Discrete Dictionary-based Decomposition Layer for Structured Representation Learning

Taewon Park[1] **Hyun-Chul Kim**[1] **Minho Lee**[1,2]
[1]Kyungpook National University, South Korea
[2]ALI Co., Ltd., South Korea
ptw7998@gmail.com, hyunchul_kim@knu.ac.kr, mholee@gmail.com

Abstract

Neuro-symbolic neural networks have been extensively studied to integrate symbolic operations with neural networks, thereby improving systematic generalization. Specifically, Tensor Product Representation (TPR) framework enables neural networks to perform differentiable symbolic operations by encoding the symbolic structure of data within vector spaces. However, TPR-based neural networks often struggle to decompose unseen data into structured TPR representations, undermining their symbolic operations. To address this decomposition problem, we propose a **D**iscrete **D**ictionary-based **D**ecomposition (D3) layer designed to enhance the decomposition capabilities of TPR-based models. D3 employs discrete, learnable key-value dictionaries trained to capture symbolic features essential for decomposition operations. It leverages the prior knowledge acquired during training to generate structured TPR representations by mapping input data to pre-learned discrete features within these dictionaries. D3 is a straightforward drop-in layer that can be seamlessly integrated into any TPR-based model without modifications. Our experimental results demonstrate that D3 significantly improves the systematic generalization of various TPR-based models while requiring fewer additional parameters. Notably, D3 outperforms baseline models on the synthetic task that demands the systematic decomposition of unseen combinatorial data.[1]

1 Introduction

Compositional generalization, aiming at understanding unseen data by combining known concepts, is essential for neural networks to handle complex tasks [2, 13, 12, 16, 8, 6]. Tensor Product Representation (TPR) framework [33] facilitates this by embedding the symbolic structure of data within vector spaces, providing neural networks with compositional capabilities. Within this framework, individual objects are decomposed at the representation level into distinct symbolic components called *role-filler* pairs[2]. The TPR framework encodes each object by taking a tensor product of its *role* vector and *filler* vector, represented as $T = filler \otimes role$, and then superimposes them to represent multiple objects within a single representation. During decoding, the TPR framework retrieves specific *fillers*—essential for solving tasks—from the superimposed representation through matrix multiplication using an *unbinding operator* correlated to a particular *role*, $filler = T \cdot unbind$. This retrieved *filler* is then utilized in downstream tasks. Based on this property, TPR-based neural networks have demonstrated significant generalization and applicability in fields such as associative reasoning [28, 30], mathematical problem-solving [29], and natural language processing [9, 32, 21, 34].

[1]The code of D3 is publicly available at https://github.com/taewonpark/D3

[2]The *roles* and *fillers* depend on the task at hand. For example, in a tree structure, the *role* corresponds to a position within the tree, while the *filler* represents the label associated with that position [34]. In associative memory, the *role* is analogous to an associative key, and the *filler* corresponds to the associative value [28, 30].

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

https://doi.org/10.52202/079017-0668

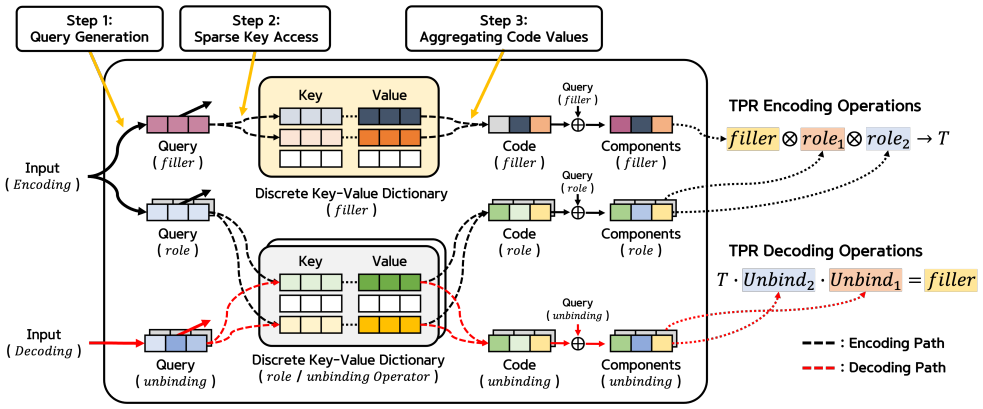

Figure 1: **Overview of D3.** D3 generates structured TPR representations by mapping input data to the nearest pre-learned symbolic features stored within discrete, learnable dictionaries. Each dictionary is linked explicitly to specific TPR components, such as *roles*, *filler*, and *unbinding operators*. Notably, D3 uses a shared dictionary configuration between the *roles* and *unbinding operators*. This figure illustrates, for example, that $role_1$ and $unbind_1$ share one dictionary, while $role_2$ and $unbind_2$ share another. T denotes a superimposed representation that represents multiple objects.

Despite their successes, the TPR-based approaches pose a significant challenge known as a decomposition problem [33, 23], which refers to the difficulty of decomposing input data into TPR components, such as *roles*, *fillers*, and *unbinding operators*. Without accurate decomposition, TPR-based models fail to represent the symbolic structure of data, causing a decline in the performance of the TPR operations. Recently, inspired by an object-centric learning method [18], Park et al. [23] proposes an attention-based iterative decomposition (AID) module to address this issue. AID uses competitive attention to iteratively refine structured representations, thereby enhancing the systematic generalization of TPR-based models. However, it still struggles to generalize all possible combinations of known symbols in simple synthetic tasks. This failure is likely attributable to its insufficient mechanism for explicitly mapping input data to known symbolic features observed during training. Therefore, the decomposition module may need an additional mechanism to store observed symbolic features during training and utilize it to effectively decompose unseen combinatorial data of known symbols.

