

## 7 Experimental Details

**The experimental setting of synthetic data.** We conduct experiments on a synthetic dataset composed of 16 distinct basic elements, which are shown in Figure 6 (a), where each panel is constructed by selecting and combining two of these elements. The examples of training and test data are displayed in Figure 6 (b-c). The model architecture consists of a single-layer Transformer using only self-attention, omitting the feedforward layer. The input sequence length is fixed at  $N = 32$ , with a feature dimension of 16 and a single attention head ( $H = 1$ ). Training is performed over 200 epochs using a batch size of 128. We optimize the model using the Adam optimizer with a learning rate of 0.001 and employ mean squared error (MSE) loss as the training objective.

**The experimental setting of S-RAVEN.** We evaluate on the S-RAVEN benchmark [17], where each task is defined by 4 features, sampled from a pool of 8 possible rules. For each task, we generate three input-output sequences of length three, using random inputs for each rule to form the context. Our model architecture follows Hyla [17], varying the number of layers between 4 and 8. The input has a feature dimension of 128 and 16 attention heads ( $H = 16$ ). For baseline comparisons, including Hyla and a standard Transformer, we adopt the original configurations as specified in the Hyla paper. All models use Root Mean Square (RMS) normalization for attention activations. To promote structured representations, our method applies soft thresholding to the attention weights, encouraging sparsity. Training is conducted for one epoch using a batch size of 128, the Adam optimizer with a learning rate of 0.001 and a weight decay of 0.1, and the cross-entropy loss as the objective.

**The experimental setting of RAVEN.** We conduct experiments on a restricted version of the RAVEN dataset [25], focusing solely on the 2-by-2 grid layout. To ensure deterministic target generation, we remove stochastic variations in rotation and color, so that the target panel is uniquely determined by the eight context panels. Each image is resized to  $40 \times 40$  pixels. The model is a standard Transformer with 4 layers, a sequence length of  $N = 36$ , a feature dimension of 512, and 16 attention heads ( $H = 16$ ). Training is performed over 2000 epochs with a batch size of 256, using the Adam optimizer with a learning rate of 0.0001. The model is trained to minimize mean squared error (MSE) loss.

## 8 Experimental Results

**Sparsity and threshold** To investigate the impact of the threshold on attention sparsity, we conduct experiments on the RAVEN dataset. Specifically, we measure the average sparsity of the attention maps across all layers, where sparsity is defined as the proportion of zero-valued entries after thresholding. As the threshold increases, more small-magnitude values are suppressed, leading to higher sparsity levels. Our results confirm this trend: larger thresholds consistently yield sparser attention maps, demonstrating the controllable nature of sparsity in our model through the threshold parameter. The results are shown in Table 1.

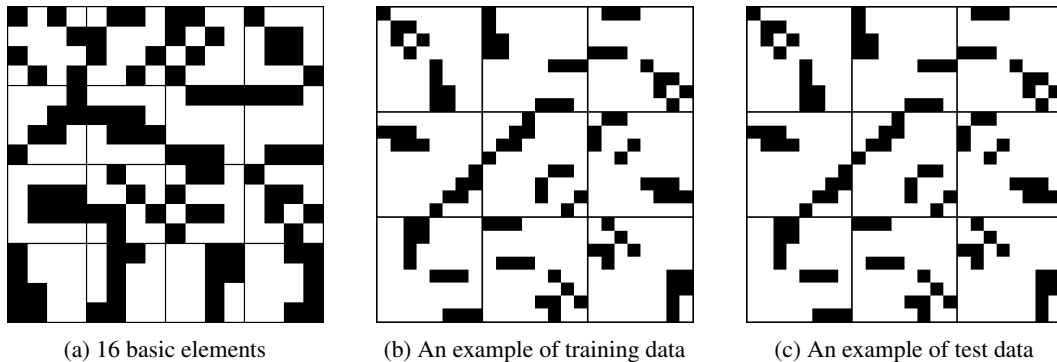


Figure 6: Examples of the synthetic dataset.

Table 1: The effect of threshold on the sparsity of the attention map.

Threshold ( $\xi$ )	0.003	0.01	0.03	0.1	0.3
Sparsity	18.53	57.82	90.45	97.82	99.38

Table 2: Variation of basis functions.

