Algebraic Positional Encodings

Konstantinos Kogkalidis[1,2]
kokos.kogkalidis@aalto.fi

Jean-Philippe Bernardy[3,4]
jean-philippe.bernardy@gu.se

Vikas Garg[1,5]
vgarg@csail.mit.edu

[1]Aalto University
[2]University of Bologna
[3]University of Gothenburg
[4]Chalmers University of Technology
[5]YaiYai Ltd

Abstract

We introduce a novel positional encoding strategy for Transformer-style models, addressing the shortcomings of existing, often *ad hoc*, approaches. Our framework implements a flexible mapping from the algebraic specification of a domain to a positional encoding scheme, where positions are interpreted as orthogonal operators. This design preserves the structural properties of the source domain, thereby ensuring that the end-model upholds them. The framework can accommodate various structures, including sequences, grids and trees, but also their compositions. We conduct a series of experiments demonstrating the practical applicability of our method. Our results suggest performance on par with or surpassing the current state of the art, without hyper-parameter optimizations or "task search" of any kind. Code is available through https://aalto-quml.github.io/ape/.

1 Introduction

Attention-based models inheriting from the Transformer [Vaswani et al., 2017] have become ubiquitous in neural computation, supplanting the go-to models of the last decade and driving a continuous stream of breakthroughs across diverse domains. Their success is perhaps at odds with the Transformer's structural lenience. Its key building block, dot-product attention, is by default unable to perceive and utilize the structure and arrangement of the input/output tokens being processed. To address this limitation, a plethora of works have sought to endow Transformers with appropriate inductive biases. The default strategy is to adjust token representations via so-called *positional encodings*; vector operations that hint at the structure being modeled. Nonetheless, most positional encoding schemes to date are either empirically motivated, or tailored to specific tasks. This renders their theoretical evaluation challenging, and hinders any prospects of a unifying framework.

In this study, we seek to fill this gap with a theory-first approach. Through the lens of group theory, we scrutinize some of the most commonly targeted data structures, and express them by means of inductive definitions that reveal and explicate their structural properties. Leveraging this analysis, our modeling strategy invokes a homomorphic interpretation that maps each domain into **algebraic positional encodings** (APE): attention-compatible vector operations parameterizing (subgroups of) the orthogonal group. In the sequential context, algebraic positional encodings streamline the widely adopted rotary encodings of Su et al. [2023], while also offering clear theoretical insights on their success. More importantly, algebraic positional encodings naturally extend to non-sequential domains, such as κ-ary trees and multidimensional regular grids, paving the way for a simple and elegant methodology for interpretable and domain-general structurally-refined Transformers. We carry out an experimental evaluation in settings that allow for reproducible and statistically sound conclusions. Across the tasks considered, algebraic positional encodings consistently and significantly outperform strong baselines at an aggregate level, providing initial but compelling evidence that they constitute not just a *sensible meta-theory* for positional encodings, but also an *actionable alternative* to the current state of the art.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

§2 BACKGROUND
We introduce the problem (§2.1) and the vocabulary of the solution (§2.2).

§3 THEORY
We provide an algebraic characterization of positions in the context of different ambient structures. We frame algebraic positional encodings as structure-preserving semantic interpretations, and present reference implementations. Concretely:

- §3.1 *Sequences* are an isomorphism of the free group $\langle \mathbb{1} \rangle$ (*i.e.*, the integers, $\mathbb{Z}$), and can be interpreted as a single generator subgroup of the orthogonal group $O(d)$.
- §3.2 *Rotary Positional Encodings* correspond to a (quite literally) special case of this interpretation: $SO(d)$.
- §3.3 *k-ary Trees* are an isomorphism of the finitely generated group $\langle \mathbb{1}, \mathbb{2} \dots \kappa \rangle$, and can be interpreted as a finitely generated subgroup of $O(d)$.
- §3.4 *Regular Grids* are the group direct sum of multiple sequences. They can be interpreted as the matrix direct sum of their components' interpretations.
- §3.5 *Extensions* can be obtained in multiple directions.

§4 PRACTICE
We carry out fair and replicable experiments across all three structures analyzed, namely *Sequence Transduction* (§4.1), *Tree Transduction* (§4.2) and *Image Recognition* (§4.3), and find that algebraic positional encodings consistently match or outperform alternative schemes in the tasks considered (§4.4).

§5 RELATED WORK
We position our work in the context of the broader literature.

§6 LIMITATIONS
We close with a brief discussion of possible limitations and potential future improvements.

Table 1: A summary of this paper.

2 Background

2.1 The Problem with Dot-Product Attention

All transformer variants employ some variation of the multi-head scaled dot-product attention mechanism of Vaswani et al. [2017]. For each attention head, the dot-product attention between queries $\boldsymbol{X} \in \mathbb{R}^{m \times d}$ and keys $\boldsymbol{Y} \in \mathbb{R}^{n \times d}$ is defined as:

$$\mathrm{atn}(\boldsymbol{X}, \boldsymbol{Y}) := \mathrm{softmax}_{(n)}\left(\frac{(\boldsymbol{X}\boldsymbol{\Phi}^{(q)})(\boldsymbol{Y}\boldsymbol{\Phi}^{(k)})^\top}{\sqrt{d}}\right)\boldsymbol{Y}\boldsymbol{\Phi}^{(v)} \tag{1}$$

In equation (1), matrices $\boldsymbol{\Phi}^{(q)}, \boldsymbol{\Phi}^{(k)}, \boldsymbol{\Phi}^{(v)} : \mathbb{R}^{d \times d}$ enact linear functions, applied point-wise (broadcasted) across all m and n entries of $\boldsymbol{X}$ and $\boldsymbol{Y}$. The dot-product term $(\boldsymbol{X}\boldsymbol{\Phi}^{(q)})(\boldsymbol{Y}\boldsymbol{\Phi}^{(k)})^\top$ contains unnormalized attention scores in the Cartesian product of queries and keys. Unmodified, dot-product attention is permutation *invariant* with respect to its second argument; that is, for any arbitrary permutation $p_n \in \mathcal{S}_n$:

$$\mathrm{atn}(\boldsymbol{X}, \boldsymbol{Y}) \equiv \mathrm{atn}(\boldsymbol{X}, p_n(\boldsymbol{Y})) \tag{2}$$

Unless one is dealing with orderless structures like multisets or fully connected graphs, this property is generally undesirable. The lack of structural biases is traditionally counteracted by the component-wise addition of unidimensional periodic signals of varying frequencies. These, however, often prove inadequate in data-scarce domains, where extensive pretraining is impossible, and structure-rich domains, where a sequence-of-tokens projection is too radical of a simplification.

2.2 Recap on Group Theory

To address this issue, we propose an algebraic treatment of positional encodings, based on principles lent from group theory. For the sake of convenience and accessibility, we provide a brief recap of the notions of interest here. A *group* G consists of a set of *elements* and a *binary operation* $(\_\cdot\_)$ satisfying four fundamental laws:

- The group is *closed* under the the group operation. For all a, b in G, $a \cdot b$ is also in G.
- The group operation is *associative*. For all a, b, c in G, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
- The group operation has an *identity* element e, such that for all a in G, $a \cdot e = e \cdot a = a$.
- Each group member has an *inverse*. For all a in G, there exists some element $\overline{a}$ such that $a\overline{a} = \overline{a}a = e$, where e is the identity element.

