A Group Theoretic Analysis of the Symmetries Underlying Base Addition and Their Learnability by Neural Networks

Cutter Dawes[a,*], Simon Segert[b], Kamesh Krishnamurthy[c], Jonathan D. Cohen[d,e]

[a]*Department of Mathematics, Princeton University*
[b]*Independent researcher*
[c]*Zyphra*
[d]*Princeton Neuroscience Institute, Princeton University*
[e]*Department of Psychology, Princeton University*

Abstract

A major challenge in the use of neural networks both for modeling human cognitive function and for artificial intelligence is the design of systems with the capacity to efficiently learn functions that support radical generalization. At the roots of this is the capacity to discover and implement symmetry functions. In this paper, we investigate a paradigmatic example of radical generalization through the use of symmetry: base addition. We present a group theoretic analysis of base addition, a fundamental and defining characteristic of which is the carry function – the transfer of the remainder, when a sum exceeds the base modulus, to the next significant place. Our analysis exposes a range of alternative carry functions for a given base, and we introduce quantitative measures to characterize these. We then exploit differences in carry functions to probe the inductive biases of neural networks in symmetry learning, by training neural networks to carry out base addition using different carries, and comparing efficacy and rate of learning as a function of their structure. We find that even simple neural networks can achieve radical generalization with the right input format and carry function, and that learning speed is closely correlated with carry function structure. We then discuss the relevance this has for cognitive science and machine learning.

Keywords: machine learning, cognitive science, symmetry, base addition, group theory

1. Introduction

One of the defining characteristics of human cognitive function, that continues to distinguish it from the most advanced systems in machine learning and artificial intelligence, is the efficiency with which people can acquire and represent knowledge that generalizes widely out of distribution. This can be distilled, in the purest form, to the discovery of symmetries: forms of structure that can be identified in one domain, and that remain invariant over a relevant set of transformations into others. It has been said that "it is

*Corresponding author
Email address: cdawes@princeton.edu

only slightly overstating the case to say that physics" – one of the most fundamental and widely applicable domains of human knowledge – "is the study of symmetry" (Anderson, 1978).

The same might be said about the understanding of human cognition, as well as efforts to approximate its capacity for sample efficient learning and radical generalization in artificial systems. This is because, where the structure of a data distribution is defined by a symmetry, then the distribution can be learned with only the amount of data needed to infer the symmetry, which can then be used for effective generalization not only over any additional data within the envelope of the training distribution, but all data that exhibits the same structure outside of the distribution as well. This follows from the definition of a symmetry: that it is invariant to transformations of the data that conform to its structure.

Neural networks have been developed that, with the appropriate inductive bias, can efficiently discover symmetries in specific domains. In so doing, these models can match or exceed human performance in those domains. For example, convolutional neural networks are imbued with mechanisms for "weight sharing" that allow them to discover spatially invariant constituents of visual displays, such as human recognizable objects, and then identify such objects even when they appear in previously unseen locations (Lecun et al., 1998; Krizhevsky et al., 2012). Here, the nature and scope of the symmetry is well defined – translational invariance over spatial position – and explicitly implemented in the model (as the spatial convolution operation). Similarly, since recurrent networks use the same weights to process all elements of a sequence, they can be viewed as implementing of a form of "weight sharing over time," which may explain their ability to discover symmetries in sequential structure in ways similar to people (Cleeremans and McClelland, 1991; Segert and Cohen, 2022; Elman, 1990; Ji-An et al., 2024). More powerful architectures, such as transformers, also appear to be capable of discovering symmetries that can be remarkably abstract. This is evidenced by the impressive generalization capabilities of large language models (LLMs), such as their ability to solve analogical reasoning problems at or above human level performance (e.g., Webb et al., 2023). In some cases, the nature of the symmetries are coming into view that appear to support simple forms of symbolic computation (e.g., Yang et al., 2025). However, in many other cases it is not clear precisely what symmetries have been discovered, and in virtually all cases the amount of data needed for the system to discover them is considerably greater than humans require to exhibit similar generalization capabilities.

As a step towards understanding the capabilities described above, we focus on one of the simplest and most fundamental forms of symmetry: base arithmetic. The ability to count algorithmically using a base (most commonly 10) is one of the earliest milestones of human learning (Wynn, 1992; Dehaene, 1997; Piantadosi, 2023), and is fundamental to base arithmetic, which lies at the heart of mathematical reasoning, our most abstract form of thought and communication. Despite this fact, little work has been done characterizing base arithmetic from a group theoretic perspective. Furthermore, to date, there are no neural networks that have been shown to achieve, strictly through learning, demonstrably algorithmic forms of base arithmetic computation. Here, we seek to make progress toward both of these closely related goals: a deeper understanding of the symmetry properties of base arithmetic; and an examination of how this relates to the ability of neural networks to learn such symmetries.

Specifically, we focus on the symmetries inherent in base addition. This is of course

an elemental component of base arithmetic, and provides a simple and well defined example of symmetry: once the operations needed to perform addition are known for some subset of the data (e.g., for a number of digits, or "places", sufficient to reveal the underlying carry function), they can be applied to indefinitely large numbers. In the sections that follow, we begin by providing a formal group theoretic analysis of the operations underlying base addition, and then examine the ability of neural networks to learn these operations. The group theoretic analysis allows us to identify, for a given base, different symmetry functions (each corresponding to a different carry function) that are functionally equivalent, but vary in structure. We describe several quantitative measures (fractal dimension, frequency of carrying, and associativity fraction) that we use to characterize the structure of each set of carry functions, and find that these quantitative measures are tightly correlated with how efficiently and effectively neural networks learn those carry functions. We suggest that these observations may provide a normative account of the form of base addition people use, that involves the simplest symmetries; and that this, in turn, may be useful in designing neural networks capable of learning and exploiting such symmetry functions in the service of fully algorithmic and generalizable base addition.

Section 2 provides a formal group theoretic analysis of base addition (including an overview of necessary group theoretic concepts in Section 2.1, which can be skipped by readers familiar with these concepts). Section 3 introduces quantitative measures that characterize carry function structure, and Section 4 reports results from training neural networks to add using each carry function. Section 5 provides a general discussion of the relevance of our findings to related questions in mathematics, neural networks, and cognitive science, and Section 6 provides a conclusion.

2. Mathematics of Symmetry and Base Addition

2.1. Symmetry in Group Theory

The branch of algebra known as group theory provides a formal definition of symmetry in terms of groups. A *group* is a set G together with a group operation $\cdot$ satisfying the group axioms:

1. *associativity*: $(x \cdot y) \cdot z = x \cdot (y \cdot z)$;
2. *identity*: there exists some $e \in G$ such that $e \cdot x = x = x \cdot e$ for all $x \in G$;
3. *inverses*: for every $x \in G$, there exists $x^{-1} \in G$ such that $x \cdot x^{-1} = e = x^{-1} \cdot x$.

The operator $\cdot$ enforces a structure on G, or equivalently expresses an underlying symmetry. For example, let G be the rotational group of the equilateral triangle, in which the operator is a composition of rotations and the elements are rotations of the triangle by multiples of 120°. Notice that this group of rotations – in which there are only three non-degenerate elements – precisely describes the geometric symmetry of the triangle; and conversely, the symmetric structure of the triangle may be defined exactly as this group G. It is important to note that two different groups can express the same symmetry (i.e., are *isomorphic*; see Appendix A.1 for details). For example, G of our example is isomorphic to the group of integers modulo 3, with elements $0, 1, 2$ and a group operation of addition modulo 3 (e.g., $2 + 2 = 1$).

2.1.1. Symmetry Functions

Groups are a natural setting in which to study functions that use symmetry for *radical generalization* (i.e., extrapolation far beyond the training distribution, requiring more than just surface correlations). This hinges on two central concepts, *invariance* and *equivariance*, that refer to the extent to which a functional form respects a given symmetry. For example, single-digit modulus arithmetic can be thought of as reflecting an invariance – insofar as it is used for all places in multi-digit arithmetic – while its application across multiple digits (i.e., over multiple scales) reflects equivariance. We formalize these concepts in the general case here, and then consider their specific application to base arithmetic in Section 2.1.2. To treat these formally, consider the case in which we wish to learn a function $f : X \to Y$, the domain (X) and co-domain (Y) of which have a common symmetry – i.e., they are acted on by the same group G (see Appendix A.1 for details). f is invariant with respect to G if transformations by G leave f unchanged; that is, if $f(g \cdot x) = f(x)$ for all $x \in X$ and $g \in G$. More generally, f is equivariant if f intertwines with transformations by G; that is, $f(g \cdot x) = g \cdot f(x)$ for all $x \in X$ and $g \in G$.[1] In this respect, invariance is a special case of equivariance in which the action of G on Y is trivial (i.e., $g \cdot y = y$ for all $g \in G$, $y \in Y$). Both invariance and equivariance imply that f respects the common symmetry of X and Y; hence, we call such an f a *symmetry function*.

Equivariance is a crucial prerequisite for radical generalizability: Consider the function f that has been observed only on some small sub-domain $A \subset X$ sufficient to reveal the underlying symmetry (i.e., the action of G on each X and Y). If f is equivariant, then we may infer $f(y) = g \cdot f(x)$ for any $y = g \cdot x \notin A$. Conversely, if f is not equivariant, it becomes impossible (at least on the basis of group-theoretic structure alone) to broadly extrapolate from the sub-domain A to its orbit $G \cdot A$ (see Appendix A.1 for details). Therefore, radical generalization is possible if and only if the unknown function f is equivariant to some known symmetry – i.e., if f is a symmetry function. Accordingly, (i) all spaces relevant to radically generalizable function learning have symmetry; and (ii) all radically generalizable functions we wish to learn are symmetry functions.

2.1.2. Group Extensions

One final group theoretic construct relevant to our considerations is that of a *group extension*, which refers to the way that a group A can be "extended" by another group G to form the extension E; specifically, E is a group in which G and A become embedded. The use of clocks to represent time provides an instructive example, and one that is closely related to the case of base addition that we consider below: the clock (E) can be seen as a rotational group (the hour hand A, with 12 non-degenerate elements), extended to have an additional scale of structure – the minute hand (G) – using a similar symmetry (in that case with 60 non-degenerate elements).

Two extensions E and E' are *equivalent* if, not only are E and E' isomorphic as groups, but A and G are embedded in "the same way" in E and E'. Just as we represent time using hours and minutes, we may seek to represent an extension E on the set $A \times G$; doing so reduces to constructing a group operation on the set $A \times G$ that makes it an equivalent extension to E. Here, *group cohomology* – a mathematical method for studying

[1] In an abuse of notation, here $\cdot$ refers to the action on both X and Y.

groups by chaining them into sequences – provides a useful approach: in particular, we use the concepts of cocycles and coboundaries. In the context of group extensions, a *cocycle* is the part of the group operation on $A \times G$ (specifically, a function $f : G \times G \to A$) that makes the extension on $A \times G$ equivalent to E, whereas a *coboundary* is the "difference" between the cocycles of two equivalent extensions both represented on $A \times G$. In a clock, the cocycle is the function specifying that the hour hand increments when the minute hand crosses 60. And, considering an alternative clock that counts counterclockwise in the minute hand, the coboundary is the function reversing the minutes (for minute m, $60 - m$) of a normal clock.

Morning (AM) vs. afternoon/evening (PM), days, and years all reflect repeated extensions of the same form of symmetry (a rotational group, differing only by the number of elements). If applied properly, this repeated extension of a rotational group achieves equivariance, such that the system is synchronized across scales. As such, in this paper we frequently refer to invariance and equivariance as properties of a repeated extension, insofar as: the symmetry invoked at each scale is invariant; and, if the extension is valid, its group operation is equivariant (i.e., associative). For all of the above, see Appendix A.2 for a formal treatment. Below, we will formulate base addition in these terms and, in so doing, create various scales of structure within the integers (in base 10, the 1's place, 10's place, 100's place, and so on).

2.2. Base Addition as a Symmetry

The integers are a paradigmatic symmetry group, that we denote by $\mathbb{Z}$: its elements are integers, and the group operation is addition ($+$; henceforth called *integer addition*). Humans exploit this for radical generalization through the use of base addition. To see this, we can rewrite $+$ as a function $a : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$. Now, suppose a is known over finite $A \subset \mathbb{Z} \times \mathbb{Z}$ (e.g., examples of addition involving only two or three digits[2]), and we wish to learn a over all of $\mathbb{Z} \times \mathbb{Z}$ (i.e., addition over numbers with any number of digits).[3] Given a minimally adequate size of A (viz., a number of digits sufficient to exemplify all possible carries) then, insofar as a is equivariant to translation by $\mathbb{Z}$ (implied by the associativity of addition; i.e., $(x + y) + z = x + (y + z)$ may be rewritten as $a(x + y, z) = x + a(y, z)$), the remaining $\mathbb{Z} \times \mathbb{Z} - A$ that can be computed is infinite. That is, learning from a minimal set of examples A (that reveal the underlying symmetry), is sufficient to support generalization over the entire domain $\mathbb{Z} \times \mathbb{Z}$ of the function a.

The foregoing shows how base addition can serve as a simple, well-characterized, and fundamental example of how discovering symmetry functions can support learning efficiency and radical generalization. It is clear that humans exploit this when learning and carrying out base addition: we have developed an algorithmic kernel that exploits

[2]Note that an integer in $\mathbb{Z}$ has no inherent structure such as digits – rather, that is exactly what we will construct in its base representation – but we refer to digits here in part for clarity (reflecting just how fundamental a base representation is to our understanding of integers) and to motivate the usefulness of a base representation.

[3]Our setup is reminiscent of Saul Kripke's skeptical response to Wittgenstein on rule-following: supposing a is known only over finite A, how can we know that we are properly generalizing a to $\mathbb{Z} \times \mathbb{Z}$, and not instead some alternative a' which agrees with a over A? We would argue that it follows from the construction of a base representation, which in turn relies on the axioms of group theory and cohomology. By making explicit what we assume, mathematical axioms address – without necessarily offering a solution to – the infinite regress problem posed by Kripke's skeptic.

the symmetry of $\mathbb{Z}$ and can be applied recursively over a. In this solution, the kernel itself is invariant (i.e., adding modulo base b in each place), and the desired equivariance is implemented by "extending" that kernel via a recursive "place-keeping" function (i.e., successively carrying between digits).

