
LARS-VSA: A Vector Symbolic Architecture For Learning with Abstract Rules

Mohamed Mejri

School of ECE

Georgia Institute of Technology

Atlanta, USA

mmejri3@gatech.edu

Chandramouli Amarnath

School of ECE

Georgia Institute of Technology

Atlanta, USA

chandamarnath@gatech.edu

Abhijit Chatterjee

School of ECE

Georgia Institute of Technology

Atlanta, USA

abhijit.chatterjee@ece.gatech.edu

Abstract

Human cognition excels at symbolic reasoning, deducing abstract rules from limited samples. This has been explained using symbolic and connectionist approaches, inspiring the development of a neuro-symbolic architecture that combines both paradigms. In parallel, recent studies have proposed the use of a "relational bottleneck" that separates object-level features from abstract rules, allowing learning from limited amounts of data. While powerful, it is vulnerable to the *curse of compositionality* meaning that object representations with similar features tend to interfere with each other. In this paper, we leverage hyperdimensional computing, which is inherently robust to such interference to build a compositional architecture. We adapt the "relational bottleneck" strategy to a high-dimensional space, incorporating explicit vector binding operations between symbols and relational representations. Additionally, we design a novel high-dimensional attention mechanism that leverages this relational representation. Our system benefits from the low overhead of operations in hyperdimensional space, making it significantly more efficient than the state of the art when evaluated on a variety of test datasets, while maintaining higher or equal accuracy.

1 Introduction

Analogical reasoning based on relationships between objects is fundamental to human abstraction and creative thinking. This capability differs from our ability to acquire semantic and procedural knowledge from sensory information, processed using contemporary connectionist approaches such as deep neural networks (DNNs). However, most of these techniques fail to extract relational abstract rules from limited samples [9]. Recent advancements in machine learning have enhanced abstract reasoning capabilities. Broadly, these rely on isolating abstract relational rules from the corresponding representations of objects, such as symbols/(key, query) [1] or (key, values) [27]. This approach, called the *relational bottleneck*, is exploited in [1], using attention mechanisms [25] to capture relevant correlations between objects in a sequence, producing relational representations. The latter structure is combined with a learnable inductive bias given by *symbols*, linked to the relational representation through a *relational cross-attention* [1] mechanism. More generally, the *relational bottleneck* [28], aims to reduce *catastrophic interference* [4, 24] between object-level and abstract-level features. This

catastrophic interference, also known as the *curse of compositionality* is due to the overuse of shared structures and the low-dimensionality of feature representations [28]. In[5], it is argued that over compositionality leads to *inefficient generalization* (i.e, requiring expensive processing capabilities).

The above problem is partially addressed by neuro-symbolic approaches that use quasi-orthogonal high-dimensional vectors [9, 16] for storing relational representations. In [16], it is shown that high dimensional vector are less prone to interference. However, recent neuro-symbolic approaches for abstract reasoning [9] rely on explicit binding and unbinding mechanisms, requiring prior knowledge of abstract rules. This paper presents LARS-VSA (for *Learning with Abstract RuleS*), an approach that combines the advantages of connectionist approaches in capturing implicit abstract rules and the ability of neuro-symbolic architectures to absorb relevant features with low risk of interference.

The key contributions of this paper are:

- We are the *first* to propose a strategy for addressing the relational bottleneck problem using a vector symbolic architecture. This performs explicit bindings in high-dimensional space, capturing relationships between symbolic representations of objects distinct from object level features.
- We implement a context-based self-attention mechanism that operates directly in a bipolar high-dimensional space. Vectors that represent relationships between symbols are developed, decoupling our high-dimensional vector symbolic approach from the need for prior knowledge of abstract rules (seen in prior work [9]).
- Our system significantly reduces computational costs by simplifying attention score matrix multiplication to binary operations, offering a lightweight alternative to conventional attention mechanisms.

In the following sections, we introduce the intuition behind and explain our novel *self-attention* mechanism for hyperdimensional computing. We then discuss the proposed vector symbolic architecture for addressing the relational bottleneck problem in high-dimensional spaces. We compare the performance of LARS-VSA with the Abstractor [1], a standard transformer architecture [25], and other state-of-the-art methods on discriminative relational tasks to demonstrate the accuracy and cost efficiency of our approach. We evaluate LARS-VSA on various synthetic sequence-to-sequence datasets and complex mathematical problem-solving tasks [22] to illustrate its potential in real-world applications. Finally, we show the resilience of LARS-VSA to weight heavy quantization [1].

2 Related Work

Several studies [3, 14, 20] have demonstrated that abstract reasoning and relational representations are notable weaknesses in modern neural networks. Large language models can tackle symbolic reasoning tasks to a certain extent but require extensive training data, which highlights their inability to generalize from limited examples. Consequently, significant effort has recently been directed towards addressing this deficiency. The Relation Network [21] proposed a model for pairwise relations by applying a Multilayer Perceptron (MLP) to concatenated object features. Another architecture, PrediNet [23], utilizes predicate logic to represent these relations. In [28], the relational bottleneck implementation aims to separate symbolic rules from object level features. Several models are based on the latter idea: *CoRelNet*, introduced in [12] simplifies relational learning by modeling a similarity matrix. A recent study [27] introduced an architecture inspired by Neural Turing Machines (NTM) [7] that separates relational representations from object feature representations. Building on this concept, the approach of [1] adapted Transformers [25] for abstract reasoning tasks, by creating an 'abstractor'—a mechanism based on relational and symbolic cross-attention applied to abstract symbols for sequence-to-sequence relational tasks. Additionally, a model known as the Visual Object Centering Relational Abstract architecture (OCRA) [26] maps visual objects to vector embeddings, followed by a transformer network for solving symbolic reasoning problems such as the Raven Progressive Matrices [19]. A subsequent study [17] combined and refined OCRA and the Abstractor to address similar challenges.

However, research [9] has shown that hyperdimensional computing (HDC), a neuro-symbolic paradigm [10], exhibits strong abstraction capabilities through the *blessing of dimensionality* mechanism [16]. Hyperdimensional Computing (HDC) is recognized for its low computational overhead [2, 15]. In contrast, Transformers [25] are known to be power-intensive and time-consuming. An

explicit binding and unbinding based VSA architecture [9] was developed to solve a visual abstract reasoning problem: Raven Progressive Matrices [19]. This *explicitly* represents Raven’s Progressive Matrix features (object shapes, color, etc) in the symbolic domain. However, this method requires prior knowledge of relevant object features and the abstract rules governing their relationships and is not always straightforward (e.g, mathematical reasoning tasks). In relation to prior research and to the best of our knowledge, this paper is the first to propose a hyperdimensional computing based attention mechanism called *LARS-VSA*, that models relational representations, and combines them through an explicit vector-binding operation with high dimensional symbolic vectors to perform sequence-to-sequence relational representation tasks.