A group is characterized as *finite* or *infinite* depending on the number of elements it has. If all elements of a group G can be expressed as a combination of a subset S of the group elements (combined by means of the group operation, applied either on the elements themselves or on their inverses), we write $G = \langle S \rangle$. We say that G is *generated* by S, and we call the elements of S the *generators* of G. A group with a single generator is called *cyclic*.

3 The Algebra(s) of Positions

Our objective is to establish a framework that offers general and extensible *semantics* for positions across various structures – what we commonly encounter in the literature as *positional encodings*. Most existing proposals adopt a rather parochial stance, relying on maneuvers or heuristics tailored to specific applications and driven, predominantly, by extensive empirical investigations. As such, they fall short with respect to accommodating or reflecting the properties of the underlying structure. We follow a different approach. We adopt Montague's perspective, succinctly paraphrased as:

> "*syntax is an algebra, semantics is an algebra, and meaning is a homomorphism between them*" [Janssen, 2014].

We begin by noting that "positions" do not exist in isolation, but only in the context of some underlying ambient structure. We contend that reasonable positional encodings (*semantics*) may only be reliably obtained by taking into account exactly this structure, its formation rules and properties (*syntax*), and then applying an appropriate interpretation (*meaning*). This is *not* just an academic exercise: a careful syntactic specification is a prerequisite if we aim for semantics that adhere to certain properties, which is arguably preferable to searching for these properties in the wild.

3.1 Sequences

Syntax We start from the simplest structure, and incidentally also the most standard one: the sequence. The full range of positions a token can occupy within a sequence coincides exactly with the naturals, $\mathbb{N}$. Relative paths $\mathbb{P}$ between any two positions can then be seen as the integers, $\mathbb{Z}$, with positive (resp. negative) numbers denoting forward (resp. backward) offsets. Using this insight, it is handy to inspect how the standard inductive definition of the integers provides the building blocks for path formation. We start with two constants: the empty path ($\mathbb{0}$), which relates any given point to itself, and the unit path ($\mathbb{1}$), which relates any point to its immediate next. We may compose simple paths into complex ones with the aid of a binary operation $+_{\mathbb{P}}$. This already suffices to specify all forward offsets. In order to construct backward offsets, we need a unary operation $(-)_{\mathbb{P}}$, such that $-\rho$ denotes the inverse of ρ. We can summarize the above by the grammar:

$$\mathbb{P} := \mathbb{0} \mid \mathbb{1} \mid \mathbb{P} +_{\mathbb{P}} \mathbb{P} \mid -\mathbb{P} \tag{3}$$

For this to make sense, the operations must be *coherent*; that is, all ways to start from point ρ_1 and end up in point ρ_2 should be equivalent, even if apparently distinct. The needed equivalences exactly correspond to the group laws, with closure internalized by the inductive definition of (3):

$$(\rho_1 +_{\mathbb{P}} \rho_2) +_{\mathbb{P}} \rho_3 = \rho_1 +_{\mathbb{P}} (\rho_2 +_{\mathbb{P}} \rho_3) \tag{L1}$$

$$\rho +_{\mathbb{P}} \mathbb{0} = \rho = \mathbb{0} +_{\mathbb{P}} \rho \tag{L2}$$

$$\rho +_{\mathbb{P}} (-\rho) = \mathbb{0} \tag{L3}$$

The (unsurprising) insight here is that paths in a sequence form a free group, generated by a single generator ($\mathbb{1}$) – the uniqueness of the generator exceptionally also makes the group abelian (*i.e.*, commutative). For convenience, we adopt the notational shorthand $\mathbb{1}^p$, where:

$$\mathbb{1}^p := \begin{cases} \underbrace{\mathbb{1} +_{\mathbb{P}} \cdots +_{\mathbb{P}} \mathbb{1}}_{p} & p \geq 0 \\ \underbrace{(-\mathbb{1}) +_{\mathbb{P}} \cdots +_{\mathbb{P}} (-\mathbb{1})}_{-p} & p < 0 \end{cases} \tag{4}$$

Semantics The syntactic specifications of the previous paragraph impose constraints on the candidate semantic targets. Among these candidates, we isolate and focus on $\langle \boldsymbol{W} \rangle$, the subgroup of the orthogonal group $O(d)$ that is generated by a single orthogonal matrix $\boldsymbol{W}$. This semantics is not only sound[1] with respect to the structure under scrutiny, but also a familiar object in machine

[1] It is also complete except for the odd case where $\boldsymbol{W}^p = \boldsymbol{I}$ for some p. In practice, this kind of periodic behaviour does not arise randomly, and we can think of $\langle \boldsymbol{W} \rangle$ as being *isomorphic* to $\mathbb{P}$.

learning literature [Arjovsky et al., 2016, Bernardy and Lappin, 2022, *inter alia*]. Note that for $\langle \boldsymbol{W} \rangle$, the group axioms are obtained for free from the orthogonal group, and the additional requirement of commutativity is again satisfied by the uniqueness of the generator.

To illustrate the correspondence between the two structures (and at risk of being pedantic), we spell out the homomorphism $\lceil . \rceil$, which maps paths $\mathbb{P}$ to elements of $\langle \boldsymbol{W} \rangle$, and path operations to operations on orthogonal matrices of size d. For the primitives, we have $\lceil \mathbb{0} \rceil := \boldsymbol{I}_d$ and $\lceil \mathbb{1} \rceil := \boldsymbol{W}$. Path composition amounts to matrix multiplication, *i.e.*, $\lceil \rho_1 +_{\mathbb{P}} \rho_2 \rceil := \lceil \rho_1 \rceil \lceil \rho_2 \rceil$, while path inversion corresponds to matrix transposition, *i.e.*, $\lceil -\rho \rceil := \lceil \rho \rceil^{-1} \equiv \lceil \rho \rceil^\top$. The fact that orthogonal matrices form a group under multiplication is folklore; one can easily verify that the group laws hold also for the semantics.[2]

Implementation In practice, we have $\lceil \mathbb{1}^p \rceil \mapsto \boldsymbol{W}^p$; a norm-preserving bilinear form $\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ which can be used to mediate the dot-product between a query q and a key k offset by a relative distance of p. The representation of all paths up to length p can thus be implemented as a matrix collection $[\boldsymbol{W}^0, \dots, \boldsymbol{W}^p]$, which can asymptotically be obtained using $\mathcal{O}(\lceil \log_2(p) \rceil)$ matrix products (of exponentially larger matrices), and taking up the storage space equivalent of (pd^2) floats. Transposed, the same matrices also serve to represent backwards paths $[\boldsymbol{W}^{-p}, \dots, \boldsymbol{W}^0]$. Storing the representations of all relative paths between queries and keys in a tensor $\boldsymbol{T} : \mathbb{R}^{m \times n \times d \times d}$, we may then substitute the dot-product term of equation (1) for the tensor contraction:

$$\sum_{\alpha,\beta} \boldsymbol{X}_{m\alpha} \boldsymbol{\Phi}^{(q)}_{\alpha\beta} \boldsymbol{T}_{mn\beta\gamma} \boldsymbol{Y}_{n\delta} \boldsymbol{\Phi}^{(k)}_{\delta\gamma} \tag{5}$$