In another line of work, discrete representation learning has been explored to improve the efficiency, interpretability, and generalization capabilities of neural networks [39, 14, 17, 37, 7]. This approach involves mapping continuous input data into discrete representations by finding the nearest features in a predefined codebook. The features within the codebook are learnable parameters, specifically trained to capture the latent features of data during training phase [39]. Some researchers have applied discrete representation techniques to extract specific types of representations from unstructured data [11, 43, 44]. Other researchers have integrated discrete symbolic embeddings within the TPR framework to improve its interpretability [21, 9]. However, these methods are designed for specific applications, such as question-answering and summarization tasks, making them difficult to integrate into other TPR-based models.

In this work, we propose a **D**iscrete **D**ictionary-based **D**ecomposition (D3) layer for structured representation learning within the TPR framework. D3 employs the discrete representations techniques to utilize prior knowledge acquired during training for decomposition operations. Inspired by prior discrete key-value architectures [14, 38], D3 consists of multiple dictionaries, each comprising discrete, learnable key-value pairs. Unlike prior work, each dictionary of D3 is linked explicitly to individual TPR components, such as *role*, *filler*, and *unbinding operator*. This design allows each dictionary to capture and store the discrete features of its corresponding TPR components during training. D3 acts as a drop-in layer that maps input data into pre-learned discrete features for the decomposition of TPR components through a three-step process, as illustrated in Fig. 1. First, it generates multiple queries from the input data, with each query utilized for different TPR components. Next, it identifies the nearest codebook keys within each dictionary based on these queries. Finally, D3 generates structured TPR representations by aggregating the codebook values corresponding to these keys. Moreover, D3 can be seamlessly integrated into any TPR-based model by replacing the TPR component generation layer without requiring further modifications.

Our main contributions are as follows.

- We propose a novel D3 layer to tackle the decomposition problem inherent in the TPR-based approaches. D3 leverages discrete, learnable dictionaries to enhance the decomposition capabilities of TPR-based models. By mapping input data to pre-learned discrete features stored within the dictionaries, D3 effectively generates structured TPR representations.
- We conduct extensive experiments across various systematic generalization tasks, including synthetic associative recall and text/visual question-answering tasks. Our experimental results show that D3 significantly enhances the generalization performance of TPR-based models, demonstrating its effectiveness on systematic generalization tasks.
- Our analyses show that D3 generates well-bound structured representations that are satisfactory for the requirements of the TPR framework, utilizing the discrete, learnable dictionaries.

2 Related Work

Decomposition Problem. Compositional generalization in neural networks, which allows for generalizing beyond training data, has been extensively studied [2, 13, 12, 16, 8, 6, 41]. One important capability for achieving this is a *segregation*, as discussed in Greff et al. [6], which enables the formation of meaningful representations from structured and unstructured data [3, 18]. TPR-based neural networks also rely on this capability to generate structured representations for TPR components such as *roles*, *fillers*, and *unbinding operators*. In the TPR framework, these structured representations must satisfy specific conditions to ensure accurate encoding and decoding. First, *roles* need to be linearly independent to avoid *filler* overlap. Second, the *unbinding operator* must correlate with the corresponding *roles* to accurately retrieve associated *fillers*. Recent work [23] has shown that existing TPR-based models often fail to generate structured representations that meet these conditions, undermining their symbolic operations. To address this, an attention-based decomposition module [23] has been introduced, but it still shows limited performance on synthetic tasks involving the decomposition of unseen combinatorial data. In this work, we address the decomposition problem within the TPR framework using a discrete dictionary-based method, advancing the research further.

Discrete Representation Learning. Discrete neural representation learning has introduced a codebook of discrete, learnable representations into neural networks [39]. During training, each discrete representation captures underlying latent features by mapping continuous input data to the nearest features within the codebook, which are then used for downstream tasks. Recent work on object-centric learning has utilized discrete representations to extract specific types of features from unstructured data, leveraging latent features learned during training [11, 43]. Some researchers have proposed a separate key-value codebook for learning discrete representations, demonstrating its effectiveness in systematic generalization [17] and robustness against distributional shifts [38]. Inspired by these findings, we develop a separate key-value-based discrete dictionary method to enhance the decomposition capabilities of TPR-based models. Other researchers have introduced a discrete symbolic embedding layer to improve the interpretability of TPR-based models, showing the feasibility of discrete representations in the TPR framework [21, 9]. However, their methods focus on encoding processes and specific tasks such as question-answering [21] and abstractive summarization [9]. In contrast, our work addresses the decomposition problem in TPR-based approaches, and our D3 method is a drop-in solution that can be easily adapted to any TPR-based model.

