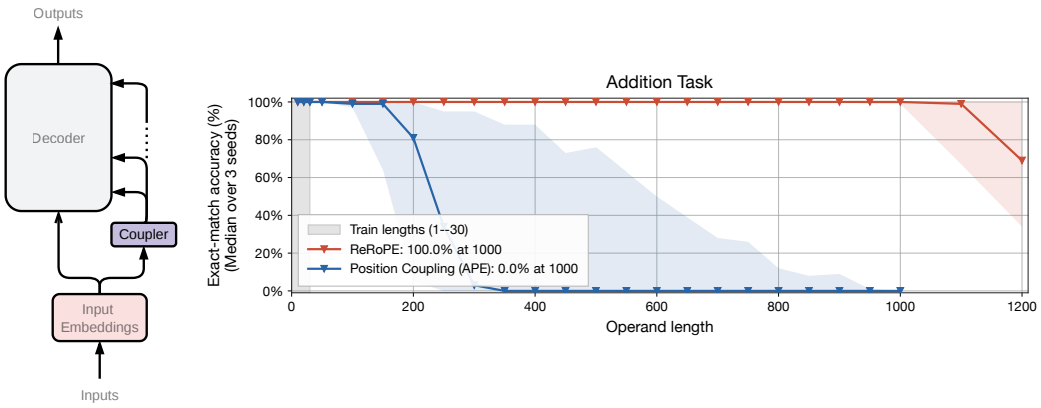

Figure 1: **ReRoPE learns positional assignments that extrapolate far beyond the training range.** **(a)** Introduction of the new Coupler mechanism. **(b)** Exact-match accuracy on addition (2L8H decoder) after training on operands of at most 30 digits. ReRoPE generalises beyond 1,000 digits and exceeds the extrapolation range of supervised Position Coupling while needing no additional supervision and learning from task loss alone.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Transformers often struggle to generalise beyond the sequence lengths encountered during training (Anil et al., 2022; Delétang et al., 2023; Press et al., 2022; Kazemnejad et al., 2023; Zhou et al., 2024a;b; Lee et al., 2024). This limitation is particularly evident in algorithmic and arithmetic tasks, where models can achieve perfect accuracy within the training range yet fail on longer inputs governed by the same underlying rule (Nogueira et al., 2021; Kazemnejad et al., 2023; Zhou et al., 2024a;b). Such failures suggest that models learn to fit their training data without recovering the underlying algorithm. Recovering such algorithms is important because many forms of reasoning and computation require composing the same operations even as inputs grow longer. Learning the relevant computational structure from short examples in a form that generalises to longer inputs, therefore, remains an important challenge.

Correcting positional encodings offers a promising way to address this problem. A token's location in a sequence does not necessarily reflect its role in the computation: in addition, for example, digits should align according to their significance rather than their absolute position in the prompt. Relative and context-dependent encodings provide more flexible positional representations (Li et al., 2024; Golovneva et al., 2024; Zhu et al., 2025; Lee, 2025), while Abacus and Position Coupling explicitly introduce task-aligned coordinates (McLeish et al., 2024; Cho et al., 2024; 2025).

Despite this progress, several obstacles remain. First, task-aligned positional methods still rely on human specification: position coupling requires assigning an index label to every token. Secondly, a central difficulty is that trying to discover these positions under a jointly trained decoder does not necessarily encourage the intended positional structure. An incorrect assignment can produce a consistent offset that the decoder learns to compensate for, allowing training loss to decrease without correcting the assignment itself. Greater decoder capacity often makes such compensation even easier, rather than improving positional learning.

Addressing these limitations requires both replacing the manual supervision with a learned mechanism and discouraging the decoder from compensating for positional misalignments. Our key idea is to construct each token's positional representation from its distances to other tokens (denoted as *references*). A new positional encoding is then computed as a mixture of RoPE matrices (Su et al., 2024) reflecting these reference-relative distances. If two tokens have the same relative distance towards tokens they highly reference, they are assigned the same new positional encoding, making them behave as if they were coupled. To discourage the decoder from compensating for mismatches within the training distribution, the spacing between reference distances is randomised during training while preserving exact distance matches (which we denote as *random-gap augmentation*). This turns fixed, nonzero mismatches into varying positional displacements, making consistent offsets less of a reliable shortcut for the decoder and favouring exact alignments.