Albeit transparent, this reduction strategy is computationally unappealing due to the doubly quadratic nature of $\boldsymbol{T}$. We can do better by noting that $\boldsymbol{T}_{mn}$ is (definitionally) equal to:

$$\boldsymbol{T}_{mn\alpha\beta} = \sum_{\gamma} \boldsymbol{A}^{(X)}_{m\gamma\alpha} \boldsymbol{A}^{(Y)}_{n\gamma\beta} \tag{6}$$

where $\boldsymbol{A}^{(X)}$ and $\boldsymbol{A}^{(Y)}$ are the matrices containing representations for the *absolute* positions of the entries in $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively. Concretely, a single relative representation is built by composing the *inverted* representation of the source with the representation of the target. Intuitively, each query follows the path that takes it *back* to the origin, which then allows it to directly combine with each forward-offset key; see Figure 1a for a visual example. This insight allows us to keep the memory footprint of equation (1) unchanged, replacing expression (5) with:

$$\sum_{\alpha,\beta,\gamma,\delta,\epsilon} \boldsymbol{X}_{m\alpha} \boldsymbol{\Phi}^{(q)}_{\alpha\beta} \boldsymbol{A}^{(X)}_{m\gamma\beta} \boldsymbol{A}^{(Y)}_{n\gamma\delta} \boldsymbol{Y}_{n\epsilon} \boldsymbol{\Phi}^{(k)}_{\epsilon\delta} \tag{7}$$

This version decomposes the tensor contraction into two matrix multiplications, essentially transforming (rotating or reflecting) the entries of $\boldsymbol{X}$ and $\boldsymbol{Y}$ independently according to their positions.

3.2 Intermezzo: Equivalence with RoPE

The story so far should be reminiscent of the rotary positional encoding scheme of Su et al. [2023, RoPE]. Not unlike our approach, RoPE substitutes the vanilla dot-product for a position-dependent bilinear form. Underlying the form is a $d \times d$-dimensional matrix $\boldsymbol{R}$ with a block-diagonal structure, where each 2×2-sized block corresponds to a rotation matrix that acts on a 2-dimensional subspace of $\mathbb{R}^d$. These independent rotations are parameterized by a (fixed) set of base angles $\Theta := [\theta_1, \dots, \theta_{d/2}]$. To incorporate position-dependence, *i.e.*, for a query/key pair at a relative distance of p, the base angles are multiplied by p, effectively altering the rotations applied.

At first glance, rotary encodings appear to be under-parameterized, and thus strictly weaker than orthogonal ones. However, any orthogonal matrix $\boldsymbol{W} \in O(d)$ admits a canonical form $\boldsymbol{W} = \boldsymbol{P}\boldsymbol{Q}\boldsymbol{P}^\top$, where $\boldsymbol{P}$ is an orthogonal change of basis, and $\boldsymbol{Q}$ is block-diagonal, with the 2×2-sized blocks being, once again, $2-$dimensional rotation matrices [Murnaghan and Wintner, 1931][3]. Owing to the orthogonality of $\boldsymbol{P}$, raising $\boldsymbol{W}$ to its pth power is equal to $\boldsymbol{P}\boldsymbol{Q}^p\boldsymbol{P}^\top$ (*i.e.*, it leaves the change of basis unaffected). In turn, raising $\boldsymbol{Q}$ to its pth power is equivalent to simply multiplying the rotation

[2]The story is no different for $\boldsymbol{W}$ unitary, with the group structure provided by the unitary group $U(d)$, and path inversion interpreted as the matrix conjugate transpose.

[3]We alert the reader that a *constructive* proof of this decomposition has proven surprisingly difficult to find.

angles of its blocks by p. Finally, given the linearity of the transformations $\mathbf{\Phi}^{(q)}$ and $\mathbf{\Phi}^{(k)}$, their compositions with $\boldsymbol{P}$ are also linear. By identifying $\boldsymbol{Q}$ with RoPE's $\boldsymbol{R}$, we can then see that, for any given collection of angles Θ, APE and RoPE coincide under the substitutions:

$$\mathbf{\Phi}^{(q)}_{\text{RoPE}} = \mathbf{\Phi}^{(q)}\boldsymbol{Q} \quad \text{and} \quad \mathbf{\Phi}^{(k)}_{\text{RoPE}} = \mathbf{\Phi}^{(k)}\boldsymbol{Q} \tag{8}$$

In practical terms, and uniquely for the sequential case, APE *is equivalent to a trainable version of* RoPE*, where the rotation angles Θ may vary and be optimized during training*.[4]

Which of the two parameterizations is preferable is debatable. On the one hand, APE's formulation is FLOP-optimized (being just matrix multiplications), and obviates the need for backpropagating through trigonometric functions (which are periodic, non-monotonic, and prone to gradient instabilities). On the other hand, RoPE's diagonalized form gives access to a memory-efficient contraction that does away with the matrix multiplications of expression (7) altogether; we direct the interested reader to Su et al. [2023, Section 3.4.2] for a reference implementation.[5]

In either case, the equivalence between the two is confined to the *sequential* setup; we will now move on to generalize our strategy to other, *previously inaccessible*, structures.

3.3 Trees

Syntax In the previous section, we characterized the structure of relative paths on a sequence as the free group with one generator, and uncovered a (practically) isomorphic interpretation in the subgroup of orthogonal matrices with a single generator. Upon closer inspection, we note that a sequence can be viewed as a special case of the more general structure of κ-ary branching trees, where the branching factor κ just so happens to be 1. Denoting the more general case as $\mathbb{P}_\kappa$, we must first extend the set of primitives to include all branching options, $\mathbb{1}, \mathbb{2}, \ldots \kappa : \mathbb{P}_\kappa$. Each primitive now denotes a choice of branch (except for $\mathbb{0}$, which is again the empty path). Paths now form a free group with κ distinct generators. The presence of multiple generators means that commutativity no longer holds; $\mathbb{1} +_{\mathbb{P}_\kappa} \mathbb{2}$ is distinct from $\mathbb{2} +_{\mathbb{P}_\kappa} \mathbb{1}$ (the former prescribes a descent down branch $\mathbb{1}$ then branch $\mathbb{2}$, whereas the latter prescribes a descent down branch $\mathbb{2}$ then branch $\mathbb{1}$). Inversion is as before: for every path from each local root to some descendant down the line, there is also an inverse path from that descendant up to its ancestor. Perhaps more interestingly, upwards and downwards paths can be joined, allowing the precise specification of relative paths between any two nodes, even when the two do not share a single line of descent (think nephews, aunts and all other sorts of distant relatives, see Figure 1b for an example). Adjusting grammar (3) accordingly, we have:

$$\mathbb{P}_\kappa := \mathbb{0} \mid \mathbb{1} \mid \mathbb{2} \mid \ldots \mid \kappa \mid \mathbb{P}_\kappa +_{\mathbb{P}_\kappa} \mathbb{P}_\kappa \mid -\mathbb{P}_\kappa \tag{9}$$

with laws L1, L2 and L3 still in effect.