Memory Network. Research on memory networks has focused on enhancing neural network capacity by integrating external memory [36, 4, 5, 24, 27, 41]. Memory-augmented neural networks store variable lengths of sequential data in this external memory and retrieve necessary information using various addressing methods [36, 5]. These writing and reading mechanisms share many similarities with our D3 approach. However, while memory networks store input features sequentially in their memory states as a continuous stream, D3 updates symbolic feature information through gradient descent into codebook parameters within dictionaries. This distinctive characteristic allows D3 to leverage the learned discrete features to decompose unseen data after training. In another work, Lample et al. [14] introduces a learnable key-value memory layer to improve the efficiency of the Transformer by replacing the feed-forward layer. Unlike their memory layer, D3 employs key-value pairs in dictionaries explicitly linked to individual TPR components, making it well-suited for the TPR framework.

https://doi.org/10.52202/079017-0668

3 Method

In this section, we explain how the D3 module generates structured representations of the TPR components using discrete, learnable dictionaries. We then introduce configurations of D3 and how it can be applied to our baseline models.

3.1 Discrete Dictionary-based Decomposition module

D3 is a discrete dictionary-based drop-in layer designed to enhance the decomposition capabilities of TPR-based approaches. At every time step, D3 decomposes input data into TPR components, such as *roles*, *fillers*, and *unbinding operators*, by mapping input data to pre-learned symbolic features within dictionaries. These dictionaries consist of discrete, learnable codebook key-value pairs, denoted as $\{\mathcal{D}^j\}_{j=1}^{N_{\text{component}}}$ as shown in Eq. 1. Each dictionary $\mathcal{D}^j$ is explicitly linked to a j-th TPR component, allowing it to learn the symbolic features required for generating the specific TPR component. This design also enables the generation of structured representations for different TPR components individually and in parallel.

$$\mathcal{D}^j := \{(\mathsf{k}_i^j, \mathsf{v}_i^j) \mid \mathsf{k}_i^j \in \mathbb{R}^{D_{\text{query}}}, \mathsf{v}_i^j \in \mathbb{R}^{D_{\text{code}}}\}_{i=1}^{N_{\text{code}}} \quad \text{where } j = 1, ..., N_{\text{component}} \tag{1}$$

where $\mathcal{D}^j$ denotes the discrete, learnable dictionary for the j-th TPR component, k denotes a learnable codebook key, and v denotes a learnable codebook value. In the next paragraph, we describe how D3 generates TPR components using these dictionaries in three steps.

Step 1: Query Generation. At each time step t, D3 takes input data, denoted as $\texttt{input}_t \in \mathbb{R}^{D\text{input}}$, and generates the query, denoted as $\texttt{queries}_t \in \mathbb{R}^{N\text{component} \times D_{\text{input}}}$, for each j-th TPR component using a query network, $f_{\text{query}}^j : \texttt{input}_t \mapsto \texttt{query}_t^j \in \mathbb{R}^{D\text{query}}$. The query network can be any neural network; in this study, we use a feed-forward network with a single layer. Additionally, we apply a layer normalization [1] and a dropout of p_{dropout} [35] to $\texttt{query}_t^j$.

Step 2: Sparse Key Access. D3 searches for the nearest keys from each dictionary, $\mathcal{D}^j$, based on the generated $\texttt{query}_t^j$. We measure the similarity using the inner product between $\texttt{query}_t^j$ and $\{\mathsf{k}_i^j\}_{i=1}^{N_{\text{code}}}$. Then, D3 selects top-$k$ codebook keys in order of largest similarity, as follows.

$$\mathcal{I}^j = \mathcal{T}_k(\texttt{query}_t^{j\top} \hat{\mathsf{k}}_i^j) \quad \text{where } \hat{\mathsf{k}}_i^j = \mathsf{k}_i^j / ||\mathsf{k}_i^j||_2 \tag{2}$$

where $\mathcal{T}_k$ denotes the top-k operator that finds the indices of k largest values, and $\mathcal{I}^j$ denotes the indices of the k most similar keys within $\mathcal{D}^j$. We found that applying $L2$ normalization to keys before the inner product mitigates the codebook collapse problem.

Step 3: Aggregation of Code Values. D3 computes the normalized score for selected codebook keys, denoted as w_t^j, and aggregates codebook values corresponding to selected codebook keys with w_t^j, as follows.

$$\texttt{code}_t^j = \Sigma_{i \in \mathcal{I}} w_{t,i}^j \mathsf{v}_i^j \quad \text{where } w_t^j = \text{Softmax}(\texttt{query}_t^{j\top} \hat{\mathsf{k}}_i^j))_{i \in \mathcal{I}^j} \tag{3}$$

Then, D3 maps $\texttt{query}_t^j$ to a dimension of D_{code} and adds this projected vector to $\texttt{code}_t^j$. The summed vectors are mapped to a dimension of $D_{\text{component}}$ to generate structured representations of TPR components, as follows.

$$\texttt{component}_t^j = \texttt{code}_t^j + \text{layer}_{\text{residual}}(\texttt{query}_t^j) \in \mathbb{R}^{D_{\text{code}}} \tag{4}$$

$$\overline{\texttt{component}}_t^j = \text{layer}_{\text{final}}(\texttt{component}_t^j) \in \mathbb{R}^{D_{\text{component}}} \tag{5}$$

where $\text{layer}_{\text{residual}}$ and $\text{layer}_{\text{final}}$ denote a feed-forward network with a single layer. Those $\overline{\texttt{components}}_t$ are then utilized for TPR operations to solve the downstream tasks.

3.2 Module Configurations

In this section, we describe the configurations of D3 when applied to TPR-based models.

https://doi.org/10.52202/079017-0668

Shared Dictionary between Role and Unbinding Operator. As discussed in Section 2, *roles* and *unbinding operators* should have correlated features for accurate TPR operations. Considering this characteristic of the TPR framework, we share the dictionaries of *roles* and *unbinding operators*. This shared dictionary also reduces the number of learnable parameters.