More precisely, at the core of the solution is $\mathbb{Z}_b$: the finite group of integers modulo b, comprised of elements $0, 1, ..., b-1$ and the group operation *addition modulo b*. It is a foundational property of arithmetic that we may uniquely (up to leading zeros) represent any non-negative integer $n \in \mathbb{Z}_{\geq 0}$ as a sequence of such digits; that is, as a tuple $(n_k, n_{k-1}..., n_1)$, where $n_j \in \mathbb{Z}_b$ is the j^{th} digit for $j \in [k]$.[4] We call such multi-digit tuples a number's *base representation*. This characteristic of arithmetic is reflected in the fact that, for decimal addition (i.e., base 10), we do not have unique symbols for integers higher than 9. However, in order to faithfully reproduce the structure of $\mathbb{Z}$, it is necessary to construct an equivalent addition operator in the base representation (viz., a group operator which makes the base representation equivalent to the integers), which we call *base addition*. Isaksen (2002) spells this out for the 2-digit case in base 10 (i.e., up to 99), and Segert (2024) extends this to the general case of multi-digit numbers with any length. Constructing a base addition operator amounts to defining a procedure for "carrying" an appropriate number to the next place depending on the pair of digits at the preceding place, which we call a *carry function*. This, in turn, can be formulated in terms of cocycles and group cohomology (introduced in Section 2.1.2). As we elaborate below, doing so reveals a variety of carry functions for bases greater than 2, that we go on to characterize in subsequent sections.

2.3. The Cohomological Construction of Base Addition

Following Isaksen (2002), we construct base addition using the formalism of group cohomology. Here, we present the construction in an intuitively accessible form; a more rigorous mathematical treatment is provided in Appendix A.2. The problem may be stated as follows. Suppose we have two non-negative integers n and m in their base representations $(n_{k_n}, ..., n_1)$ and $(m_{k_m}, ..., m_1)$. What is the sum $s = n + m$ in its base representation $(s_{k_s}, ..., s_1)$?[5]

In arithmetic, humans are first taught to approach this problem sequentially, by adding the digits in each place: start with the least significant (rightmost) place; add the digits in that place modulo the base; and, if the sum in that place is greater than or equal to the base, carry 1 over to the addition of the digits in the next-most-significant (left-adjacent) place; then the apply the same algorithm to that next place, and so forth. This can be formalized in terms of the base representation of s as:

$$\begin{aligned} s_j &= n_j + m_j + c_j, \\ c_{j+1} &= \mathbb{1}_{n_j + m_j + c_j \geq b}, \end{aligned} \tag{1}$$

[4] Here, for simplicity, we limit our consideration to non-negative integers (for details on the group cohomological construction, see Appendix A.2). However, it is straightforward – albeit not as elegant – to include the negative integers, by including them as elements and modifying the group operation appropriately.

[5] Here, we consider adding two non-negative integers, but note that if the base representation is properly constructed, this naturally extends to adding three or more non-negative integers using the associativity of addition.

where c_j is the carry to the j^{th} digit (note, $c_1 = 0$). We call this particular carry function (i.e., $c_{j+1} = \mathbb{1}_{n_j+m_j+c_j \geq b}$) the **1** carry function. However, as we consider below, it is just one of several possible ways of carrying for bases greater than 2. There are two important and separate factors to consider about Equation 1: validity and efficiency. With respect to validity, this base addition formula is equivalent to integer addition (a formal explanation is provided in Appendix A.2). With respect to efficiency, this procedure – a simple algorithmic kernel applied recursively (i.e., repeatedly over the pairs of digits at each place, in order of their increasing significance) – is remarkably compact: it reduces learning addition in infinite $\mathbb{Z}$ to learning addition in finite $\mathbb{Z}_b$ and the **1** carry function.

2.3.1. Carry Functions

In addition to the **1** carry function, it is possible to identify other carry functions and evaluate both their validity and efficiency. This may be of interest for at least two reasons: (i) mathematical inquiry (what carry functions preserve the structure of the integers, and what might their associated base representations look like?); and (ii) learnability (e.g., where there is more than one carry function, to what extent do they vary in complexity and, consequently, the ease with which they can be learned?).

To consider different carry functions, we can generalize the procedure for base addition in Equation 1 as follows. First, consider the 2-digit case. Given $f : \mathbb{Z}_b \times \mathbb{Z}_b \to \mathbb{Z}_b$, the base representation of s is given simply by

$$\begin{aligned} s_1 &= n_1 + m_1, \\ s_2 &= n_2 + m_2 + f(n_1, m_1) \end{aligned}$$

where f is the carry function.[6] For an arbitrary number of digits, the carry function must take account of what has been carried previously; hence, the j^{th} digit of s is given by

$$\begin{aligned} s_j &= n_j + m_j + c_j, \\ c_{j+1} &= f(n_j, m_j) + f(n_j + m_j, c_j), \end{aligned} \tag{2}$$

where f is again the carry function, and $c_1 = 0$ as in Equation 1 (see Appendix A.2 for a derivation of Equation 2).

2.3.2. Carry Function Validity and k-Equivariance

For a carry function f to be valid, it must preserve the structure of $\mathbb{Z}$. There are two necessary and sufficient conditions for this to hold: (i) preservation of associativity, ensuring that the resulting base representation forms a group; and (ii) equivalence to integer addition, ensuring that the base representation is isomorphic to $\mathbb{Z}$ in particular.

Regarding associativity, a given carry function f may exhibit this – and hence the corresponding base addition may be equivariant – only up to k-digit numbers, where $k \in \mathbb{N}$ can vary, and that we refer to as k-equivariance. In the simplest 2-digit case, a carry function f preserves associativity if and only if f is a cocycle. In general, we

[6] Note that, in the 2-digit case and the general case (Equation 2), if two numbers differ in length, we zero-pad the shorter one from the left to match the length of the longer one.

say that a carry function f is *k-equivariant* if it preserves associativity up to k digits, and *∞-equivariant* if it does so for an arbitrary number of digits (see Appendix A.2 for details). The notion of k-equivariance suggests one way to categorize carry functions; we consider others in the sections that follow, observing that these partly align with k-equivariance.

As with associativity, preservation of the equivalence of a carry function's corresponding base addition to integer addition may also be limited to k digits. Formally, base addition (as defined in Equation 2) is equivalent to integer addition up to k digits if and only if f is a k-equivariant cocycle differing from the **1** carry function by a coboundary. For a more complete discussion of how to construct base addition with the formalism of group cohomology – including the role of cocycles, coboundaries, and k-equivariance – see Appendix A.2.

2.4. Classifying Carry Functions

In addition to their range of equivariance, carry functions may be characterized in a number of other ways that are useful both for categorizing them mathematically, and for understanding how easily they can be learned. Here, we focus on the identity of the values that are carried. Specifically, carry functions can be divided into those for which the same integer value is always carried (*Single Value* carry functions), and those for which different integers may be carried depending on the pair of digits (*Multiple Value* carry functions).

2.4.1. Single Value Carry Functions

A carry function is designated as *Single Value* if the same integer value is carried in all cases except when nothing (i.e., 0) is carried. The paradigmatic example is the **1** carry function introduced in Section 2.3; referencing Equation 1 and Equation 2, its functional form is given by

$$f_1(n, m) = \mathbb{1}_{n+m \geq b}. \tag{3}$$

This produces the base representation of the integers commonly taught in arithmetic. However, other Single Value carry functions are also possible, that we refer to as *alternative* Single Value carry functions. These use a different carry value. In general, any coprime to b can serve as an alternative Single Value carry, where *coprime* refers to any number $< b$ that is not a factor of b. We refer to a coprime to b as a *unit* $u \in \mathbb{Z}_b$ of b, any of which can be used in a Single Value carry function, that we refer to as a **U** carry function. This is because incrementing by a unit (u) covers all b elements of $\mathbb{Z}_b$ in just b increments. However, each does so in a different order.

To see this, first consider the **1** carry function (noting that 1 is a unit for any b): incrementing by 1 always covers all b elements of $\mathbb{Z}_b$ in just b increments. In this case, the increments follow the natural order of the integers in $\mathbb{Z}_b$ (see the first row of the example below).

In contrast, consider counting up to two digits in base $b = 3$. Since here, 2 is also unit u of b, it can be used in a **2** carry function, as shown in the example below. As a reference, the first row shows counting using the **1** carry function, in which each increment is by $(0, 1)$; the second row shows the **2** carry function, in which each increment is by $(0, 2)$:

1 carry function: $(0,0)$, $(0,1)$, $(0,2)$, $(1,0)$, $(1,1)$, $(1,2)$, $(2,0)$, $(2,1)$, $(2,2)$.

2 carry function: $(0,0)$, $(0,2)$, $(0,1)$, $(2,0)$, $(2,2)$, $(2,1)$, $(1,0)$, $(1,2)$, $(1,1)$.

Note that for both carry functions, all values of $\mathbb{Z}_b$ are covered exactly once in the least significant (rightmost) place before the second (leftmost) place is used; they are simply covered in a different order. Figures 1 and 2 show graphic representations of the carry functions for $b = 3$ and 4.

In general, every unit u can be used for an alternative Single Value carry function – that is associated with a different and equally valid ordering of $\mathbb{Z}_b$ – in which either 0 or u is carried for all pairs of digits. Importantly, all of the Single Value carry functions are ∞-equivariant (see Section 2.3.2 and Appendix A.2).

2.4.2. Multiple Value Carry Functions

Finally, we note that for bases beyond $b = 2$, there are carry functions that do not always carry the same digit, and that cannot be reduced to the **1** carry function in the way discussed above. We refer to these as Multiple Value carry functions, which comprise a more heterogeneous group than Single Value carry functions. To our knowledge, there is no full, formal characterization of the relative distribution of Multiple Value versus Single Value carry functions for a given base b. However, to gain some insight into how this scales with b, we have calculated that, for 2-digit addition, there are b^{b-2} equivalent carry functions (this can be shown by fixing a cocycle and counting the possible coboundaries; see Appendix A.2). Of these, $\varphi(b)$ are Single Value carry functions, and the remaining $b^{b-2} - \varphi(b)$ are Multiple Value carry functions. $\varphi(b) \leq b$, so the fraction of Multiple Value carry functions is at least $b^{b-2} - b$. Thus, the fraction of Single Value carry functions vanishes for large b.

In Section 3) we observe that there is a subset of Multiple Value carry functions that are notably lower in complexity than the remainder. We refer to these as *Low Dimensional Multiple Value carry functions*, that we characterize in Section 3), and return to in Section 5.1 where we consider their base representations and the extent to which they exhibit ∞-equivariance.

3. Quantitative Measures of Carry Functions

As noted above, the number of carry functions grows rapidly with b, and these are heterogeneous both with respect to the nature of their associated carries and extent of their equivariance. Here, we consider quantitative measures that characterize of carry functions, as means of categorizing them, and for use in considering the efficiency with which they can be learned. To do so, it is useful to represent carry functions in matrix form, as "carry tables".

3.1. Carry Tables

For a given base b, we can represent any carry function $f : \mathbb{Z}_b \times \mathbb{Z}_b \to \mathbb{Z}_b$ as a matrix $F \in (\mathbb{Z}_b)^{b \times b}$, where $F_{n,m} = f(n, m)$. The matrix F is referred to as a *carry table*, that provides a graphical representation of the carry function f in which each element designates the value that is carried (indicated by color) depending on the pair of digits indexed by the corresponding row and column. Figure 1 shows the 16 carry tables in base 4 (see Appendix B.1 for the carry tables in base 5). The Single Value carry functions are outlined in blue (the **1** carry function in the top left and the **3** carry function, bottom right), the Low Dimensional Multiple Value carry functions in orange,

and the Other Multiple Value carry functions in grey. A carry table's entries are indexed by $\{0, 1, ..., b-1\}$ from left to right, top to bottom. Thus, with this indexing, the **1** carry function is $F_{n,m} = 0$ if $n + m < b$ or 1 if $n + m \geq b$ (in agreement with Equation 3). While the other carry functions are all valid (i.e., producing equivalent base additions to that of the **1** carry function, at least for 2-digit numbers), their structure differs, as shown in the figure, and quantified in Section 3.

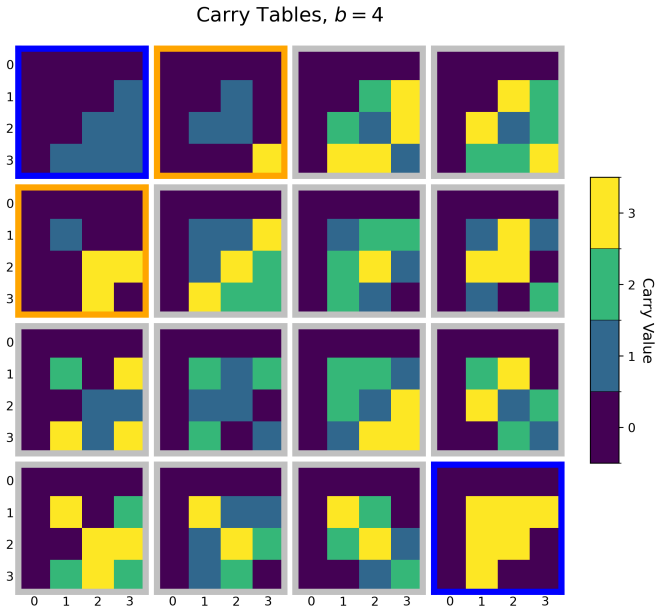

Figure 1: The 16 carry tables in base 4. The Single Value carry functions are outlined in blue (the **1** carry function in the top left and the **3** carry function, bottom right), the Low Dimensional Multiple Value carry functions in orange, and the Other Multiple Value carry functions in grey (see Section 2.4). Each table's entries are indexed by $\{0, 1, ..., b-1\}$ from left to right, top to bottom; color indicates the value that is carried (see legend at right).

As noted earlier, however, some carry functions are only finitely equivariant. For example, while the **1** carry function (top left of Figure 1) is ∞-equivariant (i.e., associative for triplets of any length), the Multiple Value carry function in the top right of Figure 1 is only 2-equivariant, and thus valid for only 2-digit numbers (i.e., not associative for all 3-digit triplets; e.g., $((0,0,1)+(0,0,2))+(0,0,3) = (3,2,2)$ whereas $(0,0,1)+((0,0,2)+(0,0,3)) = (0,2,2)$. While k-equivariant carry functions are, by definition, limited in their extension (i.e., the scale of their resulting base addition's equivariance), they may be of interest in settings where addition must learned, as solutions that may be discovered when the training set is constrained to numbers of length $\leq k$. We return to this consideration in Section 3.4.