Semantics The interpretation follows along the same lines as before. This time around, however, we cannot make do with a single orthogonal matrix $\boldsymbol{W}$ – we need a collection of κ matrices, one for each branch option. As a consequence, the semantic target is now $\langle \boldsymbol{W}_1, \boldsymbol{W}_2, \ldots \boldsymbol{W}_\kappa \rangle$. Note that the target is no longer commutative (exactly in alignment with the source).

Implementation For a tree structure of depth δ and branching factor κ, let ν denote the number of *unique* absolute positions occupied (upper bound by κ^δ in the case of a complete tree). Their representations can be computed in $\delta\kappa$ steps of parallel matrix-matrix multiplications and a memory cost of νd^2, as follows. First, we can build up a collection of all unique absolute paths, each represented as a (right-padded) word of length δ from the vocabulary of primitives. Their corresponding representations constitute a tensor of size $\nu \times d \times d$, initialized as ν identity matrices. We can then iterate across these words in parallel, one primitive per step (*i.e.*, depth) t, selecting all words that take the same branching direction at the current depth, and right-multiplying their representations by the corresponding orthogonal generator. Finally, absolute paths can be composed into relative ones using the modified dot-product attention of expression (7), just like before.

[4] An alternative reading is that even though orthogonal matrices are generally more expressive than rotation matrices (allowing not just rotations but also reflections), the Transformer's architecture makes up for RoPE's reduced expressivity by supplying a free change of basis through its trainable weights $\mathbf{\Phi}$.

[5] For more practical insights on initializing and parameterizing APE and translating between APE and RoPE, please refer to Appendix A.

3.4 Grids

The generalization from sequences to trees rests on the observation that a sequence is a tree with a deficit of choices. An altogether different axis of generalization can be obtained by recalling that composite groups can be constructed by joining together two or more elementary groups. Moreover, if it just so happens that the original groups were abelian, then so is their composition; in that case, we call the composite a *group direct sum*. This construction provides access to an extension from sequences to multidimensional regular grids.

For the sake of simplicity and without loss of generality, we consider a standard instance of a two-dimensional grid: an image. An image is a collection of pixels (or pixel patches) that inhabit a coordinate system (h, w). Each of h and w is the product of grammar (3), inheriting all path-related notions discussed earlier. Since $\mathbb{P}$ is an abelian group, the coordinate system also constitutes an abelian group $\mathbb{P}^2 := \mathbb{P} \oplus \mathbb{P}$. The new group and inversion operations are $+_{\mathbb{P}^2}$ and $(-)_{\mathbb{P}^2}$, and denote the act of joining and inverting two-dimensional paths, respectively. Both are canonically defined component-wise, on the basis of their one-dimensional counterparts:

$$(x, y) +_{\mathbb{P}^2} (z, w) := (x +_{\mathbb{P}} y, z +_{\mathbb{P}} w) \tag{10}$$

$$-(x, y) := (-x, -y) \tag{11}$$

with $\mathbb{0}^2 := (\mathbb{0}, \mathbb{0})$ as the new neutral element. Intuitively, $+_{\mathbb{P}^2}$ corresponds to vector addition, and $(-)_{\mathbb{P}^2}$ to a reflection about the origin with respect to both axes.

Semantics The specifications above allow us to reuse the notions from Section 3.1 in order to interpret the components and operations of $\mathbb{P}^2$. What is left unspecified is the interpretation of the group elements themselves; that is, we have yet to explicate what an object of $\lceil \mathbb{P} \oplus \mathbb{P} \rceil$ looks like. The quest is a short one; the notion of a direct sum carries over to matrices, and is defined as:

$$\boldsymbol{A} \oplus \boldsymbol{B} := \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B} \end{bmatrix} \tag{12}$$

From this, we get the (rather straightforward) interpretation $\lceil (\rho_1, \rho_2) \rceil \mapsto \lceil \rho_1 \rceil \oplus \lceil \rho_2 \rceil$.

Implementation In practice, we now split the vector space in two independent parts. The first part is modulated by orthogonal matrices from $\langle \boldsymbol{H} \rangle$, and the second part by orthogonal matrices from $\langle \boldsymbol{W} \rangle$. For a query q and a key k that reside at a relative distance of (h, w), their attention score is computed as $q(\boldsymbol{H}^h \oplus \boldsymbol{W}^w)k$ – see Figure 1c for an illustration. Each axis contributes an additive but separable factor to the attention score, forcing the model to learn contextual alignments between token pairs on the basis of their coordinate-wise distances. Not much else is different: we can still compute all matrices in parallel, temporally bound by a logarithmic complexity of $\log_2(\max(h, w))$ and $\max(h, w)(\frac{d}{2})^2$ storage space, given a grid of size (h, w). Subquadratic memory complexity can once more be achieved by virtue of diagonalization, just as in the sequential case.

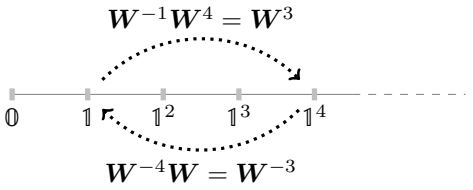
3.5 Variants & Extensions

The structures that we have seen so far are not the only ones that our methodology can tackle – in fact, many other group-like structures are amenable to similar interpretations. We sketch out some enticing examples below.

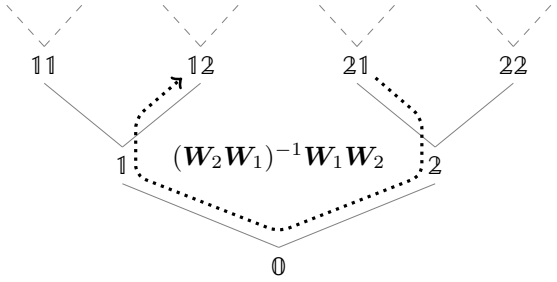
Absolute Positions Our analysis has so far focused on paths *relative* to positions. Fixing the point of origin allows a straightforward simplification to *absolute* positions. The new structure is that of a *monoid*: there's no longer an inversion, and laws L1 and L2 only are now in effect. The framework remains largely unchanged: one can still use subgroups of matrices to represent positions, except this time applying them on either the queries or the keys (rather than both).

Periodic Domains Under addition, the integers form an *infinite* cyclic group. An interesting twist would be to consider the positional encodings of *finite* cyclic groups instead. Such structures are not uncommon; in chemistry, for instance, a benzene molecule comprises six carbon atoms arranged in a ring. The semantics of such a structure would need to be of a matching period; that is, we would need a generator $\boldsymbol{W}$ such that $\boldsymbol{W}^6 = \boldsymbol{I}$. Such a parameterization is straightforward; we simply need to fix the orthogonal matrix so as to have it implement rotations at angle-multiples of $\pi/3$.