These ideas lead to Reference-relative RoPE (ReRoPE). Rather than prescribing task-specific coordinates, ReRoPE learns which token occurrences should serve as references for each token using a lightweight *coupler*. Their relative distances induce rotary positional transformations, which are combined to form the token's positional representation. Random-gap augmentation then makes fixed positional mismatches unstable across training examples, reducing the decoder's ability to compensate for an incorrect alignment. Together, these components allow ReRoPE to learn task-aligned positional structure without manually specifying the alignment itself.

Our main contributions are:

1. We introduce *ReRoPE*, a learned reference-relative rotary positional encoding that discovers task-aligned positional structure directly from the next-token prediction objective, without task-specific coordinate supervision.
2. We develop a normalised mixture of reference-relative RoPE rotations that translates soft reference selection into positional transformations of attention queries and keys. This allows reference selection and positional representations to be learned jointly and end-to-end.
3. We repurpose positional randomisation as an *anti-compensation mechanism* for learning alignments. Random-gap augmentation preserves exact reference-distance matches while randomising the displacement induced by mismatched assignments. Unlike RandPE (Ruoss et al., 2023), where randomised positions are used to expose the model to a broader

range of positional indices, here randomisation makes systematic positional errors unstable across training examples, preventing the decoder from reliably learning a fixed correction.

We evaluate ReRoPE on algorithmic tasks, asking whether the model can learn computation-aligned positions directly from minimising the next token prediction loss and whether such a model exhibits strong generalisation to longer sequences. ReRoPE matches the success of supervised Position Coupling across these tasks and, in the extended addition comparison, exceeds the extrapolation range of the tested baseline, achieving 99.7% exact-match accuracy on $1{,}000$-digit operands after training only on lengths 1–30. Ablations on addition identify random-gap augmentation as essential to extrapolation in the tested setting, while additional experiments show that increasing the depth of the decoder does not harm generalisation to longer sequences. Together, these results show that task-aligned positional structure need not be supplied as supervision: it can be learned from the prediction objective while preserving strong generalisation beyond the training range.

2 Related Work

Sequence representations for algorithmic extrapolation. The representation of an algorithmic problem can substantially change the computation learned by a Transformer. In addition, reversing the output order places the least significant digits first and can make the underlying computation easier to learn (Lee et al., 2024). Other choices, such as spacing, positional descriptions, and recursive scratchpads, likewise have a large effect on extrapolation beyond the training range (Shen et al., 2023). More generally, Zhou et al. (2024a) relates length generalisation to whether the input representation exposes a regular computation that can be repeated at unseen lengths. Task hinting and index hinting make such a structure more explicit by adding auxiliary signals or positional marker tokens to the input (Awasthi & Gupta, 2024; Zhou et al., 2024a). ReRoPE instead leaves the symbolic task representation unchanged and modifies how positional relations are inferred and consumed.

Positional encodings for length generalisation. Transformers require some means of distinguishing token order, but positional mechanisms differ considerably in how they behave outside the range observed during training (Vaswani et al., 2017). NoPE investigates whether order can be recovered without an explicit positional encoding (Kazemnejad et al., 2023), whereas ALiBi and RoPE introduce relative biases or rotations designed to support extrapolation (Press et al., 2022; Su et al., 2024). FIRE, CoPE, TAPE, and PRISM further parameterise position as a learned function of distance, content, or context (Li et al., 2024; Golovneva et al., 2024; Zhu et al., 2025; Lee, 2025). Randomized positional encodings train on multiple monotone coordinate systems rather than a single fixed sequence of indices (Ruoss et al., 2023). ReRoPE builds on this randomisation principle, but applies it to candidate reference-relative coordinates: random gaps are used to distinguish competing positional assignments, rather than only to augment a fixed assignment of positions to tokens.

Computation-aligned positional supervision. A complementary line of work assigns positions according to the structure of the target computation. Abacus embeddings represent digits by their significance within a number (McLeish et al., 2024). Position Coupling gives equal position labels to tokens that play corresponding computational roles, and Multi-operand Position Coupling extends this construction across varying numbers and lengths of operands (Cho et al., 2024; 2025). These methods demonstrate that computation-aligned coordinates can produce strong length extrapolation, but obtain those coordinates from manually specified, task-dependent rules.

Learning positional assignments. Predictive Position Coupling replaces the direct use of coupled positions with a prediction problem, while still training against supervised position targets (Golowich et al., 2025). ReRoPE tackles the problem of discovering target positions from task loss alone. The nearest visible occurrences of a very token type are treated as candidate references, and their assignments are learned from next-token prediction alone. Reference-relative coordinates are consumed as mixtures of rotary transformations, while randomly gapped coordinate systems provide pressure to reject assignments that fit short sequences but do not extrapolate.