To more precisely characterize the equivariance of carry functions, we define a *depth-k carry table* as $F_k \in (\mathbb{Z}_b)^{b^k \times b^k}$. The entries of F_k correspond to what is carried from the

k^{th} digit to the $k+1^{\text{th}}$ digit, such that, for $n = (n_k, ..., n_1)$ and $m = (m_k, ..., m_1)$ in the usual lexicographical ordering, $(F_k)_{n,m} = c_{k+1}$, for c_{k+1} as given in Equation 2. We refer to k as the *depth* of the table. Figure 2 shows depth-1, depth-2, depth-3 and depth-4 carry tables for base 3. For example, the table in the top, second from left panel of Figure 2 shows the depth-2 carry table for the **1** carry function, confirming that at depth 2 a 1 is carried from the second to third digit for sums of (1, 0, 0) or more. Note that this is true for all four depths shown, consistent with the fact that the **1** carry function is ∞-equivariant. From Figure 2 it is clear that, although all three carries are at least 4-equivariant, the **1** carry function has considerably simpler structure (is more compact) than the others, which becomes increasingly apparent at greater depths. We consider different ways of quantifying this in the sections that follow.

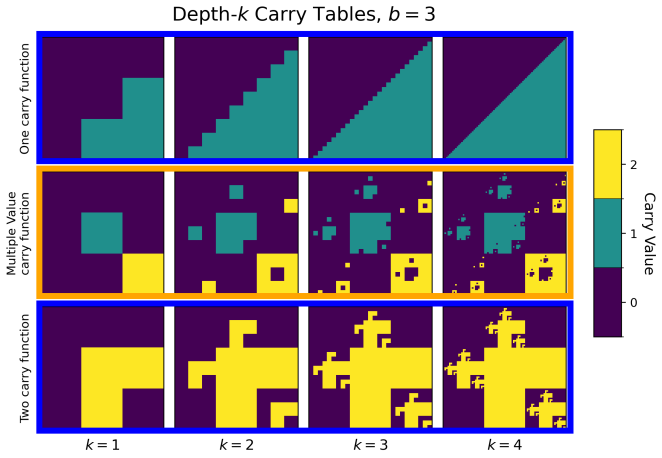

Figure 2: Structure at increasing depths of the three carry tables for $b = 3$. The panels in each row correspond to different carry functions, and those in columns show the depth-k carry tables for each carry function. As in Figure 1, the Single Value carry functions are outlined in blue (the **1** carry function in the top row and the **2** carry function in the bottom row) and the Low Dimensional Multiple Value carry function is outlined in orange (there are no Other Multiple Value carry functions for base 3). For depths $k = 1$ to 4 (columns), each depth-k carry table F_k (see Section 3.1) is indexed from left to right, top to bottom by $(n_k, ..., n_1)$ in the usual lexicographical ordering, and colors indicate the integer value carried for each pair of digits added. Note that the scale of each axis grows by a factor of b as k is increased.

3.2. Fractal Dimension

It is apparent that all of the depth-k carry tables shown in Figure 2 have at least partially fractal structure; that is, repeated, self-similar structure at progressively finer scales. This comports with the idea that a k-equivariant carry function has a self-similar structure up to some k digits (i.e., scales).

These observations suggest that fractal dimension may be a natural way to characterize the equivariance of carry functions, and to quantify their complexity. For example,

in the simplest case, as $k \to \infty$, the carry table for the **1** carry function converges to a triangle (see top row of Figure 2), whereas other carry functions exhibit more complex fractal structure.

Generally, definitions of dimension in fractal geometry take a measurement N_ϵ at scale ϵ (which ignores granularity at scales $< \epsilon$), and observe how the value changes as $\epsilon \to 0$. Though the most widely-used fractal dimension is the Hausdorff dimension, here we use the simpler box-counting dimension, given its natural application to carry tables. Letting N_ϵ be the number of boxes of side-length ϵ required to cover the fractal, the box-counting dimension d_{box} is defined as

$$d_{\text{box}} = \lim_{\epsilon \to 0} \frac{\log(N_\epsilon)}{\log(1/\epsilon)}. \tag{4}$$

Notice that Equation 4 can be re-written as $N_\epsilon \approx C(1/\epsilon)^{d_{\text{box}}}$ for constant C, which reduces to the usual notion of dimension for integral d_{box} (see Falconer (2014) for further discussion).

For present purposes, we measure the box-counting dimension of the border of the depth-k carry table in the limit $k \to \infty$; indeed, a table's complexity might correspond to the boundary between distinct carries (comparatively, the box-counting dimension of the interior corresponds to the frequency of carrying in the limit $k \to \infty$; see Section 3.3). To avoid over-counting, we define the border as those entries $(F_k)_{n,m}$ with different values than their left and upper neighbors. Box-counting dimension is natural for the case of carry tables because the "resolution" of the carry table at a given depth k is exactly boxes of width $\epsilon = 1/b^k$ (see Figure 2). Therefore, it suffices to count the number of entries on the border to measure N_ϵ at depth k, and then compute Equation 4 by setting $\epsilon = 1/b^k$. For example, for the **1** carry function at depth 2 (top, second from left of Figure 2), we have $N_\epsilon = 8$ and $\epsilon = 1/3^2 = 1/9$, giving an estimated $d_{\text{box}} \approx 0.946$.

Importantly, as the conventional ordering of $\mathbb{Z}_b$ is just one of the valid orderings generated by its alternative Single Value carry functions (see Section 2.4.1), we consider the minimal box-counting dimension across all such orderings of the depth-k carry table indices. Figure 3A shows the estimated box-counting dimensions for bases $b = 3$, 4, 5 at depths $k = 1$ to 4.

3.3. Frequency of Carrying

Another way to quantify differences between carry functions, that complements its fractal dimension, is how often a carry must be made. For the carry at digit k, this reduces to the fraction of non-zero numbers in the depth-k carry table. To measure the overall frequency of carrying, it suffices to average the frequency of carrying at each digit. Figure 3B shows the overall frequency of carrying for bases $b = 3$, 4, 5 at depths $k = 1$ to 4.

3.4. Associativity Fraction

Finally, we quantify carry functions according to the validity of their resulting base addition, measured by their associativity (see Section 2.1). Specifically, we consider the fraction of triplets for which a carry function's resulting addition is associative, up to varying depths. That is, for each depth k, we sample triplets (n, m, p) of length $k + 1$ and measure the fraction that satisfy $(n + m) + p = n + (m + p)$, using the addition

procedure of Equation 2. Note that this measure is closely related to k-equivariance (see Section 2.3), since a k-equivariant carry function has an associativity fraction of 1 up to depth $k-1$. Figure 3C shows the associativity fraction of carry functions for bases $b = 3$, 4, 5 at depths $k = 1$ to 4.

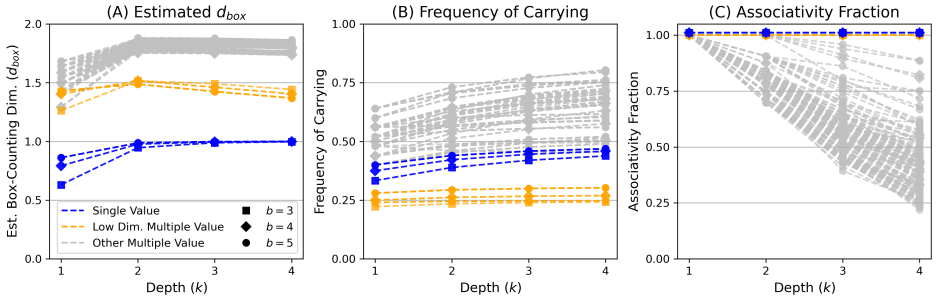

Figure 3: Quantitative characterization of different carry functions for bases 3 to 5 and depths 1 to 4, using three different measures: (A) estimated box-counting dimension of the fractal depth-k table border; (B) frequency of carrying; and (C) associativity fraction. Each is shown for bases $b = 3$ (as squares), 4 (diamonds), and 5 (circles) at depths $k = 1$ to 4. For each quantitative measure, as the depth increases the carry functions separate into the classes of the Single Value carry functions and Multiple Value carry functions. Furthermore, the Low Dimensional Multiple Value carry functions separate from the Single Value and other Multiple Value carry functions according to frequency of carrying as well as box-counting dimension. Note that, for associativity fraction, the Low Dimensional Multiple Value carry functions exactly underlie the Single Value carry functions (i.e., they all perfectly preserve associativity up to depth 4, and hence are 5-equivariant).

Figure 3 shows that each quantitative measure reflects the categorizations considered in Section 2.4 (a correlation matrix comparing the measures is provided in Appendix B.2). First, at better estimates of the box-counting dimension (in particular, at depth $k = 4$), the Single Value carry functions converge to a dimension of 1, whereas the Multiple Value carry functions all maintain dimension > 1.25. Among the Multiple Value carry functions, a subset converge to a dimension > 1.25 but < 1.5, corresponding to the Low Dimensional Multiple Value carry functions introduced in Section 2.4.2. The Low Dimensional Multiple Value carry functions can also be distinguished from the other Multiple Value carry functions in the other quantitative measures: they have a lower frequency of carrying than not only the other Multiple Value carry functions, but also the Single Value carry functions; and they have associativity fraction of 1 up to at least depth $k = 4$. In the section that follows, we consider how these factors influence the efficiency with which carry tables can be learned by a neural network.

4. Neural Network Simulations

As noted in Section 1, humans exhibit a remarkable ability for radical generalization, as exemplified by their ability to carry out base arithmetic algorithmically, over

an arbitrary number of digits. This suggests that its implementation in the brain must respect the symmetry properties of base addition considered in this paper. Here, we explore how the factors discussed in the previous sections, and their interaction with training procedures, may impact how neural networks discover the symmetries underlying base addition through learning. We begin by formulating the base addition problem in a format suitable for training a neural network, and describe the network architecture used for training and testing. We then consider how the embedding of integers, training curriculum, and characteristics of various carry functions impact learning and generalization. Code for the network architecture and simulations using it are freely available at `https://github.com/cutterdawes/BaseAddition`.

4.1. Model Representations, Architecture, and Procedures

4.1.1. Stimulus Representations

We formulated the addition problem for k-digit non-negative integers in base $b \in \mathbb{N}$, by constructing two multi-digit numbers, $n = (n_k, ..., n_1)$ and $m = (m_k, ..., m_1)$, which were composed of lists of b dimensional tensors representing each digit, ordered left to right from most significant $(_k)$ to least significant $(_1)$. We considered two forms of digit representation (or *embedding*): (i) a purely *symbolic* embedding, in which each digit was represented as a different one-hot tensor, so that all were orthogonal to one another; and (ii) a *semantic* embedding, in which numbers closer in ordinal value were more correlated with (similar to) one another than ones further apart (see Section 4.2). The two numbers n and m for a given problem could differ in length, in which case the shorter one was zero-padded from the left to match the length of the longer one.

4.1.2. Network Architecture

We were specifically interested in how a neural network can solve arithmetic problems when constrained to do so using serial processing, comparable to how humans do so. We assume that this imposes a strong inductive bias favoring discovery of the symmetry properties of carry functions considered in the previous sections, advantaged by weight sharing over time inherent to recurrent neural networks. Accordingly, the model was comprised of a single-layer GRU (input dim. b, hidden dim. b; Cho et al., 2014) followed by a single linear layer (input dim. b, output dim. b) used for decoding.

For all experiments, we also used a model with a single-layer LSTM (input dim. b, hidden dim. b; Hochreiter and Schmidhuber, 1997) in place of a GRU; refer to Appendix C.1 for details.

4.1.3. Problem Format

There are two distinct ways in which an addition problem can be presented. In the most straightforward, all the digits of one of the numbers are presented first, in standard (most significant to least significant) order, followed by all the digits of the other number, after which the agent is expected to produce the result, also with digits ordered from most significant to least significant. While this format is compact and familiar, it is also extremely difficult to learn, in part because it requires keeping track of and reordering the digits at encoding and response (Segert, 2024).[7] While this is an important capability,

[7] Hence, neural network solutions to this format may require the use of some form of external memory.

our focus here was on how the arithmetic computations themselves are performed. This format is also not the way humans learn to perform multi-digit arithmetic (perhaps for a similar reason). Rather, they do so in what we refer to as an *interleaved* format, that we used in our simulations.

In the interleaved format, one digit from each number is presented, beginning with the least significant; the numbers are added; the result is generated, and then the next-most significant set of digits is presented, etc. That is, the input sequence x is

$$x = (n_1, m_1, *, n_2, m_2, *, ..., n_k, m_k, *),$$

with special tokens denoting the (heldout) digits of the answer sequence $s = (s_k, ..., s_1)$. In human schooling, this format is implemented by placing one full number above the other, right-aligned by the least significant digit, thus allowing addition of the numbers in each column, from left to right. The problem then reduces to performing modular addition for the numbers in each column using b as the modulus, generating the result, and carrying appropriately to the next column (i.e., implementing a carry function). We simulated this procedure for the neural network by presenting it with the digits from each place, one pair at a time, in order from least to most significant, and requiring the network to generate the resulting digit for each place. Accordingly, it had to learn both the base-appropriate modular addition, encode the appropriate carry in memory, and then include that in the next addition operation. We trained it using the correct responses for different carry functions, and examined its ability to learn these as a function of the characteristics of carry functions considered in the previous section.

4.1.4. Training and Testing Procedures

Separate networks were trained to add 3-digit numbers using each possible carry function for bases $b = 3$ to 5. The loss was computed as the cross-entropy between the output logits in the response layer and the answer sequence determined by the corresponding carry function. Each network was trained for 2500 epochs over all 3-digit tuples (n, m), with a batch size of 32 and a learning rate of 0.05 using the Adam optimizer. This was done for 10 initializations of each network. To track learning and test generalization, we evaluated each network's performance on both 3-digit and 6-digit numbers every 10 epochs throughout training.

4.2. Results

Figure 4 shows the training loss as well as training and testing accuracy for bases 3 to 5, averaged for 10 runs of each configuration as noted above. These results indicate that the model learned the Single Value carry functions and Low Dimension Multiple Value carry functions significantly more effectively than the other Multiple Value carry functions.