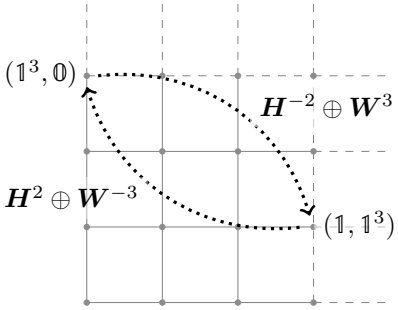
Time Series & Subsampling Our sequential case analysis assumed a dense sequence with a uniform sampling rate. However, our strategy also applies to any series, even if sparsely sampled, as long as the sampling rate is quantized (*i.e.*, a multiple of some constant step). That is, positional indices (and their representations) do not need to match the placement of tokens in the sequence.



(a) The half-axis of absolute positions on a sequence, with a visualization of the two directions of relative paths between points 1 and 4. In either case, the interpretation is the matrix multiplication of the inverted source against the target.



(b) The space of paths on binary branching trees, with an illustration of the relative path from 21 to 12. Same as before, the interpretation is the matrix multiplication of the inverted source against the target



(c) The quarter-plane of absolute positions on a 2-dimensional grid, with a visualization of the two directions of relative paths between points $(3, 0)$ and $(1, 3)$. The interpretation is now a block-diagonal matrix consisting of the blocks interpreting the path over each coordinate.

Figure 1: Example paths and their interpretations across the structures examined.

Composite Groups The direct sum interpretation of Section 3.4 is applicable for arbitrary groups that can be described as products, commutative or otherwise. This allows the representation of positional encodings for several other kinds of composite structures that can be concocted using the same principles, such as sequences of trees, trees of grids, etc.

Beyond Dot-Product Attention Throughout the previous sections, we have adopted a dot-product formulation for the attention weight function. Nonetheless, APE can be readily integrated into any other attention mechanism, such as linear [Katharopoulos et al., 2020], cluster [Vyas et al., 2020] and "softmax-free" [Lu et al., 2021] variants, *inter alia*.

4 Experiments

To assess the viability of our approach, we conduct a series of experiments across a range of tasks, in setups that allow for replicable and reliable comparisons with alternatives. When using APE, we follow Wu et al. [2021] in scaling the dot-product score between two tokens at a distance of p (*i.e.*, p steps away) by p^c; here, we set $c := 0.98$. This serves to stabilize training by introducing a locality bias (or long-distance decay) factor. For the sake of parameter compression, we share the orthogonal matrices between the different encoder/decoder layers, but use a distinct matrix (or collection of matrices) per head. To isolate and quantify the effect of initialization, we report results on two

different initialization strategies: one where the orthogonal operators are set to mimic RoPE rotations (default), and one where they are set to be close to the identity (no init). Similarly, to isolate and quantify the effect of trainability when comparing to RoPE, we report results over both fixed (frozen) and trainable (tuned) rotation angles.

We provide an extensive account of our experimental setups in Appendix B.

4.1 Sequence Transduction

Machine Translation First, we follow Vaswani et al. [2017] in training a Transformer$_{\text{BASE}}$ model on machine translation over WMT14 EN→DE [Bojar et al., 2014].

To provide a comprehensive comparison, we pit our proposed methodology against standard positional encoding schemes from the literature: the vanilla *Sinusoidal* encodings of Vaswani et al. [2017], the *Absolute* encodings of Gehring et al. [2017], the *Relative* encodings of Shaw et al. [2018] and the *Rotary* encodings of Su et al. [2023]. To ensure a fair comparison, we allow all models the exact same budgets (both memory and time).

Synthetic Tasks We further examine three standard sequence transduction tasks: sequence copying, sequence reversal, and sequence repetition. These are meant to directly assess each model's capacity for algorithmic induction, in setups where explicit position-based addressing, both absolute and relative, is required.

4.2 Tree Transduction

Next, we consider four algorithmic transduction tasks on binary branching trees: tree copying, recursive tree rotation up to a fixpoint, algebraic reduction of C_3 expressions, and self-referential tree manipulation; see Appendix B for details.

In addition to previous sequential baselines, we compare our model to the encodings of Shiv and Quirk [2019, *Tree-SQ*]. For all four tasks, we experiment with both breadth-first and depth-first decoding.

4.3 Image Recognition

Finaly, we train a Compact Convolutional Transformer [Hassani et al., 2021] on CIFAR-10 [Krizhevsky et al., 2009].

Typically, attention-based architectures for vision rely on additive positional encoding schemes, applied on the image prior to it being sequentialized (row-by-row flattened). Here, we compare fixed [Wang and Liu, 2019, *Sinusoidal 2D*] and parametric [Gehring et al., 2017, *Absolute*] variants of the above against both the sequential and the grid-structured versions of our scheme.

4.4 Results

We repeat each experiment three times, varying the seeds used for weight initialization and optimization, but fixing the data across repetitions. We report means and 95% CIs in Table 2. We highlight each category's best (in green), and underline scores where the CI spans the mean of the respective best.

At the macro level and consistently across modalities, domain-appropriate algebraic interpretations match or surpass strong and specialized baselines – without *any* hyper-parameter tuning or search. Specifically, across the 13 setups considered, APE is the uncontested top performer in 8, ranks among the best in 3, and falls within the confidence margin of the top performer in one. Exceptionally, in the breadth-first version of the tree-copy task, tree algebraic encodings are surpassed by a handful of sequential alternatives; this is no surprise, since in this case the tree structure is practically a task-irrelevant syntactic confound. Perhaps more surprisingly, in the breadth-first version of the tree-manipulation task, tree algebraic encodings are surpassed only by their non-initialized, sequential version; an anomaly likely due to a single repetition with an unusually low perplexity score.

We also note three general trends. First, initializing APE to match RoPE frequency bands at the start of training consistently and significantly improves performance, possibly because RoPE rotary primitives have undergone empirical tuning for stability and performance. Second, given identical initialization, a sequential APE generally outperforms a trainable RoPE, despite their theoretical equivalence. This might be due to the difficulty of optimizing periodic signals (*i.e.*, RoPE's trigonometric functions) compared to APE's (orthogonal) matrix multiplications. Third, a frozen RoPE performs comparably to

Task	Scheme						
	Sinusoidal	*Absolute*	*Relative*	*Rotary* *(frozen)*	*Rotary* *(tuned)*	*Algebraic* *(/w init)*	*Algebraic* *(w/o init)*
WMT14 EN→DE (BLEU / ↑)	14.57±0.12	22.09±0.11	23.15±0.03	24.03±0.06	23.92±0.20	23.93±0.10	23.84±0.10
COPY	1.01±0.00	1.11±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00
REPEAT	1.85±0.15	3.66±0.06	1.44±0.16	1.08±0.12	1.00±0.00	1.00±0.00	1.02±0.00
REVERSE (PPL. / ↓)	3.92±0.99	4.62±0.67	4.08±1.12	1.09±0.02	1.01±0.00	1.01±0.00	1.03±0.02

(a) Performance results on neural machine translation and synthetic sequence transduction.