3 Related Work: Self Attention and Relational Cross Attention

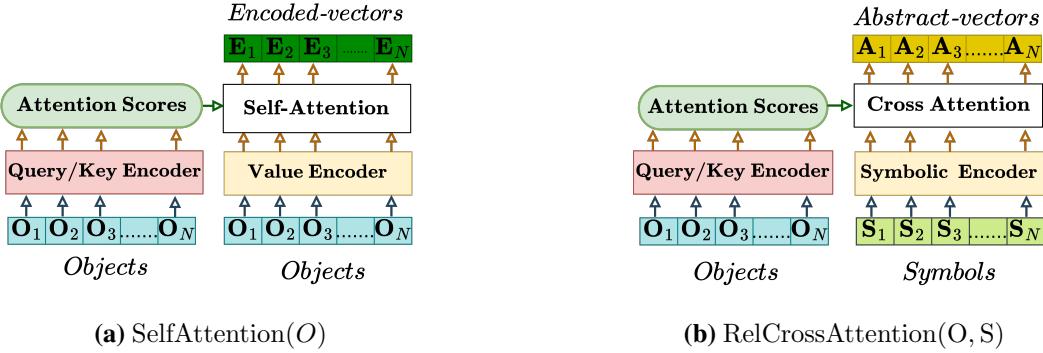


Figure 1: Relational cross-attention and self-attention. We show a single head of multi-attention. In blue, the *objects* related by $r(., .)$; in green, the inductive bias denoted by *symbols*

Figure 1 shows a single head for different attention mechanisms from prior work: Figure 1a refers to the regular *transformer self-attention mechanism* applied to a sequence of objects $O_{1..N}$ as in [25]. The Query/Key Encoder in Figure. 1a uses learnable projections $\phi_Q : O \mapsto O \cdot W_Q$ and $\phi_K : O \mapsto O \cdot W_K$. A self attention score matrix is computed from these projections. The latter matrix is used as weights for the value vectors derived from the objects O using a learnable projection function $\phi_V : O \mapsto O \cdot W_V$ to generate encoded vectors (E_1, \dots, E_N) . Equation (3) summarizes the *multi-head self-attention* mechanism applied to a sequence of objects $O_{1..N} = [O_1, \dots, O_N]$ across H heads. Here, W_o constitutes a weighted summation of the outputs of H different self-attention heads. $SelfAttention(O) = \text{concat}(\hat{E}^1, \dots, \hat{E}^H) \cdot W_o$, where $\hat{E}^i = [E_1^i, E_2^i, \dots, E_N^i] = \text{Softmax}((O \cdot W_q^i)(O \cdot W_k^i)^\top)(O \cdot W_v^i), 1 \leq i \leq H$ (3)

The work of [27] was the first to propose the use of segregated datapaths for processing keys and values with the goal of processing relational information distinctly from object level features. This led to the concept of the Abstractor [1] in which the self-attention mechanism of a transformer was adapted to allow a relational attention mechanism to be implemented and integrated with symbolic processing. Figure 1b shows a RelCrossAttention mechanism [1] (for a single head) that replaces object-level features (*Values* V) with abstract symbols denoted as $S \in \mathbb{R}^{N \times F}$. The symbols are projected into value vectors through a projection function: $\phi_O : S \mapsto S \cdot W_O$. We have $Q \leftarrow O$, $K \leftarrow O$, $V \leftarrow S$. In this case, σ_{rel} refers to the Softmax function. The formalism for multi head attention is $RelCrossAttention(O, S) = \text{concat}(A^1, \dots, A^H)W_o$, where $A^i = \sigma_{\text{rel}}((O \cdot W_q^i)(O \cdot W_k^i)^\top)S \cdot W_O^i$.

4 LARS-VSA: Attention Mechanism With Hypervector Bundling

Our hyperdimensional symbolic attention mechanism aims to extract abstract relationships $r(., .) \in \mathbb{R}^H$ between objects O_1, \dots, O_n . These abstract relations are modeled by a pairwise relationship (i.e *inner product*) between encoded objects (using learnable query and key encodings) $\{r(O_i, O_j)\}_{i,j \in [0, N]}$ [1]. However, the query and key embedding in prior work are represented in a low dimensional space (i.e, dimension less than hundred). In this research, the objects are projected

onto a *high D-dimensional hyperspace* (i.e $D \geq 1000$) with data represented as *hypervectors*. In [16], it is shown that as the dimensionality of hypervectors increases, they become *orthogonal*, making a relational modeling approach such as [1] difficult to apply in the hyperdimensional space. Applying inner product to orthogonal vectors leads to very low attention scores that do not reflect the actual object correlations and relationships. One way to address this issue is to model the relationship between objects *indirectly* by measuring their correlation to a *local context*. To illustrate this, consider two objects, A and B . Instead of examining the explicit correlation between objects A and B , we assess the correlation/relationship between the object A and a local context formed by the sequence " AB ".

The representation of a sequence of objects in a hyperdimensional space is well defined using the *bundling* operation (i.e., \oplus) between sequence elements. Given two objects O_i and O_j , we project them onto a hyperspace using a set of N learnable projection functions: $\{\phi_{B_i}\}_{i=0}^{i=N} : \mathbb{R}^F \rightarrow \mathbb{R}^D$, for a system with N objects. Hence the relation $r(O_i, O_j)$ can be expressed as in Equation 1.

$$r(O_i, O_j) = (f(\phi_{B_i}(O_i), \phi_{B_j}(O_j))) \in \mathbb{R} \quad (1)$$

The function $f : \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}$ implements the *indirect* correlation we discussed earlier. for $h_{O_1}, h_{O_2} \in \mathbb{R}^D$ we have

$$f(h_{O_1}, h_{O_2}) = \cos(h_{O_1}, h_{O_1} \oplus h_{O_2}) = \frac{\langle h_{O_1}, h_{O_1} \oplus h_{O_2} \rangle}{D} \quad (2)$$

here we define *bundling* (i.e., \oplus) as $h_1 \oplus h_2 = \text{sign}(h_1 + h_2)$. $\text{sign}(x) = -\mathbb{1}_{\{x < 0\}} + \mathbb{1}_{\{x > 0\}}$ replaces the binary coordinate-wise majority in the bipolar domain[11]. Considering all pairs of objects, we construct a *relational matrix* consisting of $R = [r(O_i, O_j)] \in \mathbb{R}^{N \times N}$. The bundling operation implicitly acts as a majority vote between two object elements. It captures the *dominant/relevant* features of an object sequence rather than just common features. Indeed, the sign of each hypervector element follows the sign of the element with higher magnitude which is amplified by dominant features of object sequences during training. Figure 3 illustrates the pipeline for HDSymbolicAttention from end to end applied to objects $O_{1..N}$. The objects were first mapped to hypervectors $h_{O_{1..N}}$ through learnable projection function $\phi_H : O \mapsto O \cdot W_H$. We extract the attention scores $r_{ij} = f(h_{O_i}, h_{O_j}) = \cos(h_{O_i}, h_{O_i} \oplus h_{O_j})$ from the hypervectors h_{O_i} and h_{O_j} . This means that the hypervector h_{O_i} attention score is computed with respect to its context illustrated in $h_{O_i} \oplus h_{O_j}$. These attention scores are then forwarded to a softmax function and we used to compute a weighted sum of h_{O_i} . We should notice that each hypervector h_{O_i} mainly contributes to its corresponding output hypervector since $r_{ii} \leq r_{ij} \forall i \neq j$. We should also notice that although the cosine similarity is theoretically between -1 and +1, since we are working in hyperdimensional space, the cosine similarity is less likely to be negative[13].