4.2.1. Embedding

The results shown in Figure 4 are for symbolic (one-hot) embeddings of digits, in which there are no meaningful representational differences among the Single Value carry functions (see Section 2.4.1). This reflects the treatment of digits as symbolic labels, without any meaningful ordering. However, whereas numbers are symbolic in one sense (that is, they can be used to refer to the quantify of any entity), they also have ordinal

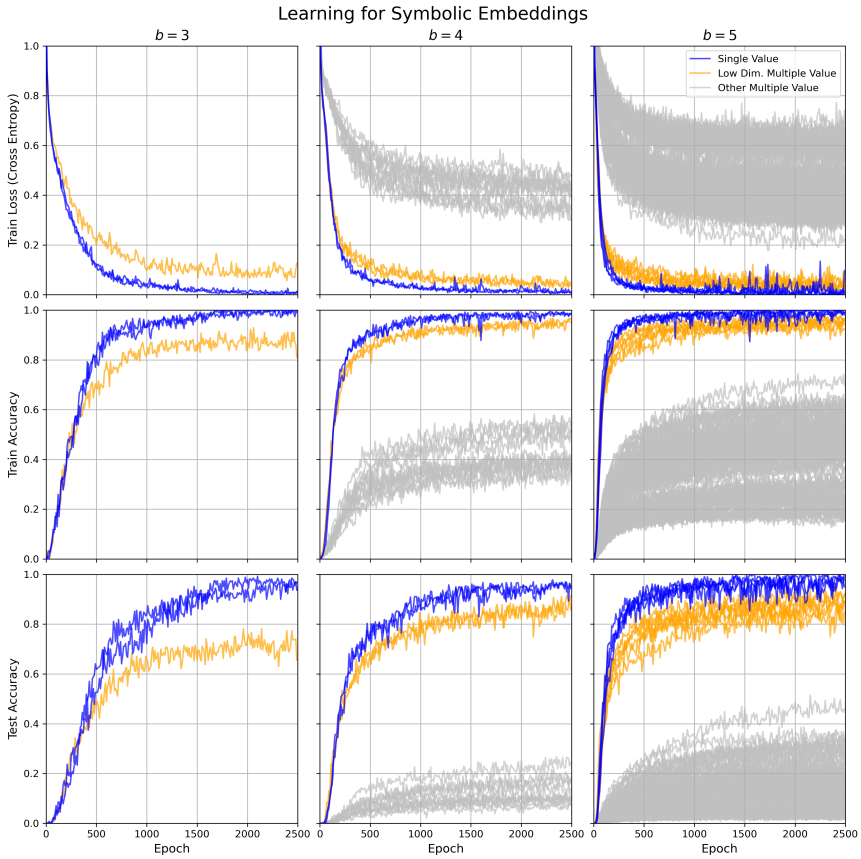

Figure 4: Learning for symbolic (one-hot) embeddings of digits. Performance over the course of training (averaged over 10 runs of each model implementation) for addition on different carry functions for bases $b = 3$ to 5.

relationships, that humans learn prior to arithmetic. To investigate how such information impacts the learning of different carry functions, we tested semantic embeddings of digits that reflected their ordinal relationships. We did so by ordering one-hots to align with the ordinal value of each digit, and then convolving each with a Gaussian envelope mean-centered on its one-hot value (e.g., for $b = 5$, the digit 1, formerly represented as the one-hot vector (0, 1, 0, 0, 0), became (0.2, 0.5, 0.2, 0.05, 0.05)). Accordingly, each representation was now partially correlated with its immediate neighbors, and progressively less so with more distant ones. Furthermore, because $\mathbb{Z}_b$ is cyclic (fundamental to its structure), we applied the convolution in circular form, so that so that 0 was equidistant to 1 and $b-1$. We encoded alternative orderings of $\mathbb{Z}_b$ similarly (see Section 2.4.1).[8]

[8]Note that, due to the cyclic structure of $\mathbb{Z}_b$, a unit and its inverse produce the same ordering: in $b = 5$ for example, the unit 4 (inverse of 1) produces the ordering (0, 4, 3, 2, 1), which produces the

For example, corresponding to the ordering that results from incrementing by the unit 2 for $b = 5$ (0, 2, 4, 1, 3), we encoded 1 as (0.05, 0.5, 0.05, 0.2, 0.2). To examine how different orderings impact the learning of Single Value carry functions, we trained different implementations of the same model with the Gaussian-convolved one-hots corresponding to each ordering for that base (for 10 initializations of each network).

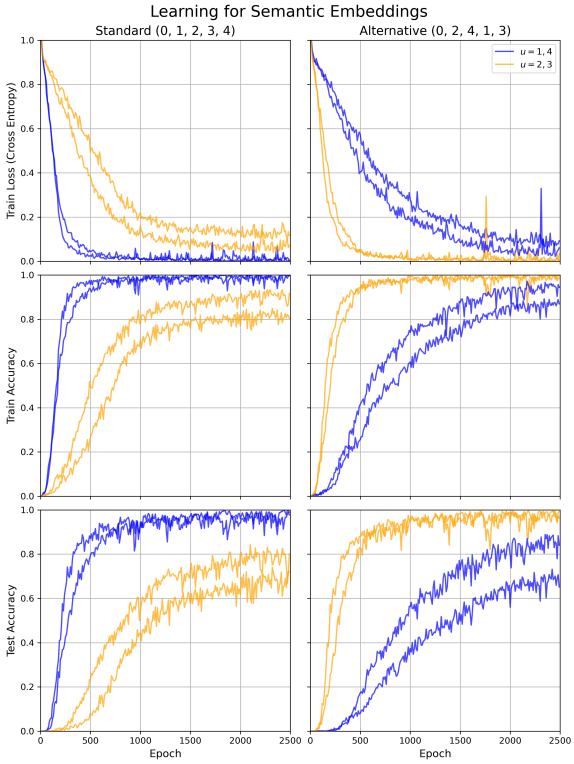

Figure 5: Learning for semantic embeddings of digits. Performance over the course of training on Single Value carry functions with order-encoded inputs in the two non-degenerate orderings (0, 1, 2, 3, 4) and (0, 2, 4, 1, 3) for $b = 5$ (see Section 2.4.1 for how Single Value carry functions correspond to orderings of $\mathbb{Z}_b$).

Figure 5 shows the training results for semantic embeddings using $b = 5$, for which there are four Single Value carry functions. These separate clearly into two groups, each corresponding to an ordering of $\mathbb{Z}_b$: one that includes the standard **1** carry function (involving incrementing by 1; so 0, 1, 2, 3, 4) and its inverse, the **4** carry function; and a second set involving the **2** carry function (involving incrementing by 2; so 0, 2, 4, 1, 3) and its inverse, the **3** carry function. Interestingly, for the semantic embeddings of each ordering, the model learned the set of Single Value carry functions corresponding to that

same semantic embeddings as (0, 1, 2, 3, 4).

ordering more quickly, indicating that it is comparatively easier to learn a carry function that reflects the ordinal structure of the digits (encoded in their embeddings) – despite no difference in structure according to the quantitative measures of Section 3.

4.2.2. Generalization

The previous experiments tested generalization on 6-digit numbers, compared to the 3-digit numbers seen in training. To probe generalization further out of domain, we tested accuracy on numbers up to 10 digits. That is, for each carry function, we trained the model on 3-digit numbers and then tested on k-digit numbers for each $k \in [3 : 10]$, averaged over 10 training/testing runs.

In particular, we did this for symbolic embeddings of digits in bases 3 to 5, shown in Figure 6. Across all lengths and bases, the same separation between Single Value, Low Dimensional Multiple Value, and Other Multiple Value carry functions that was evident in the previous experiments persisted, and even became more apparent as the number of digits increased. This corroborated that: (i) when learning a carry function of sufficient symmetry (viz., Single Value), a very small model can generalize far out of domain; and (ii) the symmetry of the learned carry function is increasingly important to performance as we move further from the training distribution.

Figure 6 also reveals a somewhat surprising trend: in each class of carry functions (Single Value, Low Dimensional Multiple Value, Other Multiple Value), out-of-domain performance increases as the base increases from 3 to 5. We discuss possible reasons for this trend in Section 5.2.2. However, not surprisingly, carry function structure also has an influence on learning, as we consider next.

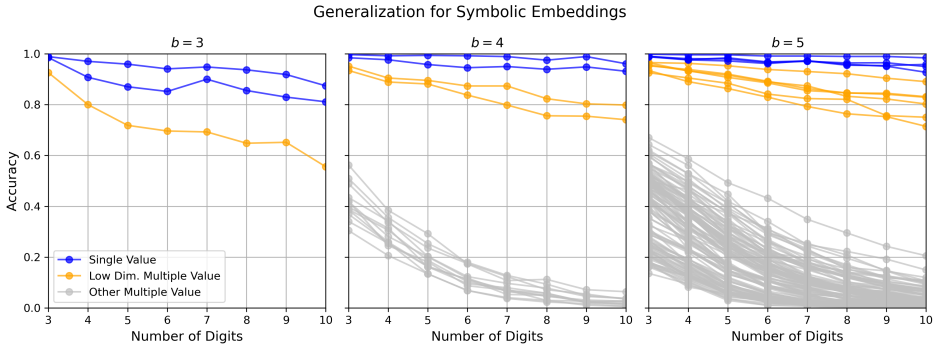

Figure 6: Out-of-domain generalization for symbolic embeddings of digits. For each carry function in bases $b = 3$ to 5, accuracy was tested on k-digit numbers for each $k \in [3 : 10]$ after training on 3-digit numbers (averaged over 10 training/testing runs).

4.2.3. Carry Function Structure

To evaluate the influence of carry function structure on learning, we measured the correlation between maximum testing accuracy and the quantitative measures described in Section 3 across the carry functions of bases $b = 3$ to 5. Figure 7 shows scatter plots of maximum testing accuracy (on 6-digit numbers) and the three quantitative measures:

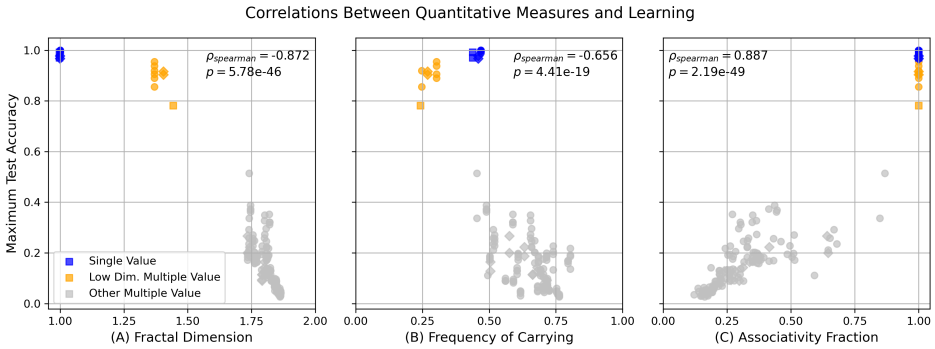

Figure 7: Relationship of carry function structure and learning, as measured by maximum testing accuracy (on 6-digit numbers). Scatter plots showing relationship of learning curves to structure measured as (A) fractal dimension, (B) frequency of carrying, and (C) associativity fraction for different carry functions, divided into three categories: Single Value carry functions (blue), Low Dimensional Multiple Value carry functions (orange), and other Multiple Value carry functions (grey). The base b of each carry function is indicated by shape ($b = 3$: squares; $b = 4$: diamonds; and $b = 5$: circles). Spearman's rank correlations (over bases $b = 3$ to 5) and significance are shown for each plot.

fractal dimension, frequency of carrying, and associativity fraction. The Spearman rank correlation was strongly significant for each measure (-0.872, -0.656, and 0.887 for fractal dimension, frequency of carrying, and associativity function, respectively). The correlations suggest that fractal dimension and frequency of carrying have a negative influence on learning, and associativity fraction a positive influence. Furthermore, across all measures, there was a clear separation into the same three groupings of Single Value carry functions, Low Dimensional Multiple Value carry functions, and other Multiple Value carry function. We discuss these findings further in Section 5.

For robustness, we used two additional measurements of learning: the upper asymptote and critical point of a sigmoid function fitted to the testing accuracy. For more details including correlations between these metrics and the quantitative measures, see Appendix C.2.

5. Discussion

In this paper, we presented a group theoretic analysis of base addition, that identified two fundamental classes of carry functions: Single Value and Multiple Value. We quantified their structure using measures of fractal dimension, frequency of carrying, and associativity fraction and found, across all measures, a clear separation between Single Value and Low Dimensional Multiple Value versus other Multiple Value carry functions. In neural network simulations, these separated into three groups that aligned with the three types of carry functions with respect to the effectiveness with which they were learned. Here, we discuss the implications of these results in terms of mathematics and neural networks.

5.1. Mathematics

The analyses reported here extend a nascent line of work considering base arithmetic from a group theoretic perspective. Specifically, they help formalize and characterize the nature of the symmetries underlying base addition, that can serve as a foundation for further work that extends the analysis in a number of ways. Most proximally, questions remain about the nature of Multiple Value carry functions: do these correspond to some factorization of the base representation, and which if any are ∞-equivariant? For Low Dimensional Multiple Value carry functions, all of which are all at least 5-equivariant (see Figure 3C), is there a factorization of their base representations that can explain their differences from the other Multiple Value carry functions and, more generally, is the scale of a base addition's equivariance related to some factorization of its associated base representation? Extending our analyses to the more general case of addition (i.e., over arbitrary numbers of addends, that may involve more complex forms of carry), and to other arithmetic operations are also important directions for future work.

As noted in Section 1, characterizing base arithmetic in group theoretic terms provides an opportunity for formalizing our understanding of how human cognition exploits symmetries for generalization. The work presented here, on base addition, should be directly relevant to a formal understanding of factors such as translational invariance. More broadly, it would be interesting to explore the extent to which other basic mathematical operations that are central to human cognitive function – such as rotations, processing of hierarchical structures (such as trees), and context-normalization – can build on the work presented here and/or be addressed using a similar approach.

5.2. Neural Networks

5.2.1. Symmetry Discovery

Notable among the results in Section 4 is the capacity for a very small model (1-layer GRU) to learn addition with all of the Single Value carry functions and generalize to considerably more digits than the number on which it was trained. For all of these carry functions, the neural network achieved near-perfect accuracy on numbers with up to twice as many digits (6) as the training set (3), and maintained high accuracy on numbers with as many as 10 digits. This no doubt reflects the underlying symmetry of the carry functions, and likely the ability of a recurrent neural network to discover that given the inductive bias of weight sharing over time, a factor we discuss in the sections that follow. It is also a direct result of the way in which the problem was constructed and presented to the network, a factor that we discuss further in Section 5.2.3. Collectively, these factors may explain where and how the human brain supports the radical generalization abilities evident in human cognition.