3 REROPE

3.1 REFERENCE

Let x_i denote the token at sequence position i. If x_i uses x_j as its *reference*, the candidate positional coordinate assigned to x_i is their unsigned sequence distance:

$$d_{i,j} = |i - j|. \tag{1}$$

For example, in the sequence `12+34=`, the digits `2` and `4` have the same significance but different sequence indices. Referencing their respective following separators, `+` and `=`, assigns both coordinate 1. The same rule assigns coordinate 1 to tokens `3` and `6` in `123+456=`, independently of operand length.

Visibility rule. Each example consists of an observed prompt followed by a target continuation. For example, in `12+34=46`, `12+34=` represents the observed prompt and `46` the target continuation. Prompt tokens may reference any token in the observed prompt, whereas target tokens may reference only tokens up to and including themselves. This rule governs reference selection; decoder self-attention retains its causal mask. A prompt token's positional representation may therefore depend on later prompt tokens, but never on future target tokens. Under teacher forcing, position i consumes x_i and predicts x_{i+1}, so self-reference does not reveal the prediction target.

3.2 LEARNING WHICH REFERENCES TO USE

Candidate references. For each vocabulary type, ReRoPE orders its visible occurrences by distance from the current token. The nearest occurrence receives rank zero, the next-nearest rank one, and so on, with equal-distance ties resolved in favour of the occurrence on the left. We call this the *within-type distance rank* and denote the rank of candidate j relative to token i by r_{ij}.

In order to bound the number of terms in the softmax (Veličković et al., 2025), ReRoPE retains the (up to) K nearest visible occurrences of every token type:

$$\mathcal{A}_i = \{j \mid j \text{ is visible to } i \text{ and } r_{ij} < K\}. \tag{2}$$

In our case, we use $K = 8$. Because the current token is visible to itself and has a distance of zero, it has a within-type distance rank of zero and is always retained.

For example, consider the token `3` in the prompt `1+2+3=`. The two occurrences of `+` receive a consecutive within-type distance rank of one and zero respectively.

Reference scores. We project the input embedding of every token type $e(v)$ into a normalised type representation:

$$\rho(v) = \frac{W_{\text{type}} e(v)}{\|W_{\text{type}} e(v)\|_2}, \qquad W_{\text{type}} \in \mathbb{R}^{d_r \times d_e}, \tag{3}$$

For a candidate $j \in \mathcal{A}_i$, write r_{ij} for its within-type distance rank and σ_{ij} for its side relative to token i (left, self, or right). Its score is:

$$\ell_{i,j} = b^{\sigma_{ij}}_{r_{ij}} + \left\langle w^{\sigma_{ij}}_{r_{ij}}, \rho(x_j) \right\rangle, \tag{4}$$

with learned parameters $b^{\sigma_{ij}}_{r_{ij}} \in \mathbb{R}$ and $w^{\sigma_{ij}}_{r_{ij}} \in \mathbb{R}^{d_r}$ associated with the rank and side pair.

Reference weights. Ordinary softmax gives types with more candidate occurrences more total weight when scores are equal. We remove this multiplicity advantage by dividing each exponentiated score by its type's retained count, $|\mathcal{A}_i(v)|$:

$$\alpha_{i,j} = \frac{\exp(\ell_{i,j})/|\mathcal{A}_i(x_j)|}{\sum_{a \in \mathcal{A}_i} \exp(\ell_{i,a})/|\mathcal{A}_i(x_a)|}, \qquad j \in \mathcal{A}_i. \tag{5}$$

Then equal scores allocate equal total weight to each available type.

3.3 REFERENCE-RELATIVE ROTARY POSITIONS

ReRoPE constructs each token's positional transformation by mixing the rotations proposed by its candidate references. We represent each two-dimensional query or key pair as a complex number, so RoPE rotations (Su et al., 2024) act by complex multiplication. For RoPE frequency ω_f, the weighted mixture (Figure 2) is:

$$z_{i,f} = \sum_{j \in \mathcal{A}_i} \alpha_{i,j} \exp(\mathrm{i}\omega_f g(d_{i,j})) \,, \tag{6}$$

where i is the imaginary unit. The map g leaves distances unchanged when augmentation is disabled, $g(d) = d$; its training-time randomisation is described in Section 3.4. The same reference weights $\alpha_{i,j}$ are used across all frequencies.