5 LARS-VSA: Symbolic Reasoning with Hypervectors

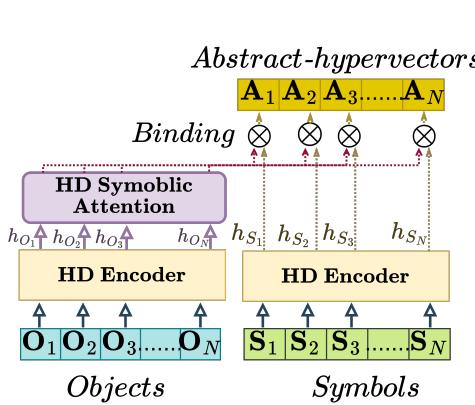


Figure 2: HDSymbolicAttention(O) $\otimes h_S$

Our proposed system shown in Figure 2 aims to resolve the relational bottleneck problem in a high dimensional space using an explicit *binding* mechanism. Similar to [1], the *object representations* are separated from the trainable symbol representations to minimize interference and construct abstract relationships between objects. A set of objects $O_{1..N} = [O_1, \dots, O_N]$ is forwarded to an *HD Encoder* function ($\phi_B : O \mapsto O \cdot W_B$) where $W_B \in \{-1, +1\}^{F \times D}$ is a learnable bipolar projection matrix that generates a set of high dimensional vectors $h_{O_{1..N}} = [h_{O_1}, \dots, h_{O_N}]$. The generated set of high dimensional vectors is then forwarded to an *HD Symbolic attention mechanism* that produces a new set of encoded high dimensional vectors. These vectors are then *bound* with the symbolic high dimensional vectors $h_{S_{1..N}} = S_{1..N} \cdot W_B$. The

final output is a set of abstract high dimensional vectors $A_{1..N} \in \mathbb{R}^{N \times D}$ that feeds into other layers of the network.

Figure 3 illustrates how the *HDSymbolicAttention* is applied to the set of objects $O_{1..N} = [O_1, \dots, O_N]$ of Figure 2. Each object $O_i, i \in \llbracket 1..N \rrbracket$ is mapped to a high dimensional vector h_{O_i} called *object hypervector* through a learnable projection function ϕ_B . All the objects and symbols are mapped to hypervectors using the *same* projection function. This set of object hypervectors is used to compute the attention scores matrix (before softmax normalization) $[R_1, \dots, R_N] \in \mathbb{R}^{N \times N}$. Each set of attention scores $R_i, i \in \llbracket 1..N \rrbracket$ is forwarded to a softmax function σ . Each encoded object hypervector (i.e, the output of the HDSymbolicAttention module at a node i) is a weighted summation of all the *object hypervectors*. The weights above, are the set of attention scores $\sigma(R_i)$. Each element r_{ij} of the attention scores $R_i = [r_{i1}, \dots, r_{iN}] \in \mathbb{R}^N$ represents the correlation (i.e, similarity or angle in the context of hyperdimensional computing) between the object hypervector h_{O_i} and the *context hypervector* defined as a h_{O_i} bundled h_{O_j} together ($h_{O_i} \oplus h_{O_j}$) (i.e, $r_{ij} = \cos(h_{O_i}, h_{O_i} \oplus h_{O_j})$). For a single head, the mechanism illustrated in Figure2 can be written as:

$$\text{HDSymbolicAttention}(O_{1..N}) \otimes h_s = \left[\sigma([R_1, \dots, R_N]) \cdot (O_{1..N} \cdot W_B)^T \right] \otimes (S_{1..N} \cdot W_B) \quad (3)$$

We note that the HDSymbolicAttention mechanism illustrated in Figure3 and Figure2 refers to a *single* head of a multi head attention. A multi head attention version was used in the prototype implemented in this research.

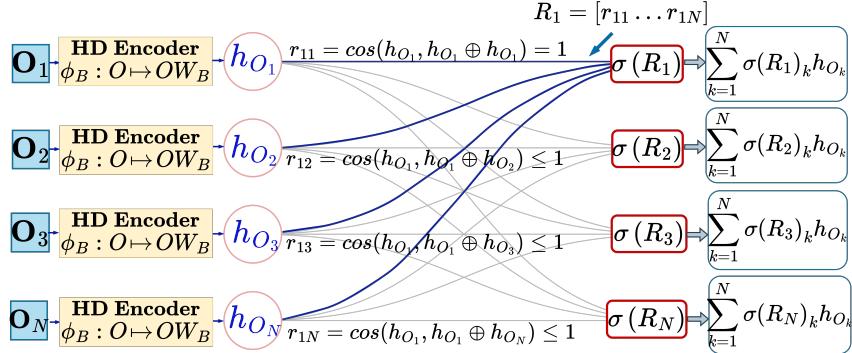


Figure 3: HDSymbolicAttention($O_{1..N}$)

5.1 Binarized HD attention score

Calculating attention scores is the most energy-intensive operation in self-attention [8]. We tackle this in our HDSymbolicAttention mechanism by taking advantage of the fact that binarization of the dot product in attention score calculation allows up to 2x *memory savings* and 60% *more speedup* with low accuracy loss [18] over real number multiply-accumulate computations used in transformer arithmetic.

Figure4 illustrates an example of our proposed binarized HDSymbolicAttention mechanism between two object hypervectors h_{O_1} and h_{O_2} involving well defined operations (i.e, binary AND, L_0 norm, element-wise real addition). The attention mechanism is expressed using the cosine similarity function \cos as follows: 5.1 $f(h_{O_1}, h_{O_2}) = \cos(h_{O_1}, h_{O_1} \oplus h_{O_2})$

Let b_{O_1} be the binary version of the bipolar hypervector derived from h_{O_1} by applying a *sign* function and a function $B(\cdot)$ (i.e., a shift and scale). The binary vector $b_{O_1 \oplus O_2}$ is obtained from the same process as b_{O_1} but applied to $h_{O_1} \oplus h_{O_2}$. Both b_{O_1} and $b_{O_1 \oplus O_2}$ are forwarded to an element-wise AND gate, and we derive the L_0 norm of the generated vector to get $w_{b_{O_1} \& b_{O_1 \oplus O_2}}$. Similarly, we obtain $w_{b_{O_1}}$ and $w_{b_{O_1 \oplus O_2}}$. Lemma 1 and the notation in Figure 4 allow us to simplify the function $r_{12} = f(h_{O_1}, h_{O_2})$ using binary operations according to Equation 4

$$f(h_1, h_2) = \frac{1}{D} (4w_{b_{O_1} \& b_{O_1 \oplus O_2}} - 2w_{b_{O_1}} - 2w_{b_{O_2}}) + 1 \quad (4)$$

Lemma 1. Given two D -dimensional vectors h_1 and h_2 that are bipolar meaning $h_{1/2,i} \in \{-1, +1\} \quad \forall i \in \llbracket 1, D \rrbracket$ and let $Bx : \mapsto \frac{x+1}{2}$ maps bipolar words to binary domain, & is the

binary AND and $\|\cdot\|_0$ is the zero-Norm. then cosine similarity between h_1 and h_2 can be expressed as following 5

$$\cos(h_1, h_2) = 1 + \frac{1}{D} [4(B(h_1) \& B(h_2))_0 - 2(B(h_1))_0 - 2(B(h_2))_0] \quad (5)$$

Proof. Let h_1 and h_2 two D-dimensional bipolar vectors then their cosine similarity could be expressed as $\cos(h_1, h_2) = \frac{\langle h_1, h_2 \rangle}{\|h_1\|_2 \cdot \|h_2\|_2}$ we have $\|h_1\|_2^2 = \|h_2\|_2^2 = \sum_{i=1}^D (\pm 1)^2 = D$. we also have $h_2 = 2B(h_2) - \mathbb{1}$ and $h_1 = 2B(h_1) - \mathbb{1}$, where $\mathbb{1}$ means a D-dimensional vector with all ones. Hence, 5.1 becomes

$$\cos(h_1, h_2) = \frac{\langle 2B(h_1) - \mathbb{1}, 2B(h_2) - \mathbb{1} \rangle}{D} = 1 + \frac{4 \|B(h_1) \& B(h_2)\|_0 - 2 \|B(h_1)\|_0 - 2 \|B(h_2)\|_0}{D} \quad (6)$$