D3 Applied to Filler. While the TPR framework requires specific conditions for *roles* and *unbinding operators* for accurate TPR operations, there are no such requirements for *fillers*. Therefore, we explore two configurations in this study: applying D3 to generate *fillers* (*w/ F*) and not applying D3 to generate fillers (*w/o F*). In the *w/o F* configuration, we follow the baseline models to generate the *filler* representations.

3.3 Integration of D3 into Existing TPR-based Models

In this section, we introduce our baseline models and explain how D3 is applied to them, considering the configurations of D3. We use three TPR-based models as our baselines: FWM [30], TPR-RNN [28], and Linear Transformer [10]. Notably, integrating D3 into these baseline models requires only substituting their TPR component generation layer with D3 without further modifications.

Fast Weight Memory. Fast Weight Memory (FWM) [30] is a TPR-based memory network designed for understanding long sequential contexts. It proposes a single word-level TPR operation related to the perceptron learning rule [25]. It has shown significant associative reasoning capability in reinforcement learning and natural language processing tasks. FWM requires two types of *roles* (*role*$_1$ and *role*$_2$) and one *filler* for encoding, as well as two types of *unbinding operators* (*unbind*$_1$ and *unbind*$_2$) for decoding. When D3 is integrated into FWM, it employs three dictionaries for the shared dictionary configuration: one for the *role*$_1$ and *unbind*$_1$, another for the *role*$_2$ and *unbind*$_2$, and the other for *filler*, as shown in Fig. 1.

TPR-RNN. TPR-RNN [28] is a sentence-level TPR-based memory network designed for basic text question-answering tasks [42]. It incorporates various encoding operations such as writing, moving, and backlink to process sequential data at the sentence level. These operations necessitate different encoding components with varying dimensions, making direct connections to the decoding components challenging. As a result, we do not apply the shared dictionary configuration to TPR-RNN; instead, we use a shared query network without layer normalization. Furthermore, due to the differing dimensions of the TPR components in TPR-RNN, we employ distinct layer$_{\text{final}}$ layers for each TPR component.

Linear Transformer. Linear Transformer [10] linearizes the attention mechanism to improve the computational efficiency of the Transformer [40]. Recently, Schlag et al. [31] demonstrated the equivalence between TPR and the linear attention mechanism, indicating that the key, value, and query in linear attention correspond to the *role*, *filler*, and *unbinding operator*, respectively. Building on this work, we apply D3 to generate the query, key, and value in the Linear Transformer. Unlike TPR-RNN and FWM, the Linear Transformer utilizes multi-head operations. Therefore, we use distinct dictionaries for each head, with the key and query of each head sharing the same dictionary.

4 Experiment

In this section, we evaluate the effectiveness of D3 across various tasks, including a synthetic task, text/visual question-answering tasks, and a language modeling task. To assess the decomposition capabilities, we follow the experimental settings of the AID [23], a prior work addressing the decomposition problem in the TPR framework, and closely compare our D3 model to baseline models and AID.

4.1 Task

Systematic Associative Recall (SAR) task. This task evaluates systematic generalization in memorizing and recalling combinatorial data [23]. It consists of a discovery phase and an inference phase. During the discovery phase, the model receives the combinatorial sequential items, each combining two symbols, $x \in X$ and $y \in Y$ where $X = X_1 \cup X_2 \cup X_3$ and $Y = Y_1 \cup Y_2$. The

 https://doi.org/10.52202/079017-0668

model is then required to predict an associated y when a specific x is presented. The SAR task uses different combination settings between training and evaluation to target systematic generalization specifically. During training, the model learns the following combination settings: (1) X_1 and Y_1, (2) X_2 and Y_2, and (3) X_3 and Y. At the evaluation, on the other hand, the model should generalize unseen combination settings, specifically X_1 and Y_2. Additionally, the task includes a hyper-parameter $p = \frac{|X_3|}{|X_2|+|X_3|}$ where $|X_i|$ denotes the cardinality of set X_i. By adjusting p, this task tests the systematic generalization of models under varying levels of exposure to different symbol combinations during training. In our study, we focus solely on the most challenging setting of the SAR task ($p = 0.0$), where the subset X_3 is excluded. In the SAR task, the TPR framework regards x as the *role* and the *unbinding operator*, and y as the *filler*. Therefore, TPR-based models should systematically decompose the combinatorial data into structured representations by mapping x to the *role* and y to the *filler* during the discovery phase, and mapping x to the *unbinding operator* during the inference phase to solve this task.

Systematic bAbI (sys-bAbI) task. This task is a variant of the bAbI task [42] designed to evaluate systematic generalization in text understanding and reasoning [23]. It consists of 20 distinct sub-tasks, each comprising stories, relevant queries, and corresponding answers. The sys-bAbI task requires the models to remember the stories and predict corresponding answers to the queries. Unlike the original bAbI task, the sys-bAbI task evaluates the models with two aspects: (a) in-distribution (*w/o sys diff*) and (b) with the systematic difference (*w/ sys diff*) where each sub-task includes unseen words during training. Therefore, the models should learn task-independent text understanding to solve the sys-bAbI task.

Sort-of-CLEVR task. This task [26] evaluates compositional generalization in visual relational reasoning. It consists of scene images, queries, and corresponding answers. This task requires the models to understand the properties of individual objects (*Unary*) or the relationships between multiple objects (*Binary* or *Ternary*) within visual scene images, and predict the correct answers to the queries [20]. Therefore, the model should capture relationships within each object and between objects to solve this task.