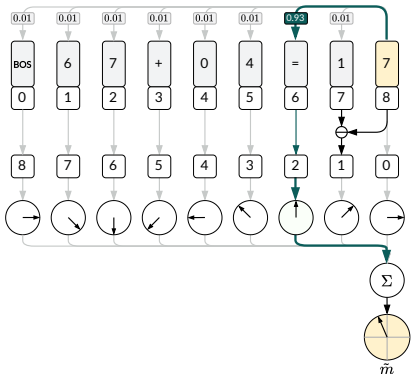

Figure 2: **Mixing reference-relative rotations.** Each candidate reference proposes a RoPE rotation determined by its distance from the current token. ReRoPE combines these rotations using the learned candidate probabilities. The resulting operator is applied to queries and keys in every decoder layer.

Normalising each frequency plane. A convex mixture of rotations is generally a scaled rotation in each two-dimensional RoPE frequency plane, which can unintentionally attenuate queries and keys. Fix a token and a frequency plane, and let z denote its mixed phasor. For $z \neq 0$, we project it onto the unit circle:

$$\widehat{z} = \frac{z}{|z|}. \tag{7}$$

We apply this operation independently to every token and frequency plane. The resulting unit-magnitude phasors define norm-preserving rotations. One-hot reference weights recover ordinary RoPE at the selected reference-relative coordinate, including its randomly remapped value during augmentation. Soft weights mix rotations rather than coordinates, so the resulting transformation need not correspond to RoPE at any single scalar coordinate.

Applying the mixtures. Continuing with the same token and frequency plane, let q and k denote the corresponding two-dimensional query and key components, represented as complex numbers. We apply the normalised phasor to both:

$$q' = \widehat{z}q, \qquad k' = \widehat{z}k. \tag{8}$$

This operation is performed independently in every frequency plane. ReRoPE replaces the backbone's native positional encoding.

3.4 Random-gap positional augmentation

With fixed coordinates, the decoder may learn to compensate for a positional mismatch rather than correct the reference weights. For example, in `12+34=`, referencing `+` for both `2` and `4` gives coordinates `1` and `2`, and this mismatch changes with operand length. To make fixed-offset compensation less reliable, we randomise the gaps between coordinates assigned to successive reference distances (Figure 3).

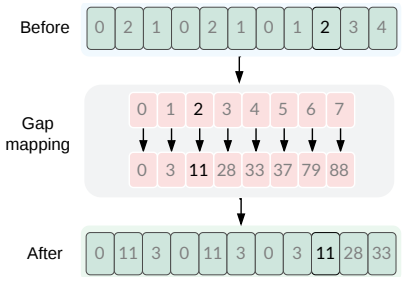

Figure 3: **Random gaps.** For each training example, canonical reference-relative distances d are mapped through a newly sampled, strictly increasing integer chart $g(d)$.

Randomising the gaps. Let D bound the reference distances encountered during training. For each example, we sample $D+1$ strictly increasing non-negative integers starting at zero and assign them to distances $0, \ldots, D$ in order. This defines a map $g : \{0, \ldots, D\} \to \mathbb{Z}_{\geq 0}$,where

$$0 = g(0) < g(1) < \cdots < g(D), \tag{9}$$

which preserves equality and order while varying the spacing between distances. During training, we use

$g(d_{i,j})$ in Equation 6. A fresh map is sampled independently for each example on every training forward pass and shared across its tokens, candidates, frequencies, heads, and layers.

For fixed inputs and parameters, resampling g changes only the rotary coordinates, not token order, candidates, or coupler weights. Unlike RandPE's augmentation of surface-token indices (Ruoss et al., 2023), ReRoPE remaps reference-relative distances. Its gaps can vary within a single example, rather than scaling all rotary angles by a shared factor (Cho et al., 2024).

Effect on positional matching. When each token selects a single reference, equal reference distances yield identical rotations across all sampled maps. Their positional factors, therefore, cancel in the query-key dot product. An off-by-one mismatch instead produces a coordinate difference $g(d+1) - g(d)$, which varies across augmentations rather than remaining a fixed unit offset. A shared shift of all coordinates would not have this effect, since it leaves the differences between them unchanged.

For soft reference weights, two tokens have identical mixtures under every shared map if they assign the same total weight to each distance.

Burn-in phase. We apply random-gap augmentation only during an initial burn-in phase early on in training, since useful computations need not rely exclusively on exact matches. We sample from a range of integers whose maximum follows a gradually increasing curriculum.

4 Experiments

Our experiments examine whether learned reference-relative positions support length generalisation without positional supervision (**Q1**), which components contribute to this generalisation (**Q2**), and how performance changes with decoder depth (**Q3**). We compare ReRoPE with alternative positional encodings for addition, copy, reverse, and parity, and conduct extended-length and component studies on addition.