Scheme	Task/Regression							
	COPY		ROTATE		C_3		TREE-OPS	
	breadth	depth	breadth	depth	breadth	depth	breadth	depth
Sinusoidal	1.06±0.01	5.68±0.63	6.93±0.38	7.13±0.35	2.66±0.10	2.78±0.08	20.53±7.11	64.86±6.41
Tree-SQ	1.29±0.01	1.07±0.00	2.60±0.16	1.87±0.24	2.27±0.59	2.29±0.24	19.18±3.23	16.41±6.14
Absolute	6.64±0.12	7.02±0.17	7.77±0.15	7.24±0.20	2.77±0.21	2.79±0.22	37.78±0.72	48.91±5.83
Relative	1.01±0.00	6.12±0.06	6.00±0.25	7.72±0.28	1.70±0.07	2.43±0.04	2.36±0.02	16.86±1.27
Rotary (frozen)	1.42±0.58	2.46±0.59	4.58±0.30	4.97±1.79	1.55±0.34	2.15±0.22	2.53±0.08	33.54±9.04
Rotary (tuned)	1.00±0.00	1.70±0.05	4.07±0.34	2.60±0.11	1.08±0.02	1.90±0.22	2.55±0.05	20.87±0.33
*Algebraic (**seq**)*	1.00±0.00	1.63±0.06	2.95±0.08	2.48±0.27	1.07±0.01	1.83±0.02	2.30±0.03	20.05±0.36
w/o init	1.00±0.00	2.36±0.63	5.18±0.10	5.72±1.23	1.45±0.08	2.29±0.06	1.75±0.74	29.26±9.15
*Algebraic (**tree**)*	1.01±0.00	1.00±0.00	1.05±0.01	1.01±0.00	1.00±0.00	1.00±0.00	2.24±0.06	1.83±0.02
w/o init	1.07±0.00	1.04±0.08	1.44±0.15	1.27±0.15	1.05±0.10	1.00±0.00	2.42±0.01	1.86±0.01
(PPL. / ↓)								

(b) Performance results on algorithmic tree manipulation tasks.

Scheme	Epoch	
	≤150	≤300
Sinusoidal 2D	91.57±0.01	92.79±0.20
Absolute	90.86±0.19	92.68±0.39
*Algebraic (**seq**)*	92.68±0.24	94.59±0.15
w/o init	88.93±0.19	91.09±0.20
*Algebraic (**grid**)*	93.13±0.33	94.67±0.06
w/o init	92.95±0.07	94.48±0.18
(ACC. / ↑)		

(c) Best-by-epoch top-1 accuracy scores on image recognition on CIFAR-10.

Table 2: Experimental results and baselines across the tasks considered.

a randomly initialized APE in most tasks considered, suggesting that adjusting rotoreflection angles during training is not necessarily better than adjusting rotation planes while keeping the angles fixed. Contrary to all the above, a frozen ROPE weakly outperforms both a tunable ROPE and an initialized APE in the neural machine translation task; likely an artifact of attention overfitting to specific positional patterns.

5 Related Work

Dense attention is by now a foundational component of various problem- and domain-general architectures. Combined with its structural indifference, this underscores the pressing need for learning strategies capable of injecting structural biases directly at the representation level. As such, positional encodings have garnered significant community attention in recent years – too much, in fact, to permit an exhaustive enumeration here. An extensive survey and meta-review is provided by Dufter et al. [2022] who group and rank these works on the basis of several criteria. Our work presents a universal, intuitive and formally grounded recipe that meets *all* these criteria: it is *trainable*, amenable to problem-specific and data-driven tuning; *reference-adjustable*, allowing both absolute and relative positional specifications; *unbounded*, capable of representing enumerably infinite positions irrespective of model instantiation and/or the targeted data size; *contextual*, implementing a dynamic effect that varies depending on token content; *effective*, consistently matching or surpassing

baselines in the tasks considered; and, finally, *efficient*, exhibiting generally favorable asymptotic complexities.

We must point out that the concept of positional encodings as sequence homomorphisms has already been hinted at, first by Wang et al. [2020] and later by Su et al. [2023], even if not explicitly formulated as such. Despite approaching the problem from different angles, both approaches interpret positions as multiplicative, norm-preserving (rotation-like) operations. Our proposal expands upon these two, first in providing a proper algebraic framing of the problem, and second in extending the interpretation from rotations around the axes to rotations and reflections about arbitrary planes. In the case of a single generator matrix (*i.e.*, sequences), this difference turns to be non-essential, being practically neutralized by the Transformer's trainable weights. This no longer holds, however, in the case of multiple generator matrices (*i.e.*, grids or trees), where each generator should be able to rotate and reflect different sets of planes. In that sense, algebraic positional encodings offer an appealing unifying perspective of a multidimensional generalization to the aforementioned rotation-based frameworks. This sentiment is shared by Lim et al. [2023] who, in parallel to our work, similarly advocate for positional encodings as group homomorphisms, there framed as irreducible group representations. Modulo presentation, the two approaches are variations on a common theme; theirs is technically concerned with post-hoc representation of symmetries and equivariances at a per-datum scale, whereas ours focuses on the interpretation of domain signatures at the dataset scale.

More generally, algebraic manipulations are not uncommon in modern machine learning literature. The recognition of abstract algebra as a practical tool for imposing structural well-behavedness has led to its increased adoption as the go-to recipe for structure-informed neural architectures, largely obsoleting the inefficient and *ad hoc* augmentation routines of the past. This line of work can be traced back to the group equivariant convolutions of Cohen and Welling [2016], which have by now bloomed into a field of their own; see Weiler et al. [2023] for an up-to-date overview.

6 Limitations

We recognize weaknesses and limitations across three fronts. On the *theoretical* front, we have limited our scope to simple inductive groups, consciously ignoring potential interpretations of more complex constructions. We defer this to future work. On the *empirical* front, having to recompute positional encodings once per batch increases a model's temporal complexity during training. While this is barely noticeable in sequential and grid constructions, which scale logarithmically, it becomes evident when dealing with complete trees, which scale linearly and require explicit for-loops. On the *epistemic* front, we conducted a limited set of experiments, focusing primarily on replicability and fairness. We leave more exhaustive empirical comparisons on practical downstream tasks to future work or interested parties.

7 Conclusion

We have presented a theoretically motivated approach towards constructing positional encodings for a variety of structures. Without any significant modification or overhead, our methodology can capture sequences and their (multi-dimensional as well as multi-branching) generalizations. In doing so, it reconciles powerful but structurally oblivious models with their missing inductive biases, permitting structure-aware architectural refinements across a range of tasks and setups (see also Kogkalidis et al. [2024] for parallel work employing the methodology in a neurosymbolic representation learning setup). Beyond that, our approach grants full control over how these biases are to be implemented, while also being amenable to adjustments and extensions. Our work indicates that generality and extensibility are not *in spite of*, but rather *due to* structural discipline and abstraction. We perceive it as an important step towards data-efficient, general, and transparent models of neural computation.