WikiText-103 task. This task [19] is a language modeling dataset consisting of lengthy corpora from Wikipedia. Although the WikiText-103 task does not directly measure the systematic generalization of the models, it is used to evaluate the effectiveness and applicability of D3 on a large-scale task beyond relatively simple tasks.

4.2 Experimental Results

In this section, we present the experimental results of the SAR task, sys-bAbI task, sort-of-CLEVR task, and WikiText-103 task. In our experiments, we set D_{query} as $D_{\text{code}}/2$.

4.2.1 TPR-based Memory Networks

First, we evaluate FWM with D3 on the SAR task, which requires understanding the composition of two types of symbols, x and y. TPR-based models are expected to solve this task perfectly by mapping each symbol to a specific TPR component during decomposition. However, as shown in Fig. 2, FWM and AID fail to generalize unseen combinations of known symbols. In contrast, our D3 module significantly outperforms other baseline models, achieving nearly 100% accuracy. This result demonstrates that D3 effectively decomposes unseen combinatorial data into TPR components using discrete dictionaries.

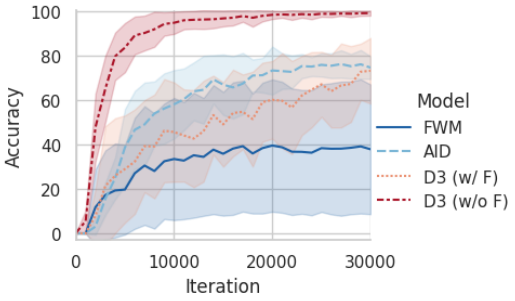

Figure 2: Test accuracy curve [%] on the SAR task for 10 seeds, with shadowed area indicating SD.

Next, we test TPR-RNN and FWM with D3 on the sys-bAbI task. This task involves compositional information in each story sentence, such as the relation between objects and their locations. It makes

Table 1: The mean word error rate [%] on the sys-bAbI task for 10 seeds, with ± indicating SD.

Model	*w/o sys diff* (↓)	*w/ sys diff* (↓)	**Gap** (↓)	# params (↓)
TPR-RNN	0.79 ± 0.16	8.74 ± 3.74	7.95	**0.14** M
+ AID	0.69 ± 0.08	5.61 ± 1.78	4.92	0.32 M
+ D3	**0.65** ± 0.25	**3.50** ± 2.07	**2.85**	0.17 M
FWM	0.79 ± 0.14	2.85 ± 1.61	2.06	**0.73** M
+ AID	**0.45** ± 0.16	**1.21** ± 0.66	**0.76**	1.23 M
+ D3 (*w/o F*)	0.79 ± 0.30	2.58 ± 1.12	1.79	0.75 M
+ D3 (*w/ F*)	0.75 ± 0.17	1.96 ± 0.88	1.21	0.75 M

Table 2: The mean accuracy [%] on the sort-of-CLEVR task for 10 seeds, with ± indicating SD.

Model	D_{code}	*Unary* (↑)	*Binary* (↑)	*Ternary* (↑)	# params (↓)
Linear Transformer	-	69.3 ± 14.8	75.5 ± 1.3	56.4 ± 4.3	**0.68** M
+ AID	-	98.9 ± 0.2	78.6 ± 0.3	63.7 ± 1.2	0.83 M
+ D3 (*w/o F*)	128	73.9 ± 16.5	77.2 ± 2.2	57.3 ± 4.6	0.75 M
	256	73.7 ± 16.5	77.8 ± 2.5	57.9 ± 5.8	0.96 M
+ D3 (*w/ F*)	128	98.9 ± 0.2	79.5 ± 0.8	63.1 ± 1.9	0.80 M
	256	**99.0** ± 0.3	**82.1** ± 2.4	**68.8** ± 1.2	1.13 M

Table 3: Perplexity on the WikiText-103 task.

Model	D_{code}	*Valid* (↓)	*Test* (↓)	# params (↓)
Linear Transformer	-	36.473	37.533	**44.02** M
+ AID	-	36.159	37.151	44.16 M
+ D3 (*w/o F*)	32	36.061	37.220	44.12 M
	64	**35.975**	**37.009**	44.36 M
+ D3 (*w/ F*)	32	36.630	37.620	44.22 M
	64	36.220	37.128	44.62 M

a sentence-level model more suitable for capturing the structural information of data than a word-level model. However, as shown in Table 1, TPR-RNN shows a larger performance gap between the *w/o sys diff* and *w/ sys diff* cases than FWM. Notably, D3 enhances the systematic generalization of both TPR-RNN and FWM with fewer additional parameters, significantly reducing the performance gap for TPR-RNN. These results highlight the efficacy of D3 in text understanding tasks.

4.2.2 Linear Transformer

We also evaluate the Linear Transformer with D3 on the sort-of-CLEVR task and WikiText-103 task. Following the AID [23], we use a 4-layered Linear Transformer with shared parameters for the sort-of-CLEVR task and apply D3 to a 16-layered Linear Transformer at intervals of 4 out of the 16 layers for the WikiText-103 task. As shown in Tables 2 and 3, D3 improves the performance of the Linear Transformer, with these improvements increasing as the capacity of the dictionaries grows. These results demonstrate the effectiveness of D3 on visual relational reasoning and language modeling tasks, as well as its applicability to the Linear Transformer. In addition, D3 shows comparable performance to the attention-based decomposition method, even with fewer parameters.