4.1 Experimental setup

Tasks and input formats. Following related literature (Delétang et al., 2023; Zhou et al., 2024a; Cho et al., 2024; 2025), we study addition, copy, reverse, and parity. Every example is represented as a decoder-only sequence of the form `query=response`, with individual symbols treated as tokens. Training uses teacher forcing and applies loss only to response tokens. Decoder self-attention remains causal. Wherever possible, we follow the formatting used by Position Coupling Cho et al. (2024; 2025) to isolate the effect of ReRoPE.

Addition. Two-operand addition. The shorter operand is left-padded with zeros to match the longer one. The sum is left-padded to one digit beyond this common width to accommodate a possible final carry, then reversed digitwise so that autoregressive decoding proceeds from the least to the most significant digit (Lee et al., 2024; Cho et al., 2024). We train on operand lengths 1–30 and evaluate on seen (1–30), medium (31–100), and long (101–200) lengths.

Copy and reverse. Both tasks take a digit string as input. Copy reproduces the string unchanged, whereas reverse returns its digitwise reversal. We train both tasks on input lengths 1–40 and evaluate all models on the same set of lengths spanning 1–200.

Parity. The input is a binary string, and the response gives the running XOR state after each input bit. We train on input lengths of 1–20 bits and evaluate on lengths of 1–100 bits.

4.2 Implementation details

Unless stated otherwise, all methods use a two-layer, eight-head decoder-only Transformer with $d_{\text{model}} = 1024$, per-head query/key dimension 128, and feed-forward dimension 2048. The backbone uses RMSNorm, gated-GELU feed-forward blocks, and no dropout. Each run uses a fixed

dataset of one million generated training examples. Training lasts 50,000 optimiser updates with an effective batch size of 1,000, corresponding to 50 million example presentations.

We use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, no weight decay, and gradient clipping at 1.0. Selector parameters use learning rate 3×10^{-4}, while positional-consumer and decoder parameters use 3×10^{-5}. A cosine schedule uses a 1% warm-up and decays to 10% of the initial learning rate. The main comparisons use three model/data seed pairs.

The default ReRoPE selector uses $K = 8$. RoPE uses base $\theta = 10{,}000$. Every observed non-padding token type is eligible to provide references.

We use a random gap burn-in phase of $20,000$ iterations. We use a curriculum over the maximum sampled integer in the mapping, respectively rising from 256 at the start of training to 2048 at $10,000$ iterations.

4.3 Evaluation Metrics

Our primary metric is autoregressive exact-match accuracy: an example is correct only if its entire generated response equals the target. All reported evaluations use the final training checkpoint and greedy autoregressive decoding.

Curves show the pointwise median across three model/data seed pairs, while shaded regions span the pointwise minimum and maximum across those seeds. Scalar results in the text are reported separately as mean $\pm$ sample standard deviation. Component ablations are reported as single-seed experiments.

4.4 Main results

We compare ReRoPE with supervised Position Coupling using learned absolute embeddings (Cho et al., 2024), standard RoPE (Su et al., 2024), RandPE–RoPE (Ruoss et al., 2023), and FIRE (Li et al., 2024). The primary comparisons use matched tokenisation, decoder dimensions, training-set size, number of optimiser updates, and autoregressive evaluation. Position Coupling uses task-specific position labels.

Length generalisation across tasks. Figure 4 reports exact-match accuracy as sequence length increases. At the maximum primary evaluation length, ReRoPE matches the success of position coupling and obtains $99.97\% \pm 0.06\%$ exact-match accuracy on 200-digit addition and $100.00\% \pm 0.00\%$ on copy and reverse at length 200 and parity at length 100. RoPE, RandPE–RoPE, and FIRE obtain 0% mean exact-match accuracy at these maximum lengths. The results showcase the potential of having learnable reference-relative positional embeddings for broader out-of-distribution generalisation.

Extended addition evaluation. Figure 1 reports a dedicated extended-length addition comparison using 100 generated examples per length and seed. At 1,000 digits, the default ReRoPE achieves $99.67\% \pm 0.58\%$ exact-match accuracy after training only on 1–30 digit operands. The Position Coupling configuration trained for this extended comparison obtains $0.00\% \pm 0.00\%$ at the same length. We hypothesise that such broader generalisation is possible due to the random-gap augmentation.

4.5 Depth and component analysis

Table 1 reports controlled single-seed ablations on addition. Additionally, Figure 5 shows the decoder depth analysis.