Configs of $\phi(\mathbf{X})$ and $\psi(\mathbf{X})$	$\mathbf{W}_{qk}^{(h)}\mathbf{X}, \mathbf{W}_{vo}^{(h)}\mathbf{X}$	$\text{ReLU}(\mathbf{W}_{qk}^{(h)}\mathbf{X}), \mathbf{W}_{vo}^{(h)}\mathbf{X}$	$\mathbf{W}_{qk}^{(h)}\mathbf{X}, \text{ReLU}(\mathbf{W}_{vo}^{(h)}\mathbf{X})$	$\text{ReLU}(\mathbf{W}_{qk}^{(h)}\mathbf{X}), \text{ReLU}(\mathbf{W}_{vo}^{(h)}\mathbf{X})$
Accuracy	71.7	72.3	72.9	73.6

**Variation of basis functions.** With the above formulation, we explore different designs for the basis functions  $\phi(\cdot)$  and  $\psi(\cdot)$  to adjust the expressiveness of models. In the baseline configuration, the basis functions are constructed through linear projections of the input, parameterized by  $\mathbf{W}_{qk}^{(h)}$  or  $\mathbf{W}_{vo}^{(h)}$ . A simple variation is to introduce nonlinearity into the basis construction by applying an activation function, such as ReLU, after the linear projections. For instance, a different basis function can be redefined as  $\phi(\mathbf{X}) = \text{ReLU}(\mathbf{W}_{qk}^{(h)}\mathbf{X})$  or  $\psi(\mathbf{X}) = \text{ReLU}(\mathbf{W}_{vo}^{(h)}\mathbf{X})$ . Incorporating nonlinearity into the basis functions can increase the representational capacity, enabling the model to capture more complex localized patterns beyond those achievable with purely linear projections.

We conduct experiments on the S-RAVEN dataset using a 4-layer Transformer architecture, training the model on a dataset of 20 million samples. We compare the different designs of  $\phi(\mathbf{X})$  and  $\psi(\mathbf{X})$  by adding the ReLU. The results are shown in Table 2.

## 9 Additional Analysis

By representing an input  $\mathbf{X}$  as  $[\mathbf{X}_1, \dots, \mathbf{X}_L]^\top$ , where  $\mathbf{X}_i, \forall i = 1, \dots, L-1$  and  $\mathbf{X}_L \in \mathbb{R}^{\frac{N}{L} \times d}$  are corresponding to context tasks and the target task, we have,

$$\begin{bmatrix} \mathbf{Z}_1 \\ \vdots \\ \mathbf{Z}_L \end{bmatrix} = \begin{bmatrix} \sigma(\mathbf{X}_1 \phi(\mathbf{X})) \psi(\mathbf{X}) \\ \vdots \\ \sigma(\mathbf{X}_L \phi(\mathbf{X})) \psi(\mathbf{X}) \end{bmatrix} = \begin{bmatrix} \alpha_1 \psi(\mathbf{X}) \\ \vdots \\ \alpha_L \psi(\mathbf{X}) \end{bmatrix}. \quad (15)$$

We set  $\mathbf{X}_L = \mathbf{0}$ , where  $\mathbf{0} \in \mathbb{R}^{\frac{N}{L} \times d}$  is a matrix with all zeros, since no observation for the target task.

**Our method.** Different from standard Transformer, our method enforces sparsity in coefficients by applying  $\sigma(\cdot) = \text{prox}(\cdot)$  to obtain  $\alpha_L = \sigma(\mathbf{X}_L \phi(\mathbf{X})) = \text{prox}(\mathbf{0}) = \mathbf{0}$ , which produces

$$\mathbf{Z}_L = \alpha_L \psi(\mathbf{X}) = \mathbf{0}. \quad (16)$$

This indicates that no estimation of the target output is made when there is no observation of the input. However, with the coefficient estimation (9),  $\alpha_L \leftarrow \alpha_L + \sum_{i=1}^{L-1} \lambda_i \alpha_i$ , we avoid a zero estimation of the target coefficients by linearly combining the coefficients of the context tasks, and produce nonzero output,

$$\mathbf{Z}_L = \alpha_L \psi(\mathbf{X}) + \sum_{i=1}^{L-1} \lambda_i \alpha_i \psi(\mathbf{X}). \quad (17)$$

Without coefficient estimation, neither standard Transformer nor our method yields informative outputs for  $\mathbf{Z}_L$ . However, by learning  $\lambda_i$  and leveraging the accurate reconstruction of context examples by  $\mathbf{Z}_i, \forall i = 1, \dots, L-1$ ,  $\mathbf{Z}_L = \alpha_L \psi(\mathbf{X}) + \sum_{i=1}^{L-1} \lambda_i \alpha_i \psi(\mathbf{X})$  is capable to generate the target outputs that reuse compositional rules from the context tasks.