□

6 LARS-VSA Architecture

The LARS-VSA architecture is composed of H HDSymbolicAttention heads (line 2). In (line 3) we apply the HDSymbolicAttention module discussed earlier to the set of objects $O_{1..N} = (O_1, O_2, \dots, O_N)$ and used the symbol hypervectors h_S^{h-1} to get the abstract rules in high dimensional space, $A_{1..N}^{h-1} = (A_1, \dots, A_N)$. In line 4, the abstract hypervectors $A_{1..N}^{h-1}$ are scaled using a BatchNormalization layer. Finally, all the abstract rules vector from different heads are *bundled* together in the real domain through a summation operation. In the line 7, the high dimensional abstract vectors $A_{1..N}^H$ are then compressed using a *Global Average Pooling* layer that reduces the dimensionality of the hypervectors to the original object's dimension. The *holistic* [10] property of hyperdimensional computing ensures none or low information loss after dimensionality reduction for hypervectors, avoiding catastrophic interference.

Algorithm 1 LARS-VSA Module

```

1:  $A_{1..N}^0 \leftarrow \emptyset$ 
2: for  $h \leftarrow 1$  to  $H$  do
3:    $A_{1..N}^{h-1} \leftarrow HDSymbolicAttention(O_{1..N}) \otimes (h_S^{h-1})$ 
4:    $A_{1..N}^{h-1} \leftarrow BatchNorm(A_{1..N}^{h-1})$ 
5:    $A_{1..N}^h \leftarrow A_{1..N}^h + A_{1..N}^{h-1}$ 
6: end for
7:  $A_{1..N}^H \leftarrow GlobalAvgPooling(A_{1..N}^H)$ 
8: return  $A_{1..N}^H$ 

```

Transformers perform object-level information retrieval and projection onto future outcomes from large volumes of training data, particularly in natural language processing, relying on decoding of relational representations. In [1], self-attention and cross-attention are utilized to construct a decoder capable of producing NLP-like solution instances. We integrated the decoder structure from [1] into our LARS-VSA pipeline. This excels in both partial as well as purely abstract reasoning tasks, necessitating a specific type of information generation, such as math problem-solving or object sorting. For partially abstract mathematical reasoning tasks, a skip connection between the encoder and decoder is required due to the identical structure of the input and output (both are

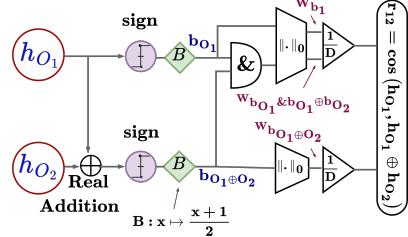


Figure 4: Binarized Attention Score Function $r_{12} = \cos(h_{O_1}, h_{O_1} \oplus h_{O_2})$

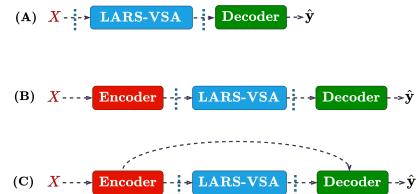


Figure 5: Examples of evaluated LARS-VSA based pipelines

text), as depicted in structure (C) of Figure 5. However, for purely abstract sorting problems, this skip connection is unnecessary, as illustrated in structure (B) of Figure 5.

For classification-based abstract reasoning tasks, like SET [1] or pairwise ordering [1], a simple decoder composed of linear layers suffices, as shown in structure (A) of Figure 5. The approach of [1] requires a transformer-based encoder to extract object-level correlations from initial inputs before forwarding them to abstractor modules, despite increasing the computational expenses of the already heavy abstractor pipeline. In our study, we employed a basic BatchNormalization layer followed by dropout to prevent overfitting. The Encoder here functions as a trainable standard scalar preprocessing step. Consequently, object-level correlation and abstraction are primarily provided by the LARS-VSA module. For sequence to sequence relational reasoning task, we used a cross-attention mechanism derived from [1] between encoded ground truth target and the generated abstract states from the LARS-VSA model.

7 Experiments

7.1 Discriminative Relational Tasks

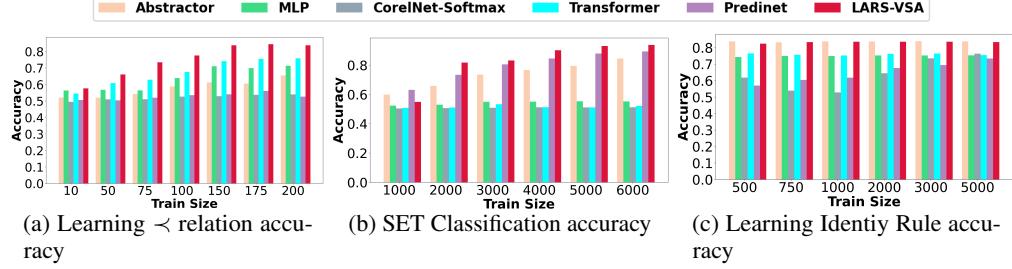


Figure 6: Experiments on discriminative relational tasks and comparison to SOTA.

Order relations: modeling asymmetric relations. We generated 64 random objects represented by iid Gaussian vectors $o_i \sim \mathcal{N}(0, I) \in \mathbb{R}^{32}$, and established an anti-symmetric order relation $o_1 \prec o_2 \prec \dots \prec o_{64}$. From 4096 possible object pairs (o_i, o_j) , 15% are used as a validation set and 35% as a test set. We train models on varying proportions of the remaining 50% and evaluate accuracy on the test set, conducting 10 trials for each training set size. The models must generalize based on the transitivity of the \prec relation using a limited number of training examples. The training sample sizes range between 10 and 200 samples. Figure 6a demonstrates the high capability of LARS-VSA to generalize with few examples, achieving over 80% accuracy with just 200 samples (1.07x better than the second best model and 1.33x better than Abstractor). The Transformer model is the second best performer, better than the Abstractor and CorelNet-Softmax due to the relatively moderate level of abstraction needed for learning asymmetric relations.

SET: modeling multi-dimensional relations. In the SET[1] task, players are presented with a sequence of cards, each varying along four dimensions: color, number, pattern, and shape. A triplet of cards forms a “set” if they either all share the same value or each have a unique value (as in Figure 7). In this experiment, the task is to classify triplets of card images as either a “set” or not. The shared architecture for processing the card images in all baselines as well as LARS-VSA is $\text{CNN} \rightarrow \{\cdot\} \rightarrow \text{Flatten} \rightarrow \text{Dense}$, where $\{\cdot\}$ is one of the aforementioned modules. The CNN embedder is obtained through a pre-training task. We compared the LARS-VSA architecture to CorelNet [12] with a softmax and a ReLU activation function, PrediNet [23], Abstractor [1], MLP [1], and a Transformer [25] architecture. For this specific task, there are four relational representations (e.g., shape, color, etc.) and one abstract rule (whether it is a triplet or not). Figure 6b shows LARS-VSA outperforms all the baselines (more than 1.05x better than the second best model and 1.11x better than Abstractor when training on 6000 samples), as it balances abstract rules with relational representations. In this particular case, PrediNet also shows high accuracy. Its relational vectors are less connected to object-level features than those of Transformers but more than those of the Abstractor.