Acknowledgments and Disclosure of Funding

KK and VG were supported by Saab-WASP via the project "Neurodynamic Programming and Reinforcement Learning" (grant 411025). VG also acknowledges the support from Academy of Finland (grant 342077) for "Human-steered next-generation machine learning for reviving drug design", and the Jane and Aatos Erkko Foundation (grant 7001703) for "Biodesign: Use of artificial intelligence in enzyme design for synthetic biology".

References

M. Arjovsky, A. Shah, and Y. Bengio. Unitary evolution recurrent neural networks. In *International conference on machine learning*, pages 1120–1128. PMLR, 2016.

J.-P. Bernardy and S. Lappin. Unitary recurrent networks: Algebraic and linear structures for syntax. In *Algebraic Structures in Natural Language*, pages 243–278. CRC Press, 2022.

O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, et al. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation*, pages 12–58, 2014.

T. Cohen and M. Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

P. Dufter, M. Schmitt, and H. Schütze. Position information in transformers: An overview. *Computational Linguistics*, 48(3):733–763, Sept. 2022. doi: 10.1162/coli_a_00445. URL https://aclanthology.org/2022.cl-3.7.

P. Gage. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38, 1994.

J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In *International conference on machine learning*, pages 1243–1252. PMLR, 2017.

A. Hassani, S. Walton, N. Shah, A. Abuduweili, J. Li, and H. Shi. Escaping the big data paradigm with compact transformers. *arXiv preprint arXiv:2104.05704*, 2021.

T. Janssen. *Foundations and applications of Montague grammar*. PhD thesis, University of Amsterdam, 2014. Originally published: April 1983 (UvA).

A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

K. Kogkalidis, O. Melkonian, and J.-P. Bernardy. Learning structure-aware representations of dependent types. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Y. Li, R. Zemel, M. Brockschmidt, and D. Tarlow. Gated graph sequence neural networks. In *Proceedings of ICLR'16*, 2016.

D. Lim, H. Lawrence, N. T. Huang, and E. H. Thiede. Positional encodings as group representations: A unified framework. 2023.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

J. Lu, J. Yao, J. Zhang, X. Zhu, H. Xu, W. Gao, C. Xu, T. Xiang, and L. Zhang. Soft: Softmax-free transformer with linear complexity. *Advances in Neural Information Processing Systems*, 34: 21297–21309, 2021.

F. Murnaghan and A. Wintner. A canonical form for real matrices under orthogonal transformations. *Proceedings of the National Academy of Sciences*, 17(7):417–420, 1931.

M. Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels, Oct. 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W18-6319.

R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, 2016.

P. Shaw, J. Uszkoreit, and A. Vaswani. Self-attention with relative position representations. In *Proceedings of NAACL-HLT*, pages 464–468, 2018.

V. Shiv and C. Quirk. Novel positional encodings to enable tree-based transformers. *Advances in neural information processing systems*, 32, 2019.

J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, page 127063, 2023.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

A. Vyas, A. Katharopoulos, and F. Fleuret. Fast transformers with clustered attention. *Advances in Neural Information Processing Systems*, 33:21665–21674, 2020.

B. Wang, Z. Donghao, L. Christina, Q. Li, Z. Peng, J. G. Simonsen, et al. Encoding word order in complex embeddings. In *ICLR 2020-Proceedings of Eighth International Conference on Learning Representations*, 2020.

Z. Wang and J.-C. Liu. Translating math formula images to latex sequences using deep neural networks with sequence-level training, 2019.

M. Weiler, P. Forré, E. Verlinde, and M. Welling. Equivariant and coordinate independent convolutional networks. *A Gauge Field Theory of Neural Networks*, 2023.

C. Wu, F. Wu, and Y. Huang. DA-Transformer: Distance-aware transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2059–2068, 2021.

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

A Parameterizing APE

A.1 Orthogonalization

The orthogonal primitives underlying APE can be procured by matrix-exponentiating skew-symmetric bases. Concretely, for some cyclic group $\langle \boldsymbol{C} \rangle$:

1. Start with an *upper triangular* matrix $\boldsymbol{A}$; this matrix parameterizes the entire group.
2. Obtain the *skew symmetric* $\boldsymbol{B} := \boldsymbol{A} - \boldsymbol{A}^\top$
3. Obtain the *matrix exponent* $\boldsymbol{C} := \exp\mathrm{m}(\boldsymbol{B})$; the resulting matrix is *orthogonal*, and acts as the group's generator.

A.2 Switching between APE and RoPE

In the commutative (direct sum of finitely many cyclic groups) case, it is possible to switch freely between APE and RoPE. Doing so might be useful, *e.g.*, for initializing APE, for inspecting the learned rotoreflections post-training, or for making use of RoPE's memory-optimized vector-multiplication formula in a system originally trained with APE. Note that here we consider the purely real-valued version of RoPE (and APE).

RoPE → APE To convert RoPE to APE for some collection of angles $\Theta := [\theta_1, \dots \theta_n]$:

1. Expand Θ into a rotation matrix $\boldsymbol{C}$,

$$\boldsymbol{C} := \begin{bmatrix} cos\theta_1 & -sin\theta_1 & 0 & 0 & \dots \\ sin\theta_1 & cos\theta_1 & 0 & 0 & \dots \\ 0 & 0 & cos\theta_2 & -sin\theta_2 & \dots \\ 0 & 0 & sin\theta_2 & cos\theta_2 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

   Note: Stop here if not interested in parameterizing $\boldsymbol{C}$.
2. Use a solver to approximate the *matrix logarithm* of $\boldsymbol{C}$, $\boldsymbol{B} := \log\mathrm{m}(\boldsymbol{C})$.
3. Find a matrix $\boldsymbol{A}$ such that $\mathrm{mse}(\boldsymbol{B}, \boldsymbol{A} - \boldsymbol{A}^\top) \leq \epsilon$, *e.g.*, using a numerical optimizer. Matrix $\boldsymbol{A}$ can be used to parameterize the group, *cf.* A.1.

APE → RoPE To convert APE to RoPE for some cyclic group $\langle \boldsymbol{W} \rangle$:

1. Find the normal form $\boldsymbol{W} = \boldsymbol{P}\boldsymbol{Q}\boldsymbol{P}^\top$.
2. Extract the angles in each block of $\boldsymbol{Q}$; the resulting collection of angles is RoPE's Θ.
3. For each attention head involved, right-compose the Transformer's $\boldsymbol{\Phi}^{(q)}$ and $\boldsymbol{\Phi}^{(k)}$ with $\boldsymbol{P}$.