### 9.1 Compositional Reconstruction of the Target Output

We have a dictionary of basis elements,  $\psi(\mathbf{X}) = \{\psi_j\}_{j=1}^N$ . Each output  $\mathbf{Z}_i$  for  $i = 1, \dots, L$  is expressed as a linear combination of elements in  $\psi(\mathbf{X})$  using coefficient vectors  $\alpha_i \in \mathbb{R}^N$ , i.e.,

$$\mathbf{Z}_i = \sum_{j=1}^n \alpha_i^{(j)} \psi_j = \alpha_i^\top \psi, \quad (18)$$

740 where  $\psi = [\psi_1, \dots, \psi_N]^\top$ .

741 **Assumption 9.1.** The dictionary  $\psi(\mathbf{X})$  is **sufficient** to represent the target output  $\mathbf{Z}_L$ .

742 **Assumption 9.2.** Each of the  $L - 1$  outputs  $\mathbf{Z}_1, \dots, \mathbf{Z}_{L-1}$  is correctly constructed using coefficient  
743 vectors  $\alpha_1, \dots, \alpha_{L-1}$ .

744 **Assumption 9.3.** Across  $\{\mathbf{Z}_1, \dots, \mathbf{Z}_{L-1}\}$ , every dictionary element  $\psi_j$  is used at least once, i.e.,  $\forall j$ ,  
745 there exists  $i$  such that  $\alpha_i^{(j)} \neq 0$ .

746 **Proposition 9.4.** There exists a set of weights  $\lambda_1, \dots, \lambda_{L-1}$  such that:

$$\alpha_L = \sum_{i=1}^{L-1} \lambda_i \alpha_i, \quad (19)$$

747 and  $\alpha_L$  reconstructs  $\mathbf{Z}_L$  using only elements in  $\psi(\mathbf{X})$ .

748 *Proof.* Let  $\mathcal{A} = \{\alpha_1, \dots, \alpha_{L-1}\} \subset \mathbb{R}^N$  denote the set of known coefficient vectors. Let  $V =$   
749  $\text{span}(\mathcal{A}) \subseteq \mathbb{R}^N$  be the subspace spanned by them. Since from Assumption 9.2, each  $\alpha_i$  reconstructs  
750  $\mathbf{Z}_i$  correctly and the union of their support covers all dictionary elements, the span  $V$  includes  
751 directions along all dictionary elements used for constructing  $\mathbf{Z}_L$ .

752 From Assumption 9.1, we know there exists some  $\alpha_L^* \in \mathbb{R}^N$  such that:

$$\mathbf{Z}_L = \alpha_L^{*\top} \psi. \quad (20)$$

753 Because  $\text{supp}(\alpha_L^*) \subseteq \bigcup_{i=1}^{L-1} \text{supp}(\alpha_i)$ , i.e., the dictionary elements needed for  $\mathbf{Z}_L$  have already been  
754 used in  $\mathcal{A}$ , and all such directions are already present in  $V$ , it follows that:

$$\alpha_L^* \in V. \quad (21)$$

755 Therefore, there exist scalars  $\lambda_1, \dots, \lambda_{L-1}$  such that:

$$\alpha_L^* = \sum_{i=1}^{L-1} \lambda_i \alpha_i.$$

756 Thus, by setting  $\alpha_L := \sum_{i=1}^{L-1} \lambda_i \alpha_i$ , we obtain the desired coefficient vector such that:

$$\mathbf{Z}_L = \alpha_L^\top \psi.$$

757 □

758 Given that the dictionary is sufficient, and the  $L - 1$  outputs collectively utilize all necessary dictionary  
759 elements, the coefficient vector for the  $L$ -th output can be expressed as a linear combination of  
760 previous coefficient vectors. This demonstrates the ability to transfer compositional rules from  
761 context examples to new tasks via linear combination of coefficients.

## 762 10 Computational Resource

763 We conducted development and experiments on a Linux workstation equipped with a single NVIDIA  
764 A5000 GPU (24GB memory). A single run of the synthetic task typically takes 3–5 minutes, while a  
765 single S-RAVEN experiment run takes between 60 and 200 minutes. For RAVEN experiments, a full  
766 run requires approximately 200 minutes.