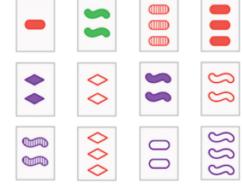


Figure 7: The SET game

ABA: Modeling Identity Relations The ABA experiment [28] aims to identify an abstract pattern in a sequence of three images from an image classification dataset, specifically the cats and dogs dataset [6]. The ABA rule requires the system to determine whether the sequence follows a "cat-dog-cat" or "dog-cat-dog" pattern. The images are processed using a pretrained VGG-like CNN (detailed in the Appendix) to extract the feature map corresponding to the final dense layer. We generated 20000 sequences of 3-image feature maps $o_i \in \mathbb{R}^{512} \forall 1 \leq i \leq 3$. The dataset is split into 50% for training, 25% for validation and 25% for testing. In this experiment, Figure 6c shows that the Abstractor and LARS-VSA show highest performance (i.e, with comparable accuracy) compared to the state of the art. The difference between this task and the two prior tasks is that the relational representation and the abstract rule (ABA) are harder to correlate since the objects are derived from real images.

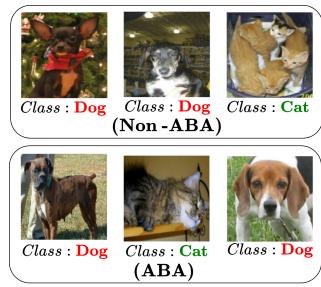


Figure 8: ABA Rule Identification Example

7.2 Object-sorting: Purely Relational Sequence-to-Sequence Tasks

We generate random objects as follows. First, we create two sets of random attributes: $\mathcal{A} = a_1, a_2, a_3, a_4$, where $a_i \stackrel{iid}{\sim} \mathcal{N}(0, I) \in \mathbb{R}^4$, and $\mathcal{B} = b_1, \dots, b_{12}$, where $b_i \stackrel{iid}{\sim} \mathcal{N}(0, I) \in \mathbb{R}^8$. Each set of attributes has a strict ordering: $a_1 \prec a_2 \prec a_3 \prec a_4$ for \mathcal{A} and $b_1 \prec b_2 \prec \dots \prec b_{12}$ for \mathcal{B} .

Our random objects are formed by taking the Cartesian product of these two sets, $\mathcal{O} = \mathcal{A} \times \mathcal{B}$, resulting in $N = 4 \times 12 = 48$ objects. Each object in \mathcal{O} is a vector in \mathbb{R}^{12} , created by concatenating one attribute from \mathcal{A} with one attribute from \mathcal{B} .

We then establish a strict ordering relation for \mathcal{O} , using the order relation of \mathcal{A} as the primary key and the order relation of \mathcal{B} as the secondary key. Specifically, $(a_i, b_j) \prec (a_k, b_l)$ if $a_i \prec a_k$ or if $a_i = a_k$ and $b_j \prec b_l$. We generated a randomly permuted set of 5 and a set of 6 objects in \mathcal{O} . The target sequences are the indices representing the sorted order of the object sequences (i.e., obtained using 'argsort'). The training data are uniformly sampled from the set of length- N (i.e, $N \in \{5, 6\}$) sequences in \mathcal{O} . Additionally, we generate non-overlapping validation and testing datasets in the following proportion: 20% for testing, 10% for validation and 70% for training.

We used *element wise accuracy* to assess the performance of LARS-VSA, also used in [1]. The accuracy of LARS-VSA is compared against the Relational Abstractor [1], Transformer and CorelNet[12]. We also examine LARS-VSA using binarized attention that uses a regular binarized attention mechanism instead of our binarized HDSymbolicAttention mechanism (i.e, instead of evaluating $f(h_{O_i}, h_{O_i} \oplus h_{O_j})$ we simply evaluate $f(h_{O_i}, h_{O_j})$), to assess the effect of vector orthogonality on LARS-VSA performance. This is referred to as the ablation model in this section.

As the number of elements to sort increases, the performance of all models decreases slightly due to the increasing complexity of the abstract rules. However, LARS-VSA achieves better accuracy than other baselines (between 1.66x and 2.25x better than Relational-Abstractor for 5 elements sorting and between 1.56x and 3.33x for 6 elements sorting) and the ablation model (around 1.1x for 5 elements sorting and 1.5x for 6 elements sorting). Relational-Abstractor [1] still outperforms the transformer and CorelNet-Softmax, validating the results obtained in [1]. LARS-VSA demonstrates greater generalizability compared to the ablation model, suggesting that vector orthogonality in high dimensions has a more significant impact as sequence length increases. This indicates that the attention score matrices of LARS-VSA become less informative as the sequence length increases.

7.3 Math problem-solving: partially-relational sequence-to-sequence tasks

The object-sorting experiments in Section 7.2 are characterized as "purely relational" because the set of pairwise \prec order relations provides sufficient information to solve the task. For general sequence-to-sequence tasks, there may not always be such a relation. In such cases an architecture with a relational bottleneck may exhibit high abstraction level which might give it an advantage over a purely object level architecture such as a transformer. We assessed the LARS-VSA architecture and other baselines on 4 different subset of the mathematical reasoning set of tasks [22]. Two of them are presented in Figure 10. The rest are described in the appendix. The baselines used to assess the LARS-VSA performance are: two version of Abstractor with symbolic relational

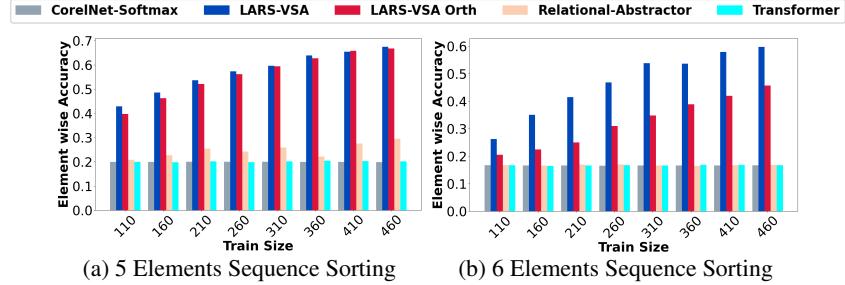


Figure 9: Performance of LARS-VSA and LARS-VSA orth compared to baselines. The CoreNet[12] uses *Softmax* activation

Task: numbers_list_prime_factors
 Question: What are the prime factors of 121
 Answer: 11, 11

Task: numbers_round_number_composed
 Question: Round 456.789 to 1 decimal place.
 Answer: 456.8

Figure 10: Examples of input/target sequences from the math problem-solving dataset.