Decoder depth. We vary decoder depth from three to six layers while holding width, head dimension, training data, and optimisation fixed. The results are depicted in Figure 5. At length 200, the three-, four-, five-, and six-layer models achieve $100.00\% \pm 0.00\%$, $99.80\% \pm 0.26\%$, $100.00\% \pm 0.00\%$, and $100.00\% \pm 0.00\%$, respectively. At 1,000 digits, their corresponding accuracies are $90.67\% \pm 15.31\%$, $64.67\% \pm 44.60\%$, $55.33\% \pm 48.43\%$, and $87.67\% \pm 18.01\%$. Increasing depth, therefore, does not remove extrapolation through length 200. Much farther outside the train-

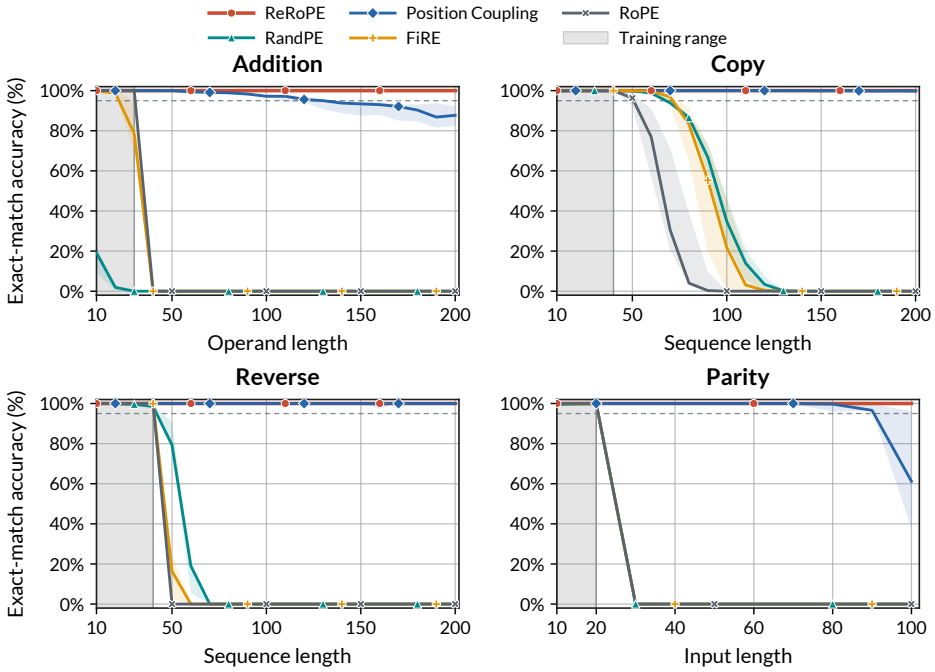

Figure 4: **ReRoPE extrapolates across four algorithmic tasks.** Exact-match accuracy under greedy autoregressive generation for addition, copy, reverse, and parity. All methods use matched 2-layer, 8-head decoders and are evaluated from their final checkpoints. Models are trained on lengths up to 30 for addition, 40 for copy and reverse, and 20 for parity; grey regions mark these training ranges. Curves show the pointwise median across three model/data seed pairs, and shaded regions span the pointwise seed range. The horizontal dashed line marks 95% exact-match accuracy.

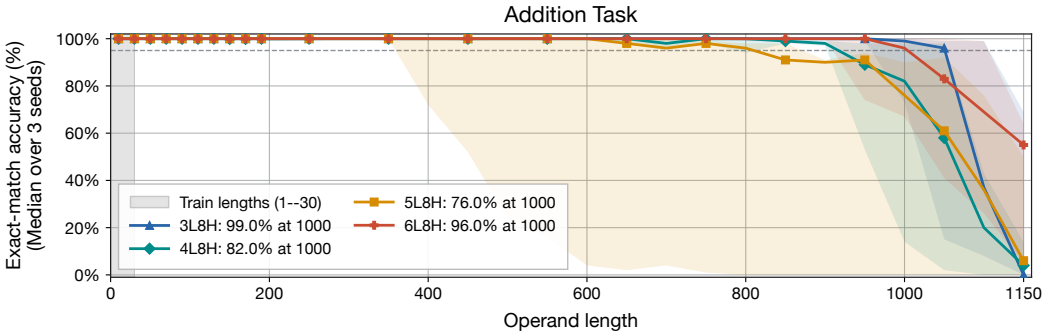

Figure 5: **ReRoPE remains effective as decoder depth increases.** Exact-match accuracy on addition for 3-, 4-, 5-, and 6-layer decoders trained on operands of 1–30 digits. Lines show the pointwise median across three seeds, and shaded regions span the seed range. All depths generalise reliably through 200 digits, while the majority of configurations remain accurate near 1,000 digits.

ing range, performance is non-monotonic and variable across runs, but still tends to significantly outperform position coupling, even at the worst seed.