4.3 Analysis

In this section, we conduct a qualitative analysis of the structured TPR representations generated by D3 and an ablation study of D3. For these analyses, we experiment with D3 (*w/o F*) on the SAR task.

4.3.1 Qualitative Analysis

TPR framework requires its structured representations to satisfy the following conditions for accurate TPR operations: (*i*) linearly independence between distinct *roles*, and (*ii*) high correlation between

https://doi.org/10.52202/079017-0668

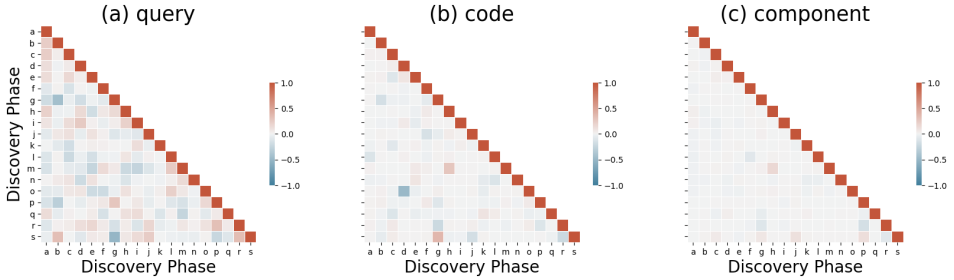

Figure 3: The heatmap displays the cosine similarity between the generated representations during the discovery phase for the SAR task. We explore the similarity across different types of representations: (a) `queries` of *roles*, (b) `codes` of *roles*, and (c) the *roles* themselves.

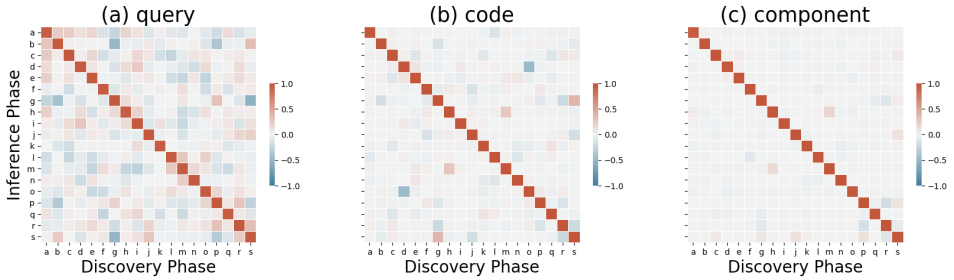

Figure 4: The heatmap displays the cosine similarity between the generated representations during the discovery phase (represented on the **x-axis**) and the inference phase (represented on the **y-axis**) for the SAR task. We explore the similarity across different types of representations: (a) `queries` of *roles* and *unbinding operators*, (b) `codes` of *roles* and *unbinding operators*, and (c) the *roles* and *unbinding operators* themselves.

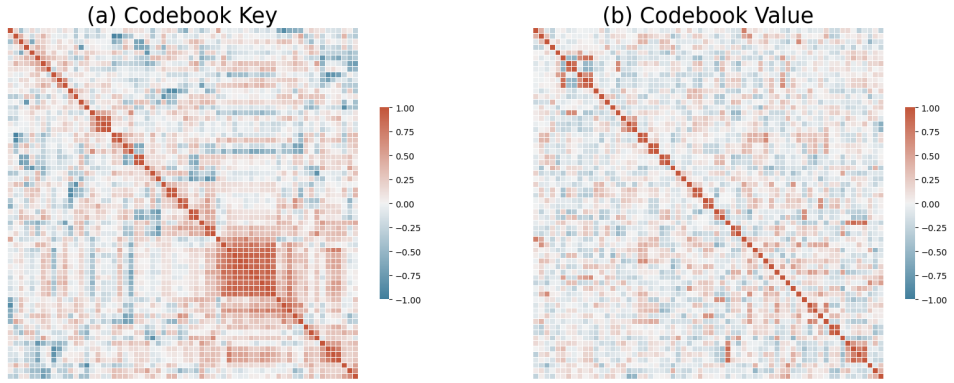

Figure 5: The heatmap visualizes the cosine similarity of the learned codebook features for the SAR task. There are two parts to each heatmap: (a) the similarity among codebook keys, denoted as $\{\mathsf{k}_i\}_{i=1}^{N_{\text{code}}}$, and (b) the similarity among codebook values, denoted as $\{\mathsf{v}_i\}_{i=1}^{N_{\text{code}}}$. For better visualization, the heatmap values are reordered to reflect the cluster of similar codebook keys.

role and *unbinding operator* for the same symbol x. We analyze the orthogonality of generated representations to investigate whether they satisfy these TPR conditions. Specifically, we consider the case of varying x while keeping y fixed for simplicity.