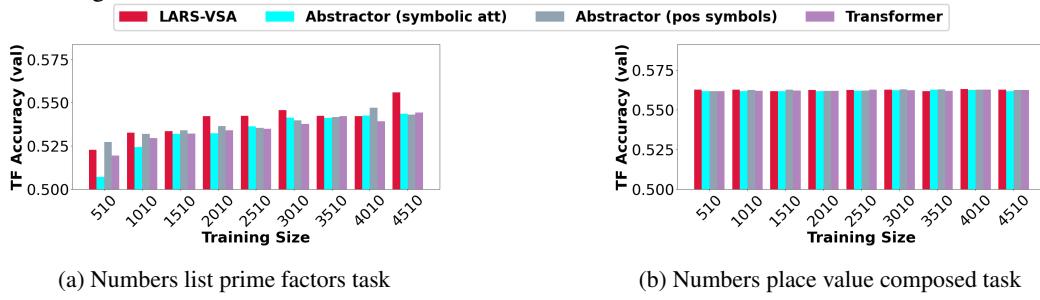


Figure 11: Accuracy of LARS-VSA compared to SOTA for two mathematical reasoning tasks

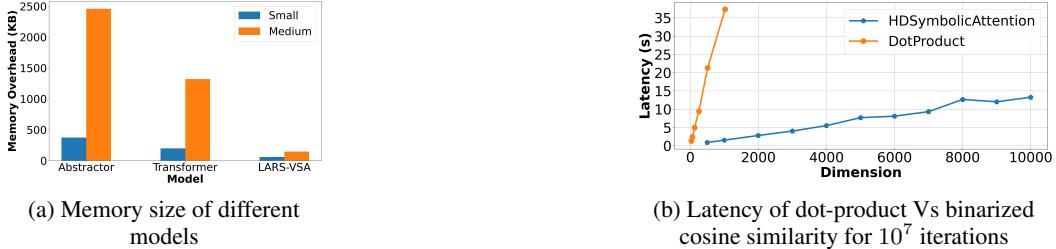


Figure 12: Memory overhead of LARS-VSA, Abstractor and Transformer Figure (a) and Latency of dot-product compared to binarized cosine similarity for 10^7 iterations Figure (b)

attention[1] and positional symbols[1], and a Transformer architecture. All architectures use a "small" Encoder-Decoder structure except LARS-VSA, which has a lightweight encoder. For the first task (Figure 11a), LARS-VSA outperforms other models by up to 4% in accuracy. For the second task (Figure 11b), all models have comparable accuracies. LARS-VSA excels in handling complex relational tasks with minimal overhead. We compared LARS-VSA's speedup and memory savings to a regular transformer and Abstractor for the mathematical reasoning task. Figure 12a shows the memory overhead in bits for Abstractor (symbolic attention), Transformer, and LARS-VSA using small and medium Encoder-Decoder structures. Figure 12b illustrates the latency of one elementary operation for self-attention (dot product) and HDSymbolicAttention (binarized attention score). LARS-VSA is up to 17x and 9x more memory efficient than Abstractor and Transformer with medium Encoder-Decoder structures, and up to 6x and 3x more efficient with small Encoder-Decoder architectures. From Figure 12b, HDSymbolicAttention and Dot product grow linearly with rates of 0.14×10^{-2} and 3.6×10^{-2} , respectively, indicating HDSymbolicAttention is about 25x faster across all latency-to-vector dimension ratios.

8 Conclusion, Limitation and Future Directions

In this paper, we introduce a novel *relational bottleneck* for vector symbolic architectures, utilizing a hyperdimensional computing-specific attention mechanism called *HDSymbolicAttention*. This mechanism binds correlated object feature high-dimensional vectors with symbolic abstract high-dimensional vectors, reducing interference between object-level features and abstract rules, effectively addressing the *curse of compositionality* problem. However, interference is not only spatial but also temporal, as relational representations can interfere with symbolic rules over time, potentially leading to *catastrophic forgetting*, a common issue in online-trained neural networks. Future work will explore the impact of this structure on online abstract reasoning abilities and develop a decoder (i.e, see Figure5) that relies solely on hyperdimensional computing capabilities, moving away from the current cost-inefficient self-attention and cross-attention mechanisms.

References

- [1] Awni Altabaa, Taylor Webb, Jonathan Cohen, and John Lafferty. Abstractors: Transformer modules for symbolic message passing and relational reasoning. *stat*, 1050:25, 2023.
- [2] Hussam Amrouch, Mohsen Imani, Xun Jiao, Yiannis Aloimonos, Cornelia Fermuller, Dehao Yuan, Dongning Ma, Hamza E Barkam, Paul R Gensler, and Peter Sutor. Brain-inspired hyperdimensional computing for ultra-efficient edge ai. In *2022 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*, pages 25–34. IEEE, 2022.
- [3] David Barrett, Felix Hill, Adam Santoro, Ari Morcos, and Timothy Lillicrap. Measuring abstract reasoning in neural networks. In *International conference on machine learning*, pages 511–520. PMLR, 2018.
- [4] Jeffrey S Bowers, Ivan I Vankov, Markus F Damian, and Colin J Davis. Neural networks learn highly selective representations in order to overcome the superposition catastrophe. *Psychological review*, 121(2):248, 2014.
- [5] Steven M Frankland, Taylor Webb, and Jonathan D Cohen. No coincidence, george: Capacity-limits as the curse of compositionality. 2021.
- [6] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Computer vision and pattern recognition (cvpr), 2012 ieee conference on. 2012.
- [7] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.
- [8] Tae Jun Ham, Sung Jun Jung, Seonghak Kim, Young H Oh, Yeonhong Park, Yoonho Song, Jung-Hun Park, Sanghee Lee, Kyoung Park, Jae W Lee, et al. A³: Accelerating attention mechanisms in neural networks with approximation. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 328–341. IEEE, 2020.
- [9] Michael Hersche, Mustafa Zegiri, Luca Benini, Abu Sebastian, and Abbas Rahimi. A neuro-vector-symbolic architecture for solving raven’s progressive matrices. *Nature Machine Intelligence*, 5(4):363–375, 2023.
- [10] Pentti Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive computation*, 1:139–159, 2009.
- [11] Pentti Kanerva. Hyperdimensional computing: An algebra for computing with vectors. *Advances in Semiconductor Technologies: Selected Topics Beyond Conventional CMOS*, pages 25–42, 2022.
- [12] Giancarlo Kerg, Sarthak Mittal, David Rolnick, Yoshua Bengio, Blake Richards, and Guillaume Lajoie. On neural architecture inductive biases for relational tasks. *arXiv preprint arXiv:2206.05056*, 2022.
- [13] Yeseong Kim, Jiseung Kim, and Mohsen Imani. Cascadehd: Efficient many-class learning framework using hyperdimensional computing. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 775–780. IEEE, 2021.

- [14] Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR, 2018.
- [15] Mohamed Mejri, Chandramouli Amarnath, and Abhijit Chatterjee. A novel hyperdimensional computing framework for online time series forecasting on the edge. *arXiv preprint arXiv:2402.01999*, 2024.
- [16] Nicolas Menet, Michael Hersche, Geethan Karunaratne, Luca Benini, Abu Sebastian, and Abbas Rahimi. Mimonets: Multiple-input-multiple-output neural networks exploiting computation in superposition. *Advances in Neural Information Processing Systems*, 36, 2024.
- [17] Shanka Subhra Mondal, Jonathan D Cohen, and Taylor W Webb. Slot abstractors: Toward scalable abstract visual reasoning. *arXiv preprint arXiv:2403.03458*, 2024.
- [18] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.
- [19] JC Raven. Raven’s progressive matrices: Western psychological services los angeles, 1938.
- [20] Matthew Ricci, Jinkyung Kim, and Thomas Serre. Same-different problems strain convolutional neural networks. *arXiv preprint arXiv:1802.03390*, 2018.
- [21] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *Advances in neural information processing systems*, 30, 2017.
- [22] David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*, 2019.
- [23] Murray Shanahan, Kyriacos Nikiforou, Antonia Creswell, Christos Kaplanis, David Barrett, and Marta Garnelo. An explicitly relational neural network architecture. In *International Conference on Machine Learning*, pages 8593–8603. PMLR, 2020.
- [24] Jakke Tamminen, Matthew H Davis, and Kathleen Rastle. From specific examples to general knowledge in language learning. *Cognitive psychology*, 79:1–39, 2015.
- [25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [26] Taylor Webb, Shanka Subhra Mondal, and Jonathan D Cohen. Systematic visual reasoning through object-centric relational abstraction. *Advances in Neural Information Processing Systems*, 36, 2024.
- [27] Taylor W Webb, Ishan Sinha, and Jonathan D Cohen. Emergent symbols through binding in external memory. *arXiv preprint arXiv:2012.14601*, 2020.
- [28] Taylor W Webb, Steven M Frankland, Awni Altabaa, Simon Segert, Kamesh Krishnamurthy, Declan Campbell, Jacob Russin, Tyler Giallanza, Randall O’Reilly, John Lafferty, et al. The relational bottleneck as an inductive bias for efficient abstraction. *Trends in Cognitive Sciences*, 2024.