Random-gap augmentation. The full configuration obtains 99.95% exact-match accuracy at length 200. Removing random gaps preserves 100% accuracy at the training boundary but reduces accuracy to 2.15% at length 100 and 0% at lengths 150 and 200. Replacing non-uniform random gaps with a mechanism that multiplies the original position of every token by a number between 1 and 5 also obtains 100% at length 30 but 0% from length 100 onward. Non-uniform random gaps are therefore necessary for extrapolation in this tested setting.

Configuration	30 (ID)	60	100	150	200
Full ReRoPE ($K = 8$)	100.00	100.00	99.95	100.00	99.95
Canonical coordinates	100.00	66.15	2.15	0.00	0.00
PID multiplier 1–5	100.00	72.50	0.00	0.00	0.00
Learned APE, residual stream	100.00	0.00	0.00	0.00	0.00
Learned APE, Q/K each layer	100.00	0.00	0.00	0.00	0.00
No type-cardinality correction	98.35	0.00	0.00	0.00	0.00
No lexical-role term	88.55	0.00	0.00	0.00	0.00
No self candidate	100.00	94.75	98.50	98.55	99.00
$K = 4$	100.00	100.00	100.00	100.00	100.00
$K = 16$	100.00	100.00	100.00	100.00	99.80
$K = 24$	100.00	100.00	100.00	100.00	100.00

Table 1: Exact-match accuracy (%) for controlled single-seed addition ablations. Models are trained on operand lengths 1–30, and each reported length uses 2,000 generated examples. Each row changes only the indicated component. All ablations use matched 2-layer, 8-head decoders.

Why RoPE? Replacing the mixtures of RoPE Su et al. (2024) with a mixture of learned absolute positional embeddings, added either to the residual stream or directly to queries and keys in every layer, obtains 100% at the training boundary but 0% at every reported out-of-range length. These runs reached high token-match accuracy, but failed to get entire instances correct. RoPE behaved much more consistently in this regard.

Reference selection. Removing type-cardinality correction (division by $|\mathcal{A}_i|$ in Equation 5) or the lexical-role scoring term (removing the second term in Equation 4) reduces training-boundary accuracy to 98.35% and 88.55%, respectively, and produces 0% at the reported out-of-range lengths. Since these variants also fit the training range less well, they do not isolate an extrapolation-specific effect. Excluding self-reference retains 99.00% accuracy at length 200, showing that self-candidates are not indispensable in this run. Values of $K \in \{4, 16, 24\}$ all obtain at least 99.80% at length 200, indicating that the result is not restricted to the default candidate budget.

5 Limitations

Our evaluation is restricted to synthetic algorithmic tasks with small vocabularies and controlled sequence formats. The results do not establish applicability to natural-language modelling, large-vocabulary settings, or unstructured long-context reasoning. Removing position labels also does not remove every representational assumption: useful token occurrences must be available as references, and repeated symbols, separators, padding, and consistent formatting make such references accessible in the tasks studied here. The extended-depth experiments also exhibit substantial run-to-run variability.

Finally, reference selection is bidirectional within the fixed observed prompt. Decoder attention remains causal, and response references remain prefix-causal, but a prompt token's positional representation may depend on a later prompt token. The current formulation, therefore, does not directly cover streaming prompts without changing reference selection and cache updates.

6 Conclusion

An open question in the community is how to improve current models to learn algorithmic behavior from data. Existing work has shown that computation-aligned positional encoding, usually provided to the model through direct supervision, is a step in the right direction, greatly improving the ability of models to extrapolate to larger size problems. In this work we took one step further, showing that such positional encoding can be learned directly from data and next token prediction objective. ReRoPE achieves near-perfect accuracy on $1{,}000$-digit operands after training on lengths 1–30 exceeding the extrapolation range of previous works. While many questions remain open, we believe this offers a potential direction for future work and re-emphasizes how contextual and flexible positional encoding mechanisms can be the key for learning structural computations directly from data.