5.2.2. Structure and Learning

The results presented in Section 4 indicated that the effectiveness with which a neural network learns a carry function is directly impacted by several quantifiable features of its structure. The most obvious and straightforward is the complexity of the function, measured as its fractal dimension. This is not surprising, as it is to be expected that a neural network finds it more difficult to learn more complex functions (Loukas et al., 2021; Goldblum et al., 2023). This may also explain why the carry function used universally by humans – the **1** carry function – is a Single Value carry function: they are the simplest

to learn. Furthermore, the simulations reported in Section 4.2.1 suggest why the **1** carry function in particular is so universal: not only is it a Single Value carry function, but it is also aligned with the ordinal structure of numbers. The latter must be encoded using some form of semantic structure, which, as shown in Figure 5, makes it comparatively easier to learn than other Single Value carry functions.

We also found that, for the more easily learned carry functions – Single Value and Low Dimensional Multiple Value – there was a positive relationship between frequency of carries and the maximum testing accuracy (evidenced by the blue and orange dots in Figure 7B). This is not surprising, insofar as higher frequency of occurrence provides more opportunities to learn a given operation. Interestingly, however, this was not so for the more complex Multiple Value carry functions which, despite considerably higher frequencies of carry, exhibited considerably longer learning times. This is likely due to the fact that, although carries are frequent in those functions, the specific values being carried vary considerably, thus diminishing the benefits of repetition.

Perhaps most importantly, the carry functions with greatest associativity — that is, the most compact forms of symmetry – are learned the fastest, as shown in Figure 7C. This could provide a formal account for why humans universally use the **1** carry function, and more generally for the importance of symmetries for learning in neural networks.

Finally, we return to the observation, in Figure 6, that accuracy improved with base size. On the surface, this might seem surprising, as carry tables for larger bases might be expected to be more complex and therefore more difficult to learn. However, a simple scaling analysis reveals that, while the complexity of modular addition and the carry function are upper-bounded by b^2 (i.e., memorization for each pair), the number of training examples scales with b^6 (i.e., all pairs of 3-digit numbers). Hence, the benefit of more training examples may outweigh the greater difficulty of learning the addition function in that base, by leveraging the underlying symmetries for more effective learning – much as people seem to do for base 10.

5.2.3. Curriculum

As discussed in Section 1, the way in which a network is trained can be as important an inductive bias as the architecture itself (e.g., recurrence) for discovering symmetries. Here, we exploited this by presenting the data to the network in a format that made the symmetry as apparent and accessible as possible: one digit at a time, from least to most significant. This made it possible for the network to learn the modulus arithmetic and carry operations, without having to discover the relevance of and learn to manage digit order. However, this leaves open the question of how a network may learn to perform base arithmetic when this is presented in a less constrained form, such as the most familiar notation to humans (in which all of the digits of one number are presented before the other). This would require it to parse the problem into a form that exposes the underlying symmetry, as well as learning to perform the other ancillary operations, such as appropriately ordering the digits for processing, caching intermediate values, and reordering them for the response. Discovering and coordinating the use of these functions turns out to be a challenging problem for neural networks, with no clear evidence that even the largest current models are able to do so in a way that supports fully algorithmic base arithmetic (Nogueira et al., 2021; Zhou et al., 2022; Qian et al., 2022; Ebrahimi et al., 2024). This suggests that other architectural elements may be important, and even necessary as inductive biases for the discovery and implementation of such functions, such

as access to an external memory and position coding. Work in cognitive science suggests that humans make use of such functions in problems that involve memory, sequential processing, and numerical comparisons (e.g., Howard and Kahana (2002); Beukers et al. (2024)). Integrating these into the model we have presented here for discovery of the symmetries underlying base arithmetic offers a promising direction for future research.

6. Summary and Conclusions

In this paper, we provide a formal group theoretical analysis of base addition in terms of the symmetry properties of underlying carry functions. We quantify three features of such carry functions with respect to multi-digit addition: their complexity (measured as fractal dimension), the frequency of carries, and the compactness of the underlying symmetry functions (or their degree of equivariance). In all cases, we find a close relationship between these properties and the ease with which a neural network can learn the corresponding functions, and discuss how these observations may provide insights into why humans universally use a particular form of carry – the **1** carry function – in base addition: it is the least complex, with highest frequency of carries, and exhibits the most compactness (highest degree of equivariance). This is supported by the observation that this is the form of carry more quickly and effectively learned in a simple recurrent neural network. We also consider the importance of the format in which training is carried out, again providing evidence that neural networks learn most effectively under the same training paradigm that is most commonly used to teach multi-digit addition to humans.

More generally, the work presented here provides an example of how the learning of underlying symmetries undergirds the capacity for radical generalization that is one of the defining characteristics of human cognitive function. Our approach offers a formal framework for characterizing such symmetries, and studying how they can be learned in neural networks. In particular, it illustrates how inductive biases – both structural (such as recurrence) and curricular (here, the interleaved training protocol) can help make symmetries much more accessible for discovery, thus greatly enhancing the sample efficiency of the network. Such insights promise to be valuable both for understanding how people learn and make use of other forms of symmetry structure, as well as the design of artificial systems that do so more efficiently that existing systems.

Acknowledgements

The authors would like to thank Mark McConnell for helping to formalize the mathematical construction of base addition, as well as the following individuals for useful conversations and suggestions regarding the work throughout this manuscript: Taylor Webb, Zack Dulberg, Adel Ardalan, Tim Buschman, and Shanka Mondal. SS was supported by a T32 Training Grant in Computational Neuroscience (T32MH065214), and KK was supported by a C. V. Starr fellowship from the Princeton Neuroscience Institute and a CPBF (Center for the Physics of Biological Function) fellowship (through NSF PHY-1734030). This work was supported in part by a Vannevar Bush Faculty Fellowship from the Office of the Under Secretary of Defense for Research & Engineering (ONR N00014-22-1-2002) awarded to JDC.

References

Anderson, P.W., 1978. Local moments and localized states. Science 201, 307–316.

Beukers, A., Hamin, M., Norman, K.A., Cohen, J.D., 2024. When working memory may be just working, not memory. Psychological Review 131, 563.

Brown, K., 2012. Cohomology of Groups. Graduate Texts in Mathematics, Springer New York.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 .

Cleeremans, A., McClelland, J.L., 1991. Learning the structure of event sequences. Journal of Experimental Psychology: General 120, 235.

Dehaene, S., 1997. The Number Sense: How the Mind Creates Mathematics. Oxford University Press, New York.

Ebrahimi, M., Panchal, S., Memisevic, R., 2024. Your context is not an array: Unveiling random access limitations in transformers. arXiv preprint arXiv:2408.05506 .

Elman, J.L., 1990. Finding structure in time. Cognitive Science 14, 179–211. doi:https://doi.org/10.1207/s15516709cog1402_1.

Falconer, K., 2014. Fractal Geometry: Mathematical Foundations and Applications. John Wiley & Sons.

Goldblum, M., Finzi, M., Rowan, K., Wilson, A.G., 2023. The no free lunch theorem, kolmogorov complexity, and the role of inductive biases in machine learning. arXiv preprint arXiv:2304.05366 .

Hochreiter, S., Schmidhuber, J., 1997. Long Short-Term Memory. Neural Computation 9, 1735–1780. doi:10.1162/neco.1997.9.8.1735, arXiv:https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf.

Howard, M.W., Kahana, M.J., 2002. A distributed representation of temporal context. Journal of mathematical psychology 46, 269–299.

Isaksen, D., 2002. A cohomological viewpoint on elementary school arithmetic. The American Mathematical Monthly .

Ji-An, L., Benna, M.K., Mattar, M.G., 2024. Discovering cognitive strategies with tiny recurrent neural networks. bioRxiv URL: https://www.biorxiv.org/content/early/2024/10/05/2023.04.12.536629, doi:10.1101/2023.04.12.536629, arXiv:https://www.biorxiv.org/content/early/2024/10/05/2023.04.12.536629.full.pdf.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, Curran Associates, Inc.

Lecun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324. doi:10.1109/5.726791.

Loukas, A., Poiitis, M., Jegelka, S., 2021. What training reveals about neural network complexity. Advances in Neural Information Processing Systems 34, 494–508.

Nogueira, R., Jiang, Z., Lin, J., 2021. Investigating the limitations of transformers with simple arithmetic tasks. arXiv preprint arXiv:2102.13019 .

Piantadosi, S.T., 2023. The algorithmic origins of counting. Child Development 94, 1472–1490.

Qian, J., Wang, H., Li, Z., Li, S., Yan, X., 2022. Limitations of language models in arithmetic and symbolic induction. arXiv preprint arXiv:2208.05051 .

Segert, S., 2024. Maximum Entropy, Symmetry, and Relational Bottleneck: Unraveling the Impact of Inductive Biases on Systematic Reasoning. Ph.D. thesis. Princeton University.

Segert, S., Cohen, J., 2022. A self-supervised framework for function learning and extrapolation. Transactions on Machine Learning Research URL: https://openreview.net/forum?id=ILPFasEaHA.

Webb, T., Holyoak, K.J., Lu, H., 2023. Emergent analogical reasoning in large language models. Nature Human Behavior .

Wynn, K., 1992. Addition and subtraction by human infants. Nature 358, 749–750.

Yang, Y., Campbell, D., Huang, K., Wang, M., Cohen, J., Webb, T., 2025. Emergent symbolic mechanisms support abstract reasoning in large language models. URL: https://arxiv.org/abs/2502.20332, arXiv:2502.20332.

Zhou, H., Nova, A., Larochelle, H., Courville, A., Neyshabur, B., Sedghi, H., 2022. Teaching algorithmic reasoning via in-context learning. arXiv preprint arXiv:2211.09066 .

Appendix A. Supplement to Section 2: Mathematics of Symmetry and Base Addition

Appendix A.1. Supplemental Group Theory

Two foundational concepts of group theory relevant to this work are *homomorphism* and *isomorphism*: Given two groups G and G', a group homomorphism is a mapping $f : G \to G'$ that preserves the structure of both groups (i.e., $f(g \cdot h) = f(g) \cdot f(h)$ for all $g, h \in G$); and a group isomorphism is a homomorphism that is also bijective (i.e., forms a one-to-one mapping). Homomorphisms and isomorphisms formalize what it means for two groups to have similar or the same structure, respectively, and hence allow for comparing and classifying groups. For example, let G' be the group of integers modulo 3, with elements $0, 1, 2$ and a group operation of addition modulo 3 (e.g., $2+2 = 1$). In Section 2.1, we mention that G' is isomorphic to the rotational group of the equilateral triangle (which we denote by G). To see this, let $f : G \to G'$ map the rotations of $0°, 120°, 240°$ clockwise to $0, 1, 2$, respectively. This reflects that, though seemingly entirely different on the surface, the two groups are different expressions of the same underlying symmetry.

Another important construct in group theory is that of a *group action*, in which G "acts" on a set X in a way that respects the structure of G. For example, the rotational group of the triangle may act on the Euclidean plane by rotation about the origin. The *orbit* of an element $x \in X$ denotes the subset of X that x is sent to by all $g \in G$; e.g., the orbit of $(1, 0)$ in the preceding example is $\{(1, 0), (-\frac{1}{2}, \frac{\sqrt{3}}{2}), (-\frac{1}{2}, -\frac{\sqrt{3}}{2})\}$. Similarly, the orbit $G \cdot A$ of a subset $A \subset X$ denotes the subset of X to which all $x \in A$ are sent by all $g \in G$. A group action serves to separate a space's symmetry from the space itself, and provides a framework with which to compare different spaces with the same symmetry.

Appendix A.2. Formal Derivation of Base Addition and Carry Functions

In this appendix, we derive the formalism underpinning base addition.

Choose a base $b \in \mathbb{N}_{\geq 2}$, and denote the integers modulo b by $\mathbb{Z}_b$. For any integer $n \in \mathbb{Z}$, we wish to present it in a *base representation*; i.e., we seek a tuple $(n_k, n_{k-1}, ..., n_1)$, where $n_j \in \mathbb{Z}_b$ is the j^{th} digit for $j \in [k]$. To properly preserve the structure of $\mathbb{Z}$, we will construct base representations with use of concepts from group cohomology, following Isaksen (2002). Note that the material and exposition in Appendix A.2.1, Appendix A.2.2, and Appendix A.2.3 closely follows Brown (2012), which establishes the formalism for the 2-digit case. In Appendix A.2.4, we iteratively extend to arbitrary digits.

Note: Throughout, in a slight abuse of notation (and because we only consider abelian groups), we use "+" to jointly refer to group operations of different groups, "$-n$" to refer to inverse of n, and "$m \cdot n$" to refer to the multiple $n = \underbrace{n + \cdots + n}_{m \text{ times}}$.

Appendix A.2.1. Group Extensions

To start, we seek to represent $n \in \mathbb{Z}_{b^2}$ as a 2-digit number $(n_2, n_1) \in \mathbb{Z}_b \times \mathbb{Z}_b$. We may approach this as a group extension problem, framing $\mathbb{Z}_{b^2}$ as an extension of $\mathbb{Z}_b$ by itself. Thus, we have the short exact sequence

$$0 \to \mathbb{Z}_b \xrightarrow{i} \mathbb{Z}_{b^2} \xrightarrow{\pi} \mathbb{Z}_b \to 0, \tag{A.1}$$

where $i : \mathbb{Z}_b \to \mathbb{Z}_{b^2}$ is the inclusion map of the second digit and $\pi : \mathbb{Z}_{b^2} \to \mathbb{Z}_b$ is the projection map to the first digit.

To properly construct a base representation, we seek an extension equivalent to $\mathbb{Z}_{b^2}$ defined on the set $\mathbb{Z}_b \times \mathbb{Z}_b$; i.e., an appropriate group law on $\mathbb{Z}_b \times \mathbb{Z}_b$ (we denote the resulting group by $\mathbb{Z}_b \times_f \mathbb{Z}_b$), new inclusion and projection maps $\hat{\imath}$ and $\hat{\pi}$, and an isomorphism $\phi_s : \mathbb{Z}_b \times_f \mathbb{Z}_b \to \mathbb{Z}_{b^2}$ making the diagram in Figure A.8 commute (we explain these notations in Appendix A.2.2).