A Appendix / supplemental material

Code and Reproducibility

Code, detailed experimental logs, and instructions for reproducing our experimental results are available at: <https://github.com/mmejri3/LARS-VSA>.

Discriminative Tasks

In this section, we provide comprehensive information on the architectures, hyperparameters, and implementation specifics of our experiments. All models and experiments were developed using TensorFlow. The code, along with detailed experimental logs and reproduction instructions, is available in the project’s public repository.

A.1 Computational Resources

For training the LARS-VSA and SOTA on the Discriminative relational tasks we used a CPU (11th Gen Intel® Core™ i7) For training LARS-VSA and SOTA on the purely and partially sequence to sequence abstract reasoning we used a cluster of 4 GPUs RTX4090 with 24GB of RAM each.

A.2 Discriminative Tasks

A.2.1 Pairwise Order

Each model in this experiment follows the structure: `input` → `{module}` → `flatten` → MLP, where `{module}` represents one of the described modules and MLP is a multilayer perceptron with one hidden layer of 32 neurons activated by ReLU.

LARS-VSA Architecture Each model is composed of a single head $H = 1$ and a hypervector of dimensionality equal to $D = 1000$. We use a dropout of 0.1 to avoid overfitting. The two hypervectors are flattened and passed to the hidden layers with 32 neurons with a ReLU activation. It is forwarded to a 1 neuron final layer with sigmoid activation.

Abstractor Architecture The Abstractor module utilizes the following hyperparameters: number of layers $L = 1$, relation dimension $d_r = 4$, symbol dimension $d_s = 64$, projection (key) dimension $d_k = 16$, feedforward hidden dimension $d_{ff} = 64$, relation activation function $\sigma_{rel} = \text{softmax}$. No layer normalization or residual connection is applied. Positional symbols, which are learned parameters of the model, are used as the symbol assignment mechanism. The output of the Abstractor module is flattened and passed to the MLP.

CoRelNet Architecture CoRelNet has no hyperparameters. Given a sequence of objects, $X = (x_1, \dots, x_m)$, standard CoRelNet [12] computes the inner product and applies the Softmax function. We also add a learnable linear map, $W \in \mathbb{R}^{d \times d}$. Hence, $\bar{R} = \text{Softmax}(R)$, $R = [\langle Wx_i, Wx_j \rangle]_{ij}$. The CoRelNet architecture flattens \bar{R} and passes it to an MLP to produce the output. The asymmetric variant of CoRelNet is given by $\bar{R} = \text{Softmax}(R)$, $R = [\langle W_1 x_i, W_2 x_j \rangle]_{ij}$, where $W_1, W_2 \in \mathbb{R}^{d \times d}$ are learnable matrices.

PrediNet Architecture Our implementation of PrediNet [23] is based on the authors’ publicly available code. We used the following hyperparameters: 4 heads, 16 relations, and a key dimension of 4 (see the original paper for the meaning of these hyperparameters). The output of the PrediNet module is flattened and passed to the MLP.

MLP The embeddings of the objects are concatenated and passed directly to an MLP. The MLP has two hidden layers, each with 32 neurons and a ReLU activation.

Training/Evaluation We use the cross-entropy loss and the AdamW optimizer with a learning rate of 10^{-4} . We use a batch size of 64. Training is conducted for 50 epochs. The evaluation is performed on the test set. The experiments are repeated 10 times and we take the mean accuracy and will report the standard deviation later.

A.2.2 SET

The card images are RGB images with dimensions of $70 \times 50 \times 3$. A CNN embedder processes these images individually, producing embeddings of dimension $d = 64$ for each card. The CNN is trained to predict four attributes of each card, after which embeddings are extracted from an intermediate layer and the CNN parameters are frozen. The common architecture follows the structure: CNN Embedder

$\rightarrow \{\text{Abstractor}, \text{CoRelNet}, \text{PrediNet}, \text{MLP}\} \rightarrow \text{Flatten} \rightarrow \text{Dense}(2)$. Initial tests with the standard CoRelNet showed no learning. By adjusting hyperparameters and architecture, we found that removing the softmax activation in CoRelNet improved its performance slightly. Details of hyperparameters are provided below.

Common Embedder Architecture: The architecture is structured as: Conv2D \rightarrow MaxPool2D \rightarrow Conv2D \rightarrow MaxPool2D \rightarrow Flatten \rightarrow Dense(64, relu) \rightarrow Dense(64, relu) \rightarrow Dense(2). The embedding is taken from the penultimate layer. The CNN is trained to perfectly predict the four attributes of each card, achieving near-zero loss.

LARS-VSA Architecture: The LARS-VSA module has the following hyperparameters: number of heads $H = 1$, hypervector dimension $D = 1000$. The output are flattened and then feedforward hidden dimension $d_{\text{ff}} = 64$ and then to a final layer with a single neuron with *sigmoid* activation. We used a dropout of 0.4 to avoid overfitting.

Abstractor Architecture: The Abstractor module has the following hyperparameters: number of layers $L = 1$, relation dimension $d_r = 4$, symmetric relations ($W_q^i = W_k^i$ for $i \in [d_r]$), a relation activation ReLU, symbol dimension $d_s = 64$, projection (key) dimension $d_k = 16$, feedforward hidden dimension $d_{\text{ff}} = 64$, and no layer normalization or residual connection. Positional symbols, which are learned parameters, are used as the symbol assignment mechanism.

CoRelNet Architecture: The standard CoRelNet, described above, computes $R = \text{Softmax}(A)$, where $A = [\langle Wx_i, Wx_j \rangle]_{ij}$. This variant was stuck at 50% accuracy regardless of the training set size. Removing the softmax improved performance. Figure 6b compares both variants of CoRelNet.

This finding indicates that making σ_{rel} a configurable hyperparameter is beneficial. Softmax normalizes relations contextually, which might be useful at times but can hinder relational models when absolute relations between object pairs are needed, independent of other relations.

PrediNet Architecture: We used the following hyperparameters: 4 heads, 16 relations, and a key dimension of 4 (as per the original paper). The output of the PrediNet module is flattened and passed to the MLP.

MLP: The embeddings of the objects are concatenated and passed directly to an MLP. The MLP has two hidden layers, each with 32 neurons and ReLU activation.

Data Generation: Data is generated by randomly sampling a “set” with probability 1/2 and a non-“set” with probability 1/2. The triplet of cards is then randomly shuffled.