Fig. 3(c) shows the cosine similarity between the *roles* during the discovery phase, and Fig. 4(c) shows the cosine similarity between the *roles* during the discovery phase and the *unbinding operator* during the inference phase. Both results demonstrate that the generated representations by `D3` satisfy the TPR conditions, resulting in an accuracy of nearly 100%. We also conduct the same analysis for intermediate features, particularly `query` and `code`. Figs. 3 and 4 show that each intermediate representation complements the others to satisfy the TPR condition, indicating the effectiveness of `D3`.

https://doi.org/10.52202/079017-0668

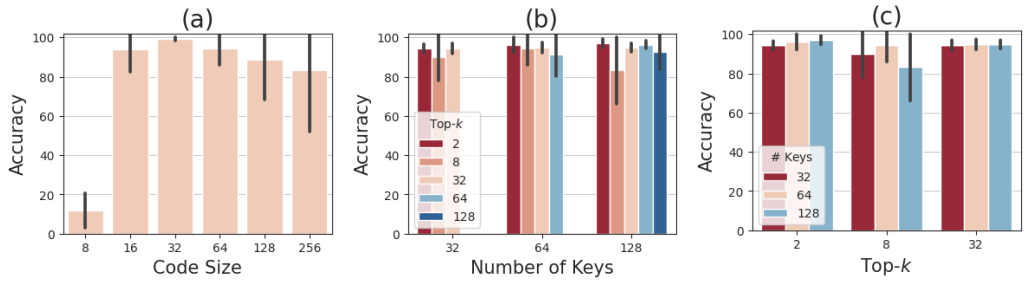

Figure 6: The mean accuracy on the SAR task for 10 seeds in the ablation study, with error bar indicating SD. The default setting uses D_{code} of 64, N_{code} of 64, and top-k of 8. Each figure shows the experimental results for the following settings: (a) Varying D_{code}. (b) Varying N_{code} with top-k constant. (c) Varying top-k with N_{code} constant.

Furthermore, we analyze the similarity patterns of codebook keys and codebook values. Fig. 5 shows that the codebook features learn orthogonal patterns despite being learned without constraints. This result implies that the learnable parameters of dictionaries implicitly capture TPR conditions to ensure accurate TPR operations.

4.3.2 Ablation Study

We investigate the effect of hyper-parameters of D3, specifically N_{code}, D_{code}, and top-k, on performance on the SAR task. Fig. 6(a) shows the effect of D_{code}. We observe that the value of D_{code} significantly affects the performance of D3. Notably, D3 fails to solve the SAR task when D_{code} is set to 8, indicating a need for adequate capacity of D_{code}. Fig. 6(b) shows the effect of varying top-k while holding N_{code} constant, indicating that D3 achieves optimal performance when top-k is set to 2. This result demonstrates the efficacy of the sparse mechanism employed by D3. Fig. 6(c) examines the effect of varying N_{code} while holding top-k constant, showing that D3 generally performs better with larger values of N_{code}.

5 Discussion and Limitations

Motivation. From the perspective of systematic generalization, the decomposition operations in the TPR framework can be viewed as mapping unseen data to TPR components observed during training. Motivated by this, we design a decomposition module based on discrete representations, which maps input data to discrete, learned features facilitating systematic generalization in the decomposition operations of TPR. This design choice differentiates our contribution from AID's competitive attention-based decomposition module. Additionally, each dictionary in D3 is explicitly linked to a specific TPR component, ensuring that each dictionary is responsible solely for generating its corresponding component. The generated components are then utilized in predefined TPR operations of the TPR-based models. This design ensures that each dictionary is trained to specialize in a specific TPR component.

Interpretability. The TPR framework decomposes data at the representation level into distinct symbols, such as *role-filler* pairs for encoding and *unbinding operators* for decoding. This characteristic enhances the interpretability of models because the relationships between *roles* and *unbinding operators* explain which parts of the input the model focuses on to predict the output. However, this interpretability is reliable only when the generated structured representations satisfy the TPR conditions. In this context, D3 enhances the interpretability of models by providing structured representations that more effectively satisfy the TPR conditions than baseline models like FWM and AID. Figs. 9 and 10 demonstrate that the representations generated by D3 better conform to the TPR conditions than those from other baseline models, supporting our claim that D3 contributes to increased interpretability.

D3 Applied to Filler (w/o F and w/ F). In the TPR framework, *roles* and *unbinding operators* must meet specific conditions, such as linear independence among *roles* and high correlation between *roles* and *unbinding operators*, to ensure accurate TPR operations. However, there are no such

https://doi.org/10.52202/079017-0668

requirements for *fillers*, which are features related to downstream tasks. This characteristic affects the performance of D3 depending on whether it is applied to generate the *fillers* (*w/ F*) or not (*w/o F*). In our experiments, the *w/ F* configuration performs well on the sys-bAbI and sort-of-CLEVR tasks with relatively few labels (~200). In contrast, the *w/o F* configuration excels on the SAR and WikiText-103 tasks, which have a larger number of labels (500~). These findings suggest that the *w/o F* configuration may be more effective for large-scale practical tasks. Nevertheless, beyond these experimental results, we do not fully understand the conditions under which each configuration performs better. Consequently, one limitation of D3 is the additional burden of determining the suitable configuration for various tasks when applying it to other domains.

Sparse Key Selection. D3 integrates seamlessly with existing TPR-based models, significantly enhancing their generalization performance across various tasks. However, this integration introduces additional computational overhead to the baseline models. Specifically, the sparse key selection mechanism of D3 has a computational complexity of $\mathcal{O}(N_{\text{code}} \times (D_{\text{query}} + \log k))$ for each TPR component. Therefore, this complexity can become a drawback as the capacity of the dictionaries increases. One potential solution to address this capacity issue is to incorporate product keys into the sparse key selection mechanism of D3, a technique studied in prior discrete key-value architectures [14]. We leave this enhancement for future work.