B Experimental Setups

B.1 Machine Translation

For our machine translation experiments, we use the official dataset breakdown (including the extended evaluation set). We tokenize the training and evaluation sets with MOSES, using the standard pipeline: punctuation normalization → unicode normalization → language-specific tokenization.[6] We apply byte-pair encoding [Gage, 1994, Sennrich et al., 2016] using the `subword-nmt` package.[7] We apply 32k merges across the source and target training corpora, without truncating the resulting (shared) vocabulary (of size 35 533). Our loss term is given as the cross-entropy between the teacher-forced predictions and the ground-true labels, smoothed by 10%. We train in a distributed environment consisting of 4 GPUs, with a batch size of 3 072 target tokens per GPU. We average gradients and update parameters once every 2 GPU iterations (or: 8 batches). We optimize using Adam with a learning rate dictated by the schedule prescribed by Vaswani et al. [2017]. We stop optimizing after 150 000 parameter updates or 16 hours, whichever comes first. Throughout training, we circularly store the 10 best checkpoints, ranked on the basis of dev set loss (evaluated once every 500 updates). During inference, we average the 10 checkpoints into a single model, and select hypotheses from a beam of width 4 and a length penalty of 0.6 [Wu et al., 2016]. We report BLEU scores over the *test set* (`newstest2014`), comparing the BPE-merged and detokenized output against the raw references using `sacrebleu` [Post, 2018].[8]

[6] See https://github.com/moses-smt/mosesdecoder

[7] See https://github.com/rsennrich/subword-nmt.

[8] Signature: `nrefs:1 | case:lc | eff:no | tok:13a | smooth:exp | version:2.4.2`.

Parameter	Experiment/Value		
	NMT	Transduction	Image
Convolution Size	–	–	(3,3)
Convolution Stride	–	–	1
Embedding Size	512	512	256
Feedforward Size (enc)	2048	512	512
Feedforward Size (dec)	2048	1024	–
Feedforward Activation	ReLU	ReLU	GELU
# Layers (enc, dec)	(6, 6)	(2,2)	(7, 0)
# Heads	8	8	4
Norm	LayerNorm	LayerNorm	LayerNorm
Norm Position	Post	Pre	Pre

Table 3: Hyperparameter setups, grouped by experiment.

B.2 Synthetic Transduction

Tree Task Descriptions The tree copy task is morally identical to its sequential version – the tree structure (and its positional specification) is practically a confound.

In the tree rotation* task, the output tree is the result of recursively right-rotating all subtrees of the input. The task is challenging but purely structural, in the sense that its resolution requires no real interaction between content and position.

For the algebraic expression reduction task, we consider input trees that specify a complex expression from the cyclic group C_3, and task the model with producing the result of a single reduction step (*i.e.*, reducing all subtrees of depth 1 into a leaf). This time around, the model has to identify reducible subtrees, match operators to their argument and collapse the three into a single node depending on their content.

The tree operations task, finally, combines the aspects of the other three, requiring content-based addressing, structure manipulation and dynamic semantics resolution. Concretely, we generate an input tree consisting of unique nodes, and randomly select one of its subtrees as well as one of four operators. We then construct a deeper tree, where the new root corresponds to the chosen operator, its left branch corresponds to the numerical index of the selected subtree, and the right branch is the original tree in its entirety. The model is then tasked with producing the correct output given this combination of an operator, a tree, and an index. We consider four operations: extraction (*i.e.*, return the indexed subtree), flip-extraction (*i.e.*, return the indexed subtree, rotated), truncation (*i.e.*, return the full tree with the indexed subtree removed) and a no-op (*i.e.*, return the full tree as-is, ignoring indexing).

Hyperparameters For all synthetic tasks, we generate disjoint train, dev and test sets of sizes 6 000, 2 000 and 2 000. We train a small Transformer model, optimizing with AdamW [Loshchilov and Hutter, 2017] for 400 epochs and a batch size of 64, using a linear warmup – cosine decay schedule. For the sequential tasks, we populate the datasets with words of random lengths from $\mathcal{N}(100, 10)$ and a vocabulary size of 20 (to ensure token repetition and diffuse the possibility for leaning on content-based addressing). For the tree tasks, we populate the datasets with non-uniform trees of random depths sampled from $\mathcal{N}(7, 1)$. For the tree-ops task, exceptionally, we set the vocabulary size to 128 so as to have enough unique nodes to allow content-based addressing.

When using a positional encoding scheme that requires fixing the size of the structure being modeled (*i.e.*, the *Tree*, *Relative*, and *Absolute* schemes), we fix it at approximately the maximum training size, practically ensuring the most stringent comparison.

In all experiments, we share source and target embedding weights between both the encoder-decoder embedding layers, and the decoder's classification head.

B.3 Image Recognition

For our image recognition experiments, we largely rely on the setup of Hassani et al. [2021]. Concretely, we apply a small-step "tokenizing" convolution on the input image, downsample the result with max pooling and flatten the result into a sequence. After we pass the sequence through the encoder, we apply a global soft attention [Li et al., 2016, *inter alia*] (rediscovered by Hassani et al. [2021], there dubbed "sequence pooling") to aggregate into a single vector prior to applying the classifier. To attain competitive scores, we apply standard CIFAR-10 data augmentations and

more aggressive regularization: a 10% attention weight dropout, a stochastic depth of 10% for each consecutive layer, and a weight decay of $3 \cdot 10^{-2}$. The above settings and the hyperparameter setup are taken without modification from Hassani et al. [2021].

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We carefully summarize our contributions and refrain from making any claims that we cannot theoretically or empirically support.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We have a dedicated limitations section (§6), and openly and explicitly discuss algorithm complexity and experimental scope in the relevant sections.

   Guidelines:

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

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

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

   Answer: [Yes]

   Justification: Our algebraic connections make no assumptions and are fully explicit in their presentation. The equivalence with RoPE clarifies all assumptions it makes.

   Guidelines:

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

4. **Experimental Result Reproducibility**

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

   Answer: [Yes]

   Justification: We provide the reviewers with both an extensive appendix detailing our experimental setup, and the code used to implement our methodology and conduct our experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

(a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

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

   Answer: [Yes]

   Justification: Yes – see answer above. Our training scripts are provided virtually unchanged.

   Guidelines:

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

6. **Experimental Setting/Details**

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

   Answer: [Yes]

   Justification: Yes, see above.

   Guidelines:

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

- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

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

   Answer: [Yes]

   Justification: We take extra care to conduct our experiments openly and transparently so as to deliver statistically sound results and draw solid conclusions. We repeat *all* experiments multiple times, and report means and 95% confidence intervals. For each experiment, we visually mark all models that overlap with the best performer in the category.

   Guidelines:

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

8. **Experiments Compute Resources**

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

   Answer: [No]

   Justification: While we do report hardware infrastructure, we do not report memory consumption or clock times. With the exception of machine translation, our experiments are moderately cheap to run, requiring no specialized hardware other than GPU accelaration.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: Checked and done.

   Guidelines:

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

10. **Broader Impacts**

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

    Answer: [Yes]

    Justification: We do, albeit briefly. We do not see possible negative implications.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [NA]

Justification: We perceive no risks that would require safeguards of any kind.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: We cite all software libraries and datasets we use, and comply with their licenses.

Guidelines:

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

13. **New Assets**

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

Answer: [No]

Justification: While we do provide reference implementations, we do not see them as assets per se, neither do we hand them out as ready-to-use integrations.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.

- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

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

    Answer: [NA]

    Justification: No human subjects were involved in this study.

    Guidelines:

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

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

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

    Answer: [NA]

    Justification: See above.

    Guidelines:

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