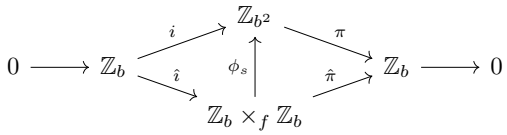

Figure A.8: The base b representation $\mathbb{Z}_b \times_f \mathbb{Z}_b$ is equivalent to $\mathbb{Z}_{b^2}$ with proper choice of section s and cocycle f (see Appendix A.2.2).

Appendix A.2.2. Constructing an Equivalent Extension

We seek to construct a group law on $\mathbb{Z}_b \times \mathbb{Z}_b$ such that it is equivalent to $\mathbb{Z}_{b^2}$. We will see that this reduces to choosing a way to "carry" from the first to second digits when adding, which we will define in purely group cohomological terms.

First, choose a set-theoretic cross-section of the projection π, i.e., a function $s : \mathbb{Z}_b \to \mathbb{Z}_{b^2}$ such that $\pi s = \mathrm{id}_{\mathbb{Z}_b}$. Assume that s satisfies the *normalization condition*, $s(0) = 0$. Note that s may or may not be a homomorphism; to measure the failure of s to be one, we define $f : \mathbb{Z}_b \times \mathbb{Z}_b \to \mathbb{Z}_b$ by

$$s(n) + s(m) = i(f(n, m)) + s(n + m).$$

Notice f is well-defined because, for any $n, m \in \mathbb{Z}_b$, $s(n) + s(m)$ and $s(n + m)$ differ exactly by some element in $i(\mathbb{Z}_b)$ and i is injective. This definition also implies that f is *normalized*; i.e., $f(n, 0) = 0 = f(0, n)$ for all $n \in \mathbb{Z}_b$. In the spirit of base addition, we will sometimes refer to f as the *carry function* associated with s.

We may use any such s to: (i) construct a bijection ϕ_s between $\mathbb{Z}_b \times \mathbb{Z}_b$ and $\mathbb{Z}_{b^2}$; and (ii) compute the group law on $\mathbb{Z}_b \times \mathbb{Z}_b$ such that ϕ_s is an isomorphism. Notice that $s(\mathbb{Z}_b)$ is a set of coset representatives for $i(\mathbb{Z}_b)$ in $\mathbb{Z}_{b^2}$. Therefore, $\phi_s : \mathbb{Z}_b \times \mathbb{Z}_b \to \mathbb{Z}_{b^2}$ defined by $\phi_s(n_2, n_1) = i(n_2) + s(n_1)$ is a bijection. Next, for two elements $(n_2, n_1), (m_2, m_1) \in \mathbb{Z}_b \times \mathbb{Z}_b$, we may write

$$\begin{aligned}
\phi_s(n_2, n_1) + \phi_s(m_2, m_1) &= i(n_2) + s(n_1) + i(m_2) + s(m_1) \\
&= i(n_2) + i(m_2) + s(n_1) + s(m_1) \\
&= i(n_2) + i(m_2) + i(f(n_1, m_1)) + s(n_1 + m_1) \\
&= i(n_2 + m_2 + f(n_1, m_1)) + s(n_1 + m_1),
\end{aligned}$$

where in the second equality we have used that $\mathbb{Z}_{b^2}$ is abelian. Choosing the group law on $\mathbb{Z}_b \times \mathbb{Z}_b$

$$(n_2, n_1) + (m_2, m_1) = (n_2 + m_2 + f(n_1, m_1), n_1 + m_1), \tag{A.2}$$

then ϕ_s is an isomorphism. We henceforth use $\mathbb{Z}_b \times_f \mathbb{Z}_b$ to denote $\mathbb{Z}_b \times \mathbb{Z}_b$ with the group law (A.2) and carry function f.

For our inclusion and projection maps, define $\hat{\imath} : \mathbb{Z}_b \to \mathbb{Z}_b \times \mathbb{Z}_b$ by $\hat{\imath}(n) = (n, 0)$, and $\hat{\pi} : \mathbb{Z}_b \times \mathbb{Z}_b \to \mathbb{Z}_b$ by $\hat{\pi}(n_2, n_1) = n_1$. Notice $\hat{\imath}$ is just the canonical inclusion map of the second digit, and $\hat{\pi}$ the canonical projection map to the first digit. With $\hat{\imath}$, $\hat{\pi}$ as defined, Figure A.8 commutes for any choice of section s. For any $n \in \mathbb{Z}_b$, $(n_2, n_1) \in \mathbb{Z}_b \times_f \mathbb{Z}_b$,

$$\phi_s(\hat{\imath}(n)) = \phi_s((n, 0)) = i(n),$$
$$\pi(\phi_s((n_2, n_1))) = \pi(i(n_2) + s(n_1)) = n_1 = \pi(n_1).$$

Therefore, $\mathbb{Z}_b \times_f \mathbb{Z}_b$ is equivalent to $\mathbb{Z}_{b^2}$, and we have established how to properly represent numbers in $\{0, 1, ..., b^2 - 1\}$ as two digits, each in $\{0, 1, ..., b\}$.

Note that not any function $f : \mathbb{Z}_b \times \mathbb{Z}_b \to \mathbb{Z}_b$ defines a group $\mathbb{Z}_b \times_f \mathbb{Z}_b$. In fact, $\mathbb{Z}_b \times_f \mathbb{Z}_b$ is associative if and only if f satisfies

$$f(n, m) + f(n + m, p) = f(m, p) + f(n, m + p). \tag{A.3}$$

For, choosing arbitrary $(n_2, n_1), (m_2, m_1), (p_2, p_1) \in \mathbb{Z}_b \times_f \mathbb{Z}_b$, then

$$\begin{aligned}
&\big((n_2, n_1) + (m_2, m_1)\big) + (p_2, p_1) \\
&\qquad = (n_2 + m_2 + f(n_1, m_1), n_1 + m_1) + (p_2, p_1) \\
&\qquad = (n_2 + m_2 + p_2 + f(n_1, m_1) + f(n_1 + m_1, p_1), n_1 + m_1 + p_1), \\
&(n_2, n_1) + \big((m_2, m_1) + (p_2, p_1)\big) \\
&\qquad = (n_2, n_1) + (m_2 + p_2 + f(m_1, p_1), m_1 + p_1) \\
&\qquad = (n_2 + m + 2 + p_2 + f(m_1, p_1) + f(n_1, m_1 + p_1), n_1 + m_1 + p_1),
\end{aligned}$$

which are equal in the second digit exactly when f satisfies (A.3). We call (A.3) the *cocycle condition* because it implies that f is a 2-cocycle of $\mathbb{Z}_b$ with coefficients in $\mathbb{Z}_b$.

Appendix A.2.3. Comparing Equivalent Extensions

For any normalized section s, we have shown that $\mathbb{Z}_b \times_f \mathbb{Z}_b$ is equivalent to $\mathbb{Z}_{b^2}$ defined entirely by the associated cocycle f (as well as canonical inclusion and projection maps $\hat{\imath}$ and $\hat{\pi}$). Now, we will show that changing our choice of section s' corresponds 1-1 to modifying the cocycle f by a coboundary δc. Furthermore, we will show that two extensions $\mathbb{Z}_b \times_f \mathbb{Z}_b$ and $\mathbb{Z}_b \times_{f'} \mathbb{Z}_b$ are equivalent if and only if $f' = f + \delta c$ for some coboundary δc (see Figure A.9 for the commutative diagram).

Choose an arbitrary normalized section $s' : \mathbb{Z}_b \to \mathbb{Z}_{b^2}$. We may express s' in terms of our original s; i.e., $s'(n) = i(c(n)) + s(n)$, where $c : \mathbb{Z}_b \to \mathbb{Z}_b$ satisfies $c(0) = 0$. To compute the cocycle f' associated with s', we have

$$\begin{aligned}
s'(n) + s'(m) &= i(c(n)) + s(n) + i(c(m)) + s(m) \\
&= i(c(n)) + i(c(m)) + s(n) + s(m) \\
&= i(c(n) + c(m)) + i(f(n, m)) + s(n + m) \\
&= i(c(n) + c(m) + f(n, m)) + s(n + m) \\
&= i(c(n) + c(m) + f(n, m) - c(n + m)) \\
&\quad + i(c(n + m)) + s(n + m) \\
&= i(c(n) + c(m) + f(n, m) - c(n + m)) + s'(n + m),
\end{aligned}$$

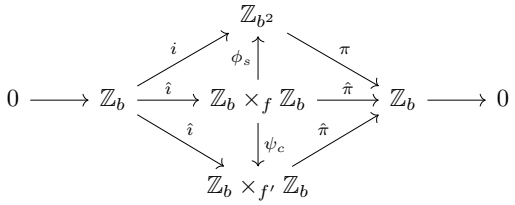

Figure A.9: We have already shown that $\mathbb{Z}_{b^2}$ and $\mathbb{Z}_b \times_f \mathbb{Z}_b$ are equivalent (see Appendix A.2.2 and Figure A.8). With $f' = f + \delta c$, $\mathbb{Z}_b \times_{f'} \mathbb{Z}_b$ is also equivalent (see Appendix A.2.3).

where the second equality holds because $\mathbb{Z}_{b^2}$ is abelian, and the third by definition of f. Then $\delta c : \mathbb{Z}_b \times \mathbb{Z}_b \to \mathbb{Z}_b$ defined by $\delta c(n, m) = c(n) + c(m) - c(n + m)$ is a 2-coboundary of $\mathbb{Z}_b$ with coefficients in $\mathbb{Z}_b$. So, the cocycle $f' : \mathbb{Z}_b \times \mathbb{Z}_b \to \mathbb{Z}_b$ associated with s' is given by $f' = f + \delta c$.

It remains to show that $\mathbb{Z}_b \times_{f'} \mathbb{Z}_b$ is equivalent to $\mathbb{Z}_b \times_f \mathbb{Z}_b$. Let $\psi_c : \mathbb{Z}_b \times_f \mathbb{Z}_b \to \mathbb{Z}_b \times_{f'} \mathbb{Z}_b$ be defined by $\psi_c(n_2, n_1) = (n_2 + c(n_1), n_1)$. We will show that ψ_c is an isomorphism. First, $\psi_c(0, 0) = (0 + c(0), 0) = (0, 0)$, so ψ_c preserves the identity element. Moreover, we have

$$\begin{aligned}
\psi_c((n_2, n_1) + (m_2, m_1)) &= \psi_c((n_2 + m_2 + f(n_1, m_1), n_1 + m_1)) \\
&= (n_2 + m_2 + f(n_1, m_1) + c(n_1 + m_1), n_1 + m_1) \\
&= (n_2 + m_2 + c(n_2) + c(m_1) + f'(n_1, m_1), n_1 + m_1) \\
&= (n_2 + c(n_1), n_1) + (m_2 + c(m_1), m_1) \\
&= \psi_c(n_2, n_1) + \psi_c(m_2, m_1),
\end{aligned}$$

where the third equality holds because $f' = f + \delta c$. So, ψ_c also preserves addition. Furthermore, Figure A.9 commutes properly. For any $n \in \mathbb{Z}_b$, $(n_2, n_1) \in \mathbb{Z}_b \times_f \mathbb{Z}_b$,

$$\begin{aligned}
\psi_c(\hat{\imath}(n)) = \psi_c((n, 0)) = (n, 0) = \hat{\imath}(n), \\
\hat{\pi}(\psi_c((n_2, n_1))) = \hat{\pi}((n_2 + c(n_1), n_1)) = n_1 = \hat{\pi}((n_2, n_1)).
\end{aligned}$$

Therefore, $\mathbb{Z}_b \times_f \mathbb{Z}_b$ and $\mathbb{Z}_b \times_{f'} \mathbb{Z}_b$ are equivalent.

Appendix A.2.4. Iteratively Extending

Thus far, we've shown how to represent 2-digit numbers with an addition (namely, a way to carry from the first to second digit) that preserves the structure of $\mathbb{Z}_{b^2}$. For a complete base representation of the integers, we wish to present any number in $\mathbb{Z}$ as an arbitrary-length sequence of digits, and when adding, for the carry function between digits to be somehow consistent or recursive. With this aim, we proceed by iteratively extending our base representation and, each time, choosing the same cocycle (up to embedding). We will see that – if the cocycle meets certain constraints – this procedure produces a base representation with the exact form we want. We first extend to three digits and then inductively extend for an arbitrary number of digits.

For the 3-digit case, we extend $\mathbb{Z}_{b^2}$ by $\mathbb{Z}_b$ to get $\mathbb{Z}_{b^3}$, similarly to $\mathbb{Z}_{b^2}$ in Appendix A.2.1. Denote the inclusion and projection maps by $i_2 : \mathbb{Z}_{b^2} \to \mathbb{Z}_{b^3}$ and $\pi_2 : \mathbb{Z}_{b^3} \to \mathbb{Z}_b$, respectively. As before, we seek an equivalent extension defined on the product $(\mathbb{Z}_b)^3 = \mathbb{Z}_b \times \mathbb{Z}_b \times \mathbb{Z}_b$ so it is a proper 3-digit base representation preserving the structure of $\mathbb{Z}_{b^3}$. Having already shown that the diagram in Figure A.8 commutes, we now extend the diagram in Figure A.10 below. The remainder of this section is devoted to defining a group law on $(\mathbb{Z}_b)^3$ (we denote the resulting group by $(\mathbb{Z}_b)^3_f$), inclusion and projection maps $\hat{\imath}_2$ and $\hat{\pi}_2$, and an isomorphism $\phi_{s_2} : (\mathbb{Z}_b)^3_f \to \mathbb{Z}_{b^3}$.

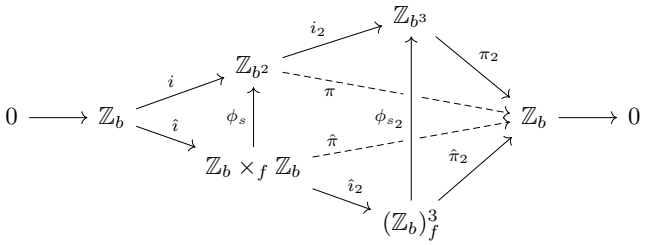

Figure A.10: The base b representation $(\mathbb{Z}_b)^3_f$ is equivalent to $\mathbb{Z}_{b^3}$ with proper choice of 2-equivariant cocycle f (see Appendix A.2.4). With π and $\hat{\pi}$ included as dashed lines, notice that the diagram from Figure A.8 is embedded as a sub-diagram with the same mappings as before.