Training/Evaluation: We use cross-entropy loss and the AdamW optimizer with a learning rate of 10^{-4} . The batch size is 512. Training is conducted for 200 epochs. Evaluation is performed on the test set. We train our model on a set of randomly N number of samples $N \in \{1000, 2000, 3000, 4000, 5000, 6000\}$

A.2.3 ABA



Figure 13: CNN Feature Extractor

This task consists of an Identity rule ABA, so the input is a sequence of 3 vectors of shape 512. The class of each vector is either A or B. If the sequence is ordered in an ABA order the relational model should label it as 1 otherwise as 0. The dataset used is CatVsDog with an 20000 images in total. The CNN used as a Feature extractor is shown in Figure 13. It is trained over 100 epochs with an Adam optimizer with $lr = 10^{-3}$. The last layer of the CNN is taken as a feature map of shape 512. We trained the relational models on N samples with $N \in \{500, 750, 1000, 2000, 3000, 5000\}$. We used the same baseline settings as SET experiment.

A.3 Relational sequence-to-sequence tasks

A.3.1 Object-sorting task

LARS-VSA Architecture We used the architecture (B) of Figure 5. The encoder used in a BatchNormalization layer. The LARS-VSA model includes 2 heads $H = 2$ with a hyperdimensional dimension of $D = 1000$. The decoder used 4 layers, 2 attention heads, a feedforward network with 64 hidden units, and a model dimension of 64.

Abstractor Architecture Each of the Encoder, Abstractor, and Decoder modules consists of $L = 2$ layers, with 2 attention heads/relation dimensions, a feedforward network with $d_{\text{ff}} = 64$ hidden units, and a model/symbol dimension of $d_{\text{model}} = 64$. The relation activation function is $\sigma_{\text{rel}} = \text{Softmax}$. Positional symbols are utilized as the symbol assignment mechanism, which are learned parameters of the model.

Transformer Architecture We implement the standard Transformer architecture as described by [25]. Both the Encoder and Decoder modules use matching hyperparameters per layer, with an increased number of layers. Specifically, we use 4 layers, 2 attention heads, a feedforward network with 64 hidden units, and a model dimension of 64.

Training and Evaluation The models are trained using cross-entropy loss and the Adam optimizer with a learning rate of $5 \cdot 10^{-4}$. We use a batch size of 64 and train for 150 epochs. To evaluate learning curves, we vary the training set size, sampling random subsets ranging from 110 to 460 samples in increments of 50. Each sample consists of an input-output sequence pair. For each model and training set size, we perform 10 runs with different random seeds, reporting the mean of accuracy.

A.3.2 Math Problem-Solving

The dataset comprises various math problem-solving tasks, each featuring a collection of question-answer pairs. These tasks cover areas such as solving equations, expanding polynomial products, differentiating functions, predicting sequence terms, and more. Figure 16 provides an example of these question-answer pairs. The dataset includes 2 million training examples and 10,000 validation examples per task. Questions are limited to a maximum length of 160 characters, while answers are restricted to 30 characters. Character-level encoding is utilized, employing a shared alphabet of 95 characters, which includes upper and lower case letters, digits, punctuation, and special tokens for start, end, and padding.

Abstractor Architectures. The Encoder, Abstractor, and Decoder modules share identical hyperparameters: number of layers $L = 1$, relation dimension/number of heads $d_r = n_h = 2$, symbol dimension/model dimension $d_s = d_{\text{model}} = 64$, projection (key) dimension $d_k = 32$, feedforward hidden dimension $d_{\text{ff}} = 128$. The relation activation function in the Abstractor is $\sigma_{\text{rel}} = \text{softmax}$. One model employs positional symbols with sinusoidal embeddings, while the other utilizes symbolic attention with a symbol library of $n_s = 128$ learned symbols and 2-head symbolic attention.

Transformer Architecture. The Transformer Encoder and Decoder possess hyperparameters identical to those of the Encoder and Decoder in the Abstractor architecture.

LARS-VSA Architectures. The LARS-VSA uses the architecture (C) of Figure 5. We used the same Decoder as the Abstractor architecture. However, the encoder structure is limited to a BatchNormalization layer. The LARS-VSA has 2 heads $H = 2$ and a hyperdimensional dimension of $D = 1000$.

Training/Evaluation. Each model is trained for 150 epochs using categorical cross-entropy loss and the Adam optimizer with a learning rate of 6×10^{-4} , $\beta_1 = 0.9$, $\beta_2 = 0.995$, and $\varepsilon = 10^{-9}$. The batch size used is 16. The training set is composed of N samples $N \in \{510, 1010, 1510, 2010, 2510, 3010, 3510, 4010, 4510\}$.

A.4 Additional Results

Standard deviation discriminative relational tasks for the Discriminative relational tasks (pairwise ordering, SET, ABA) we repeated the experiment 10 times. In the previous section we presented the mean value, In the Figure 14 we presented the standard deviation of the mean value.

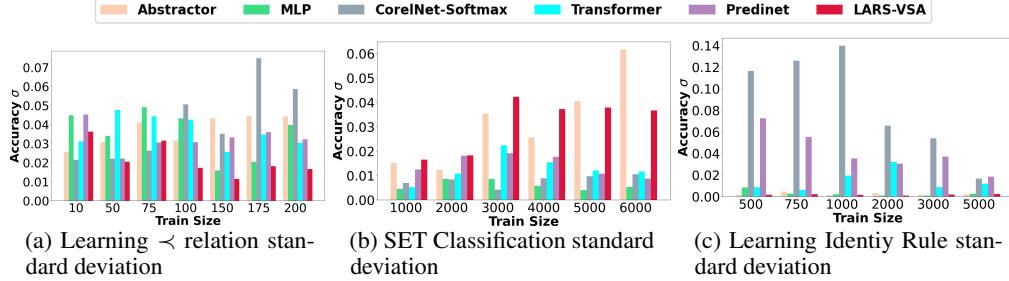


Figure 14: Standard Deviation of experiments on discriminative relational tasks and comparison to SOTA.

It turns out from Figures 14c and 14a that LARS-VSA showed the highest robustness to random seed proving its stability to randomness. For Figure 14b, LARS-VSA showed higher standard deviation compared to SOTA but it's still low ($\sigma \leq 0.06$)

Task: algebra_linear_2d Question: Solve for x and y: $3x + 2y = 6$ and $x - y = 2$ Answer: $x = 2$, $y = 0$	Task: numbers_is_prime Question: Is 97 a prime number? Answer: Yes
---	---

Figure 15: Examples of input/target sequences from the math problem-solving dataset.

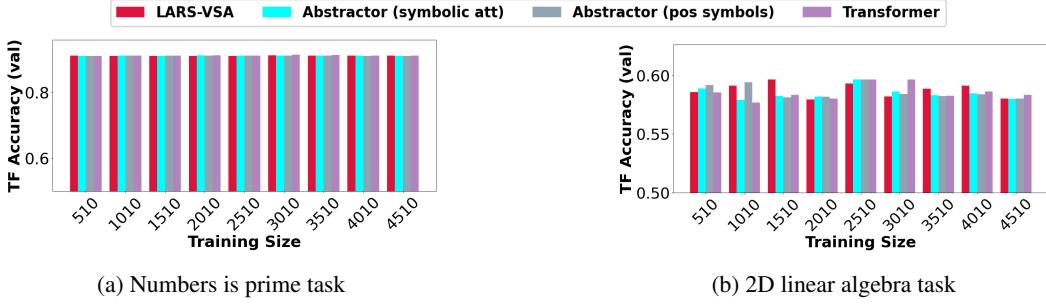


Figure 16: Accuracy of LARS-VSA compared to SOTA for two mathematical reasoning tasks

Mathematical reasoning tasks Figures 16a and 16b show that LARS-VSA have better or comparable results compared to Abstractor and Transformer.

B Multi-Attention Decoder

This Decoder is used for purely and partially sequence to sequence abstract reasoning tasks. It is derived from [1].