AI USE STATEMENT

In this work, we used generative AI tools to assist with method implementation and testing, and the interpretation of experimental diagnostics and results. We have not used generative AI tools to generate datasets, formulate or prove theoretical results, perform translation, process datasets, or conduct qualitative analysis; the remaining required disclosure tasks are not applicable to this work. The research hypothesis and core method were developed by the authors.

Additionally, we used generative AI tools for plotting code, literature search and summarisation, and editing the manuscript and figure captions. We have reviewed all AI-assisted work, tested the implementations, verified the reported results, and checked claims and citations against primary sources. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Cem Anil, Yuhuai Wu, Anders Johan Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Venkatesh Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. Exploring length generalization in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=zSkYVeX7bC4.

Pranjal Awasthi and Anupam Gupta. Improving length generalization in transformers via task hinting, 2024. URL https://openreview.net/forum?id=oTl1ABwM4n.

Hanseul Cho, Jaeyoung Cha, Pranjal Awasthi, Srinadh Bhojanapalli, Anupam Gupta, and Chulhee Yun. Position coupling: Improving length generalization of arithmetic transformers using task structure. In *Advances in Neural Information Processing Systems 37*, 2024.

Hanseul Cho, Jaeyoung Cha, Srinadh Bhojanapalli, and Chulhee Yun. Arithmetic transformers can length-generalize in both operand length and count, 2025. URL https://arxiv.org/abs/2410.15787.

Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, and Pedro A. Ortega. Neural networks and the chomsky hierarchy, 2023. URL https://arxiv.org/abs/2207.02098.

Olga Golovneva, Tianlu Wang, Jason Weston, and Sainbayar Sukhbaatar. Contextual position encoding: Learning to count what's important, 2024. URL https://arxiv.org/abs/2405.18719.

Noah Golowich, Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. The role of sparsity for length generalization in llms. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 19809–19840, 2025.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 24892–24928. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/4e85362c02172c0c6567ce593122d31c-Paper-Conference.pdf.

Nayoung Lee, Kartik Sreenivasan, Jason D. Lee, Kangwook Lee, and Dimitris Papailiopoulos. Teaching arithmetic to small transformers. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=dsUB4bst9S.

Philip Heejun Lee. Position as probability: Self-supervised transformers that think past their training for length extrapolation, 2025. URL https://arxiv.org/abs/2506.00920.

Shanda Li, Chong You, Guru Guruganesh, Joshua Ainslie, Santiago Ontanon, Manzil Zaheer, Sumit K. Sanghai, Yiming Yang, Sanjiv Kumar, and Srinadh Bhojanapalli. Functional interpolation for relative positions improves long context transformers. *ArXiv*, abs/2310.04418, 2024. URL https://api.semanticscholar.org/CorpusID:263829377.

Sean Michael McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, and Tom Goldstein. Transformers can do arithmetic with the right embeddings. In *Proceedings of the Thirty-Eighth Annual Conference on Neural Information Processing Systems*, 2024.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. Investigating the limitations of the transformers with simple arithmetic tasks. *CoRR*, abs/2102.13019, 2021. URL https://arxiv.org/abs/2102.13019.

Ofir Press, Noah Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=R8sQPpGCv0.

Anian Ruoss, Grégoire Delétang, Tim Genewein, Jordi Grau-Moya, Róbert Csordás, Mehdi Bennani, Shane Legg, and Joel Veness. Randomized positional encodings boost length generalization of transformers. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1889–1903, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-short.161. URL https://aclanthology.org/2023.acl-short.161/.

Ruoqi Shen, Sébastien Bubeck, Ronen Eldan, Yin Tat Lee, Yuanzhi Li, and Yi Zhang. Positional description matters for transformers arithmetic, 2023. URL https://arxiv.org/abs/2311.14737.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomput.*, 568(C), February 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, 2017.

Petar Veličković, Christos Perivolaropoulos, Federico Barbero, and Razvan Pascanu. Softmax is not enough (for sharp size generalisation), 2025. URL https://arxiv.org/abs/2410.01104.

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Joshua M. Susskind, Samy Bengio, and Preetum Nakkiran. What algorithms can transformers learn? a study in length generalization. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=AssIuHnmHX.

Yongchao Zhou, Uri Alon, Xinyun Chen, Xuezhi Wang, Rishabh Agarwal, and Denny Zhou. Transformers can achieve length generalization but not robustly, 2024b. URL https://arxiv.org/abs/2402.09371.

Jiajun Zhu, Peihao Wang, Ruisi Cai, Jason D. Lee, Pan Li, and Zhangyang Wang. Rethinking addressing in language models via contextualized equivariant positional encoding. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 80155–80186, 2025.