We construct our equivalent representation $(\mathbb{Z}_b)^3_f$ as follows (and recall our section $s : \mathbb{Z}_b \to \mathbb{Z}_{b^2}$, cocycle $f : \mathbb{Z}_b \times \mathbb{Z}_b \to \mathbb{Z}_b$, and $\phi_s : \mathbb{Z}_b \times_f \mathbb{Z}_b \to \mathbb{Z}_{b^2}$ from Appendix A.2.2): first, we define section $s_2 : \mathbb{Z}_b \to \mathbb{Z}_b^3$, cocycle $f_2 : \mathbb{Z}_b \times \mathbb{Z}_b \to \mathbb{Z}_b \times_f \mathbb{Z}_b$, and inclusion and projection maps $\hat{\imath}_2$, $\hat{\pi}_2$ similarly to the 2-digit case; second, we limit our attention to cocycles f_2 of the form $(0, f)$; and, third, we rewrite our representation $(\mathbb{Z}_b \times_f \mathbb{Z}_b) \times_{f_2} \mathbb{Z}_b$ as $(\mathbb{Z}_b)^3_f$ with an appropriate group law on the set $(\mathbb{Z}_b)^3$. Choose a normalized section $s_2 : \mathbb{Z}_b \to \mathbb{Z}_{b^3}$ such that $\pi_2 s_2 = \mathrm{id}_{\mathbb{Z}_b}$. Similarly to before, we define $f_2 : \mathbb{Z}_b \times \mathbb{Z}_b \to \mathbb{Z}_b \times_f \mathbb{Z}_b$ by

$$s_2(n) + s_2(m) = i_2(\phi_s(f_2(n, m))) + s_2(n + m).$$

f_2 is normalized and well-defined by definition of s_2, i_2, and ϕ_s (see Appendix A.2.2 for details). Furthermore, if we define the map $\phi_{s_2} : (\mathbb{Z}_b \times_f \mathbb{Z}_b) \times_{f_2} \mathbb{Z}_b \to \mathbb{Z}_{b^3}$ as $\phi_{s_2}((n_3, n_2), n_1) = i_2(\phi_s(n_3, n_2)) + s_2(n_1)$, we have that ϕ_{s_2} is an isomorphism if $(\mathbb{Z}_b \times_f \mathbb{Z}_b) \times_{f_2} \mathbb{Z}_b$ has group law

$$((n_3, n_2), n_1) + ((m_3, m_2), m_1) = ((n_3, n_2) + (m_3, m_2) + f_2(n_1, m_1), n_1 + m_1), \quad \text{(A.4)}$$

which is very similar to (A.2). For our inclusion and projection maps, define $\hat{\imath}_2 : \mathbb{Z}_b \times_f \mathbb{Z}_b \to (\mathbb{Z}_b \times_f \mathbb{Z}_b) \times_{f_2} \mathbb{Z}_b$ by $\hat{\imath}_2(n_2, n_1) = ((n_2, n_1), 0)$, and $\hat{\pi}_2 : (\mathbb{Z}_b \times_f \mathbb{Z}_b) \times_{f_2} \mathbb{Z}_b \to \mathbb{Z}_b$ by $\hat{\pi}_2((n_3, n_2), n_1) = n_1$. As with $\hat{\imath}$ and $\hat{\pi}$, notice $\hat{\imath}_2$ is just the canonical inclusion map of the second and third digits, and the $\hat{\pi}_2$ the canonical projection map to the first digit, and it is easy to show that Figure A.10 commutes for any choice of section s_2. Thus, $(\mathbb{Z}_b \times_f \mathbb{Z}_b) \times_{f_2} \mathbb{Z}_b$ is equivalent to $\mathbb{Z}_{b^3}$.

For the purposes of a base representation, we would like to somehow apply f iteratively rather than having two distinct carries f and f_2. So, suppose f_2 is of the form

$f_2(n,m) = (0, f(n,m))$. If $f_2 = (0,f)$ is a cocycle (which is not necessarily the case for arbitrary cocycle f), then we call f a *2-equivariant cocycle*. For any 2-equivariant cocycle f, our group law (A.4) may be re-written as

$$\begin{aligned}
&\big((n_3,n_2),n_1\big) + \big((m_3,m_2),m_1\big) \\
&\quad = \big((n_3,n_2) + (m_3,m_2) + (0,f(n_1,m_1)), n_1+m_1\big) \\
&\quad = \big((n_3+m_3+f(n_2,m_2), n_2+m_2) + (0,f(n_1,m_1)), n_1+m_1\big) \\
&\quad = \big(n_3+m_3+f(n_2,m_2)+f(n_2+m_2,f(n_1,m_1)), n_2+m_2+f(n_1,m_1)), n_1+m_1\big),
\end{aligned}$$

and we have essentially just defined an addition on the set $(\mathbb{Z}_b)^3$ making it isomorphic to $\mathbb{Z}_{b^3}$. To be explicit, let our group rule on $(\mathbb{Z}_b)^3$ be

$$\begin{aligned}
&(n_3,n_2,n_1) + (m_3,m_2,m_1) \\
&\quad = \big(n_3+m_3+f(n_2,m_2)+f(n_2+m_2,f(n_1,m_1)), n_2+m_2+f(n_1,m_1)), n_1+m_1\big),
\end{aligned}$$

and denote the resulting group by $(\mathbb{Z}_b)^3_f$ for 2-equivariant cocycle f.

Redefining $\phi_{s_2} : (\mathbb{Z}_b)^3_f \to \mathbb{Z}_{b^3}$, $\hat{\imath}_2 : \mathbb{Z}_b \times_f \mathbb{Z}_b \to (\mathbb{Z}_b)^3_f$, and $\hat{\pi}_2 : (\mathbb{Z}_b)^3_f \to \mathbb{Z}_b$ as

$$\begin{aligned}
\phi_{s_2}(n_3,n_2,n_1) = i_2(\phi_s(n_3,n_2)) + s_2(n_1) = i_2(i(n_3)) + i_2(s(n_2)) + s_2(n_1), \\
\hat{\imath}_2(n_2,n_1) = (n_2,n_1,0), \\
\hat{\pi}_2(n_3,n_2,n_1) = n_1,
\end{aligned}$$

then $(\mathbb{Z}_b)^3_f$ is equivalent to $\mathbb{Z}_{b^3}$ defined entirely in terms of 2-equivariant cocycle f.

Finally, we define the k-digit base representation $(\mathbb{Z}_b)^k_f$ for any $k \in \mathbb{N}$, which allows us to represent arbitrary $n \in \mathbb{Z}$ as a sequence of digits $(n_k, n_{k-1}, ..., n_1)$. First, let $(\mathbb{Z}_b)^k_f$ be the product $(\mathbb{Z}_b)^k$ with addition defined by

$$n_j + m_j + f(n_{j-1}, m_{j-1}) + f(n_{j-1} + m_{j-1}, c_{j-1}) \tag{A.5}$$

for digits $j \in [k]$, where $c_j = f(n_{j-1},m_{j-1}) + f(n_{j-1}+m_{j-1},c_{j-1})$ for $j \in [k]$, and $c_0 = n_0 = m_0 = 0$, and $f : \mathbb{Z}_b \times \mathbb{Z}_b \to \mathbb{Z}_b$ is some function. If f is a *k-equivariant cocycle*, i.e., a cocycle of $\mathbb{Z}_b$ with coefficients in $\mathbb{Z}_b$ such that $(\mathbb{Z}_b)^{k+1}_f$ forms a group with the addition (A.5) – or, in group cohomological terms, such that $f_k = (0,...,0,f)$ is a cocycle of $\mathbb{Z}_b$ with coefficients in $(\mathbb{Z}_b)^k_f$ – then $(\mathbb{Z}_b)^k_f$ forms a group. Notice this reduces exactly to $(\mathbb{Z}_b)^2_f = \mathbb{Z}_b \times_f \mathbb{Z}_b$ and $(\mathbb{Z}_b)^3_f$ for $k = 2$ and 3, respectively. It is easy to show by induction, with a similar diagram to Figure A.10, that if $(\mathbb{Z}_b)^k_f$ is equivalent to $\mathbb{Z}_{b^k}$ and f is a k-equivariant cocycle, then $(\mathbb{Z}_b)^{k+1}_f$ is equivalent to $\mathbb{Z}_{b^{k+1}}$. Simply define $\phi_{s_k} : (\mathbb{Z}_b)^{k+1}_f \to \mathbb{Z}_{b^{k+1}}$, $\hat{\imath}_k : (\mathbb{Z}_b)^k_f \to (\mathbb{Z}_b)^{k+1}_f$, and $\hat{\pi}_k : (\mathbb{Z}_b)^{k+1}_f \to \mathbb{Z}_b$ by

$$\begin{aligned}
\phi_{s_k} = i_k\big(\phi_{s_{k-1}}(n_{k+1},...,n_2)\big) + s_k(n_1), \\
\hat{\imath}_k(n_k,...,n_1) = (n_k,...,n_1,0), \\
\hat{\pi}_k(n_{k+1},...,n_1) = n_1,
\end{aligned}$$

where $s_k : \mathbb{Z}_b \to \mathbb{Z}_{b^{k+1}}$ is a section corresponding to cocycle $f_k = (0,...,0,f)$ and $i_k : \mathbb{Z}_{b^k} \to \mathbb{Z}_{b^{k+1}}$ is the usual inclusion map, and the diagram will commute as desired. Therefore, we are able to faithfully represent any $n \in \mathbb{Z}$ by a sequence of digits $(n_k, n_{k-1}, ..., n_1)$, and so we have a complete base representation of the integers.

Note that, $(\mathbb{Z}_b)^k_f$ and $(\mathbb{Z}_b)^k_{f'}$ are equivalent if and only if $\mathbb{Z}_b \times_f \mathbb{Z}_b$ and $\mathbb{Z}_b \times_{f'} \mathbb{Z}_b$ are equivalent and f, f' are k-equivariant. So, for a given base representation $(\mathbb{Z}_b)^k_f$ and associated f, it suffices to check that f' is k-equivariant and $f' = f + \delta c$ in order to show that $(\mathbb{Z}_b)^k_{f'}$ is equivalent to $(\mathbb{Z}_b)^k_f$.

Appendix B. Supplement to Section 3: Quantitative Measures of Carry Functions

Appendix B.1. Additional Carry Tables

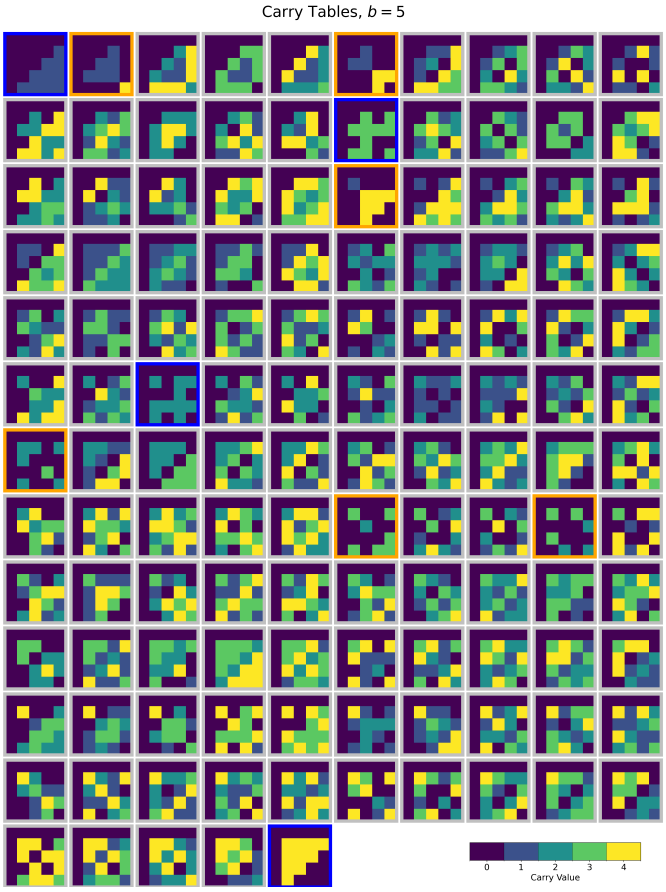

Figure B.11: The 125 carry tables in base 5. The Single Value carry functions are outlined in blue (including the **1** carry function in the top left), the Low Dimensional Multiple Value carry functions in orange, and the Other Multiple Value carry functions in grey (see Section 2.4). Each table's entries are indexed by $\{0, 1, ..., eb-1\}$ from left to right, top to bottom; color indicates the value that is carried (see legend at bottom right).

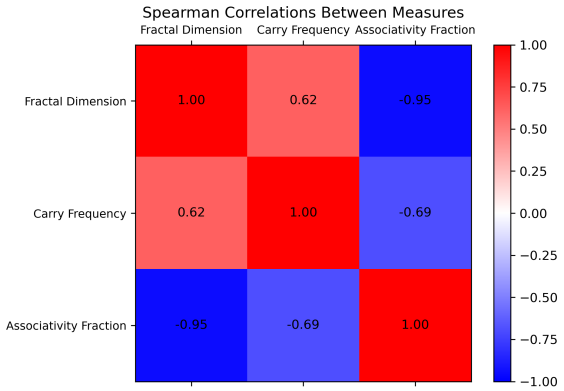

Figure B.12: Spearman correlations between the three quantitative measures: fractal dimension, carry frequency, and associativity fraction. This demonstrates that the measures are not identical, and reflect different aspects of a carry function's structure.

Appendix C. Supplement to Section 4: Neural Network Simulations

Appendix C.1. LSTM Results

Here we present the same results as in Section 4, but using a single-layer LSTM in place of a GRU. Otherwise, the training procedure was identical. Figures C.13 to C.16 show the training results for symbolic (one-hot) embeddings and semantic embeddings, the generalization results, and scatter plots of maximum testing accuracy (on 6-digit numbers) and the three quantitative measures.

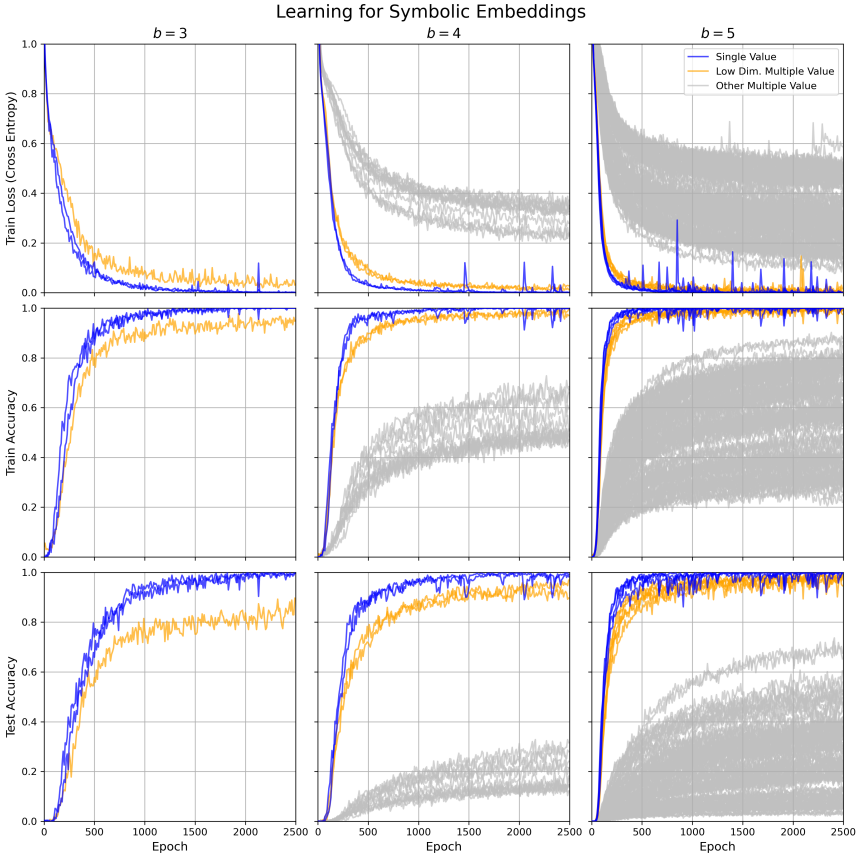

Figure C.13: Learning for symbolic (one-hot) embeddings of digits using an LSTM. Performance over the course of training (averaged over 10 runs of each model implementation) for addition on different carry functions for bases $b = 3$ to 5.

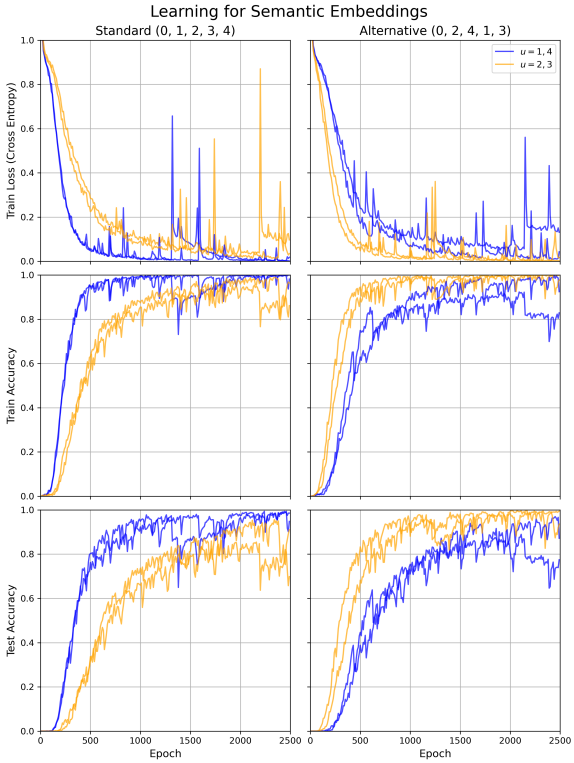

Figure C.14: Learning for semantic embeddings of digits using an LSTM. Performance over the course of training on Single Value carry functions with order-encoded inputs in the two non-degenerate orderings (0, 1, 2, 3, 4) and (0, 2, 4, 1, 3) for $b = 5$ (see Section 2.4.1 for how Single Value carry functions correspond to orderings of $\mathbb{Z}_b$).

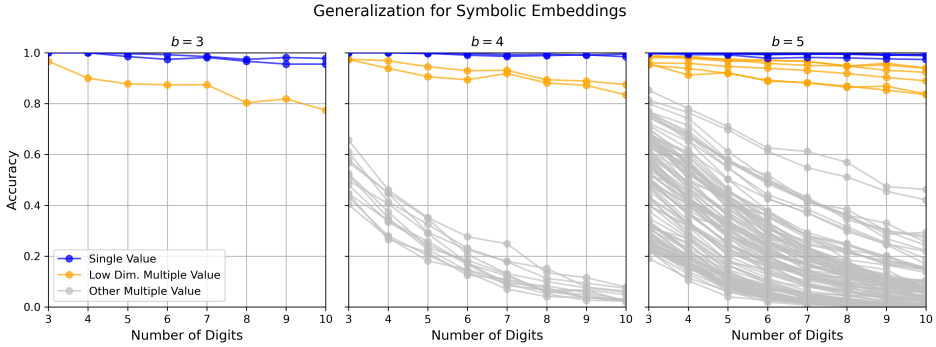

Figure C.15: Out-of-domain generalization for symbolic embeddings of digits using an LSTM. For each carry function in bases $b = 3$ to 5, accuracy was tested on k-digit numbers for each $k \in [3 : 10]$ after training on 3-digit numbers (averaged over 10 training/testing runs).

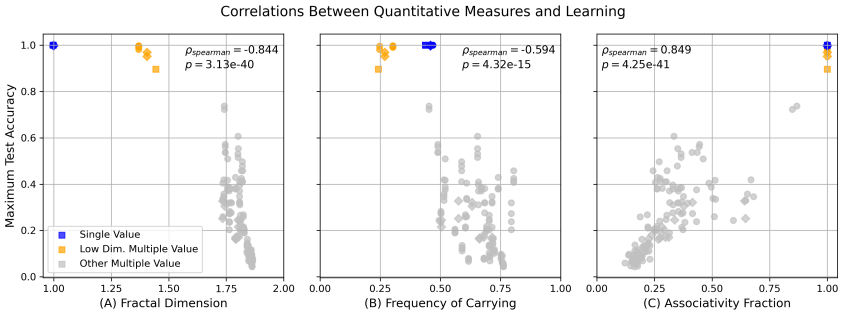

Figure C.16: Relationship of carry function structure and learning using an LSTM, as measured by maximum testing accuracy (on 6-digit numbers). Scatter plots showing relationship of learning curves to structure measured as (A) fractal dimension, (B) frequency of carrying, and (C) associativity fraction for different carry functions, divided into three categories: Single Value carry functions (blue), Low Dimensional Multiple Value carry functions (orange), and other Multiple Value carry functions (grey). The base b of each carry function is indicated by shape ($b = 3$: squares; $b = 4$: diamonds; and $b = 5$: circles). Spearman's rank correlations (over bases $b = 3$ to 5) and significance are shown for each plot.

Appendix C.2. Sigmoid Fit

For robustness, we also quantified learning by fitting a sigmoid function $\sigma(x) = \frac{a}{1+e^{-b(x-c)}}$, with parameters of upper asymptote a, growth rate b, and critical point c, to each test accuracy curve (on 6-digit numbers). To avoid overfitting to fluctuations in the tails of the test accuracy curves, we restricted our fit to the portion of the curve up until it reached its maximum observed value. Curve fitting was performed using non-linear least squares via `scipy.optimize.curve_fit`. For both GRUs and LSTMs, the sigmoid provided a good fit to test accuracy curves ($R^2 = 0.84 \pm 0.09$ and $R^2 = 0.92 \pm 0.04$, respectively). Figure C.17 shows well-fit test accuracy curves for a GRU (top) and LSTM (bottom).

From the best-fit sigmoid, we took two measurements of learning: its upper asymptote and critical point. For the critical points specifically, we divisively normalized by the minimal critical point for each base (corresponding to the fastest-learned carry function), in order to examine the structure of carry functions independent of base. Figures C.18 and C.19 show scatter plots of these measurements of learning speed and the three quantitative measures using a GRU and LSTM, respectively.

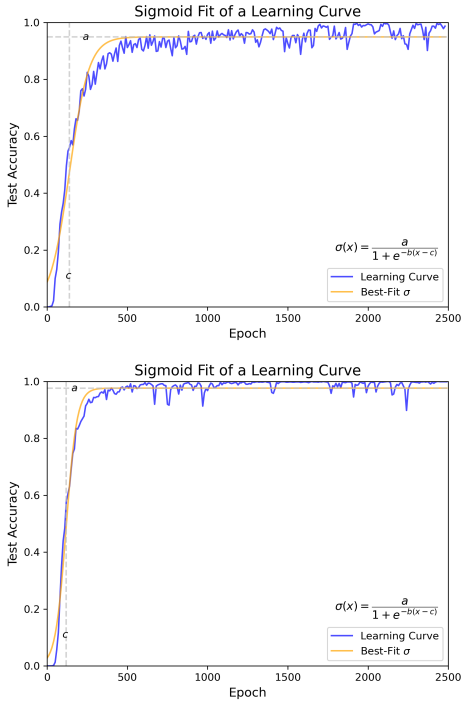

Figure C.17: Sigmoid fit to learning curve of a GRU (top) and LSTM (bottom). A well-fit example of the sigmoid function σ fit to a test accuracy curve (top: $R^2 = 0.96$; bottom: $R^2 = 0.98$), and the corresponding upper asymptote (a) and critical point (c).

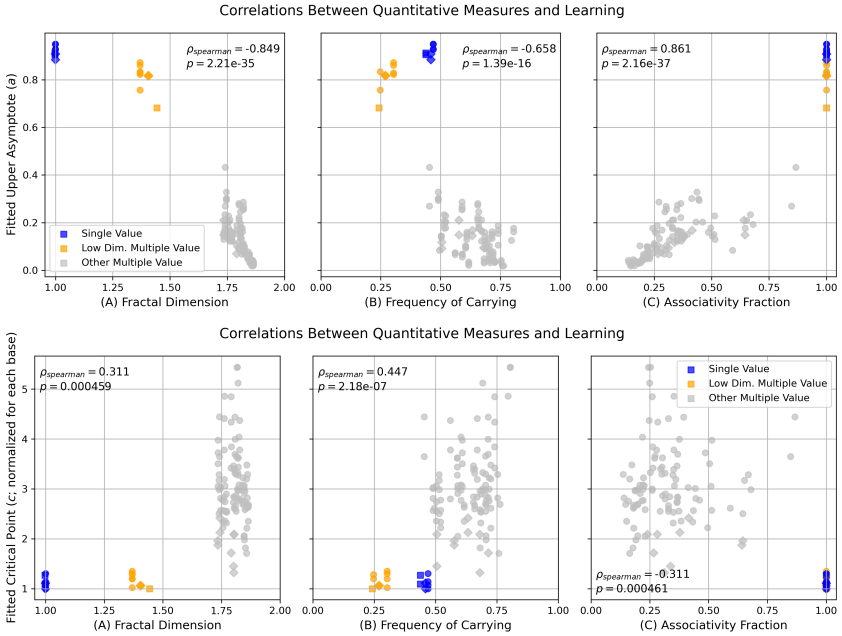

Figure C.18: Relationship of carry function structure and learning using a GRU, as measured by the best-fit sigmoid's upper asymptote (top) and critical point (bottom). Scatter plots showing relationship of learning curves to structure measured as (A) fractal dimension, (B) frequency of carrying, and (C) associativity fraction for different carry functions, divided into three categories: Single Value carry functions (blue), Low Dimensional Multiple Value carry functions (orange), and other Multiple Value carry functions (grey). The base b of each carry function is indicated by shape ($b = 3$: squares; $b = 4$: diamonds; and $b = 5$: circles). Spearman's rank correlations (over bases $b = 3$ to 5) and significance are shown for each plot. Note that `scipy.optimize.curve_fit` was unable to learn some learning curves, which are excluded from the figure ($\sim 14.6\%$).

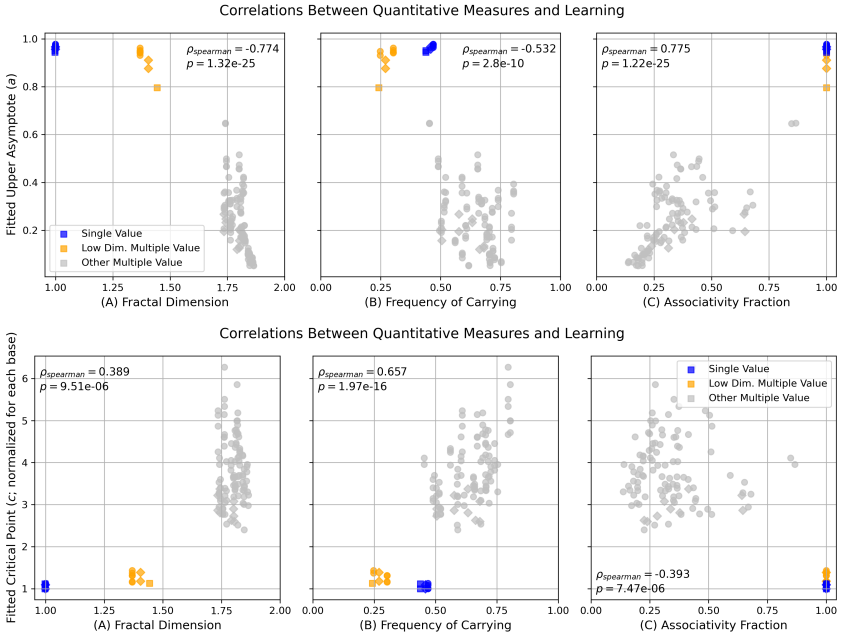

Figure C.19: Relationship of carry function structure and learning using an LSTM, as measured by the best-fit sigmoid's upper asymptote (top) and critical point (bottom). Scatter plots showing relationship of learning curves to structure measured as (A) fractal dimension, (B) frequency of carrying, and (C) associativity fraction for different carry functions, divided into three categories: Single Value carry functions (blue), Low Dimensional Multiple Value carry functions (orange), and other Multiple Value carry functions (grey). The base b of each carry function is indicated by shape ($b = 3$: squares; $b = 4$: diamonds; and $b = 5$: circles). Spearman's rank correlations (over bases $b = 3$ to 5) and significance are shown for each plot. Note that `scipy.optimize.curve_fit` was unable to learn some learning curves, which are excluded from the figure ($\sim 15.3\%$).