Scalability. The scalability of D3 is inherently linked to TPR operations of baseline models since the number of dictionaries in the D3 layer aligns with the number of TPR components required for their operations. As TPR operations require increasing components to handle large datasets, our method also requires a proportional increase in dictionaries, resulting in significant computational and memory overhead. As explored in prior work, one potential solution to mitigate this issue is distributing shared dictionaries across multiple heads or layers [14]. However, this approach requires further investigation and experimentation, which we plan to research in future work.

6 Conclusion

In this paper, we tackle the decomposition problem inherent in the TPR framework, which poses a significant challenge for TPR-based models. To address this, we introduce a discrete dictionary-based layer, D3, designed to enhance the decomposition capabilities of TPR-based models. D3 employs the discrete dictionaries to map input data to pre-learned symbolic features within each dictionary, thereby generating structured TPR representations. Our comprehensive experiments demonstrate that D3 significantly enhances the systematic generalization of the TPR-based models with fewer additional parameters. Furthermore, our qualitative analysis verifies that D3 effectively generates structured representations that are satisfactory for the requirements of the TPR framework.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1A2C3011169 & No. 2022R1A5A7026673 & No. RS-2022-00166735 & No. RS-2023-00218987).

References

[1] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] J. A. Fodor and Z. W. Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, 1988.

[3] A. Goyal, A. Lamb, J. Hoffmann, S. Sodhani, S. Levine, Y. Bengio, and B. Schölkopf. Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893*, 2019.

[4] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[5] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[6] K. Greff, S. Van Steenkiste, and J. Schmidhuber. On the binding problem in artificial neural networks. *arXiv preprint arXiv:2012.05208*, 2020.

[7] K. Hsu, W. Dorrell, J. Whittington, J. Wu, and C. Finn. Disentanglement via latent quantization. *Advances in Neural Information Processing Systems*, 36, 2024.

[8] D. Hupkes, V. Dankers, M. Mul, and E. Bruni. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795, 2020.

[9] Y. Jiang, A. Celikyilmaz, P. Smolensky, P. Soulos, S. Rao, H. Palangi, R. Fernandez, C. Smith, M. Bansal, and J. Gao. Enriching transformers with structured tensor-product representations for abstractive summarization. *arXiv preprint arXiv:2106.01317*, 2021.

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

[11] A. Kori, F. Locatello, F. D. S. Ribeiro, F. Toni, and B. Glocker. Grounded object-centric learning. In *The Twelfth International Conference on Learning Representations*, 2023.

[12] B. Lake and M. 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.

[13] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253, 2017.

[14] G. Lample, A. Sablayrolles, M. Ranzato, L. Denoyer, and H. Jégou. Large memory layers with product keys. *Advances in Neural Information Processing Systems*, 32, 2019.

[15] H. Le, T. Tran, and S. Venkatesh. Self-attentive associative memory. In *International Conference on Machine Learning*, pages 5682–5691. PMLR, 2020.

[16] A. Liška, G. Kruszewski, and M. Baroni. Memorize or generalize? searching for a compositional rnn in a haystack. *arXiv preprint arXiv:1802.06467*, 2018.

[17] D. Liu, A. M. Lamb, K. Kawaguchi, A. G. ALIAS PARTH GOYAL, C. Sun, M. C. Mozer, and Y. Bengio. Discrete-valued neural communication. *Advances in Neural Information Processing Systems*, 34:2109–2121, 2021.

[18] F. Locatello, D. Weissenborn, T. Unterthiner, A. Mahendran, G. Heigold, J. Uszkoreit, A. Dosovitskiy, and T. Kipf. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 33:11525–11538, 2020.

[19] S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[20] S. Mittal, S. C. Raparthy, I. Rish, Y. Bengio, and G. Lajoie. Compositional attention: Disentangling search and retrieval. *arXiv preprint arXiv:2110.09419*, 2021.

[21] H. Palangi, P. Smolensky, X. He, and L. Deng. Question-answering with grammatically-interpretable representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[22] T. Park, I. Choi, and M. Lee. Distributed associative memory network with memory refreshing loss. *Neural Networks*, 144:33–48, 2021.

[23] T. Park, I. Choi, and M. Lee. Attention-based iterative decomposition for tensor product representation. In *The Twelfth International Conference on Learning Representations*, 2023.

 https://doi.org/10.52202/079017-0668

[24] J. Rae, J. J. Hunt, I. Danihelka, T. Harley, A. W. Senior, G. Wayne, A. Graves, and T. Lillicrap. Scaling memory-augmented neural networks with sparse reads and writes. In *Advances in Neural Information Processing Systems*, pages 3621–3629, 2016.

[25] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[26] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. *Advances in neural information processing systems*, 30, 2017.

[27] A. Santoro, R. Faulkner, D. Raposo, J. Rae, M. Chrzanowski, T. Weber, D. Wierstra, O. Vinyals, R. Pascanu, and T. Lillicrap. Relational recurrent neural networks. *Advances in neural information processing systems*, 31, 2018.

[28] I. Schlag and J. Schmidhuber. Learning to reason with third order tensor products. *Advances in neural information processing systems*, 31, 2018.

[29] I. Schlag, P. Smolensky, R. Fernandez, N. Jojic, J. Schmidhuber, and J. Gao. Enhancing the transformer with explicit relational encoding for math problem solving. *arXiv preprint arXiv:1910.06611*, 2019.

[30] I. Schlag, T. Munkhdalai, and J. Schmidhuber. Learning associative inference using fast weight memory. *arXiv preprint arXiv:2011.07831*, 2020.

[31] I. Schlag, K. Irie, and J. Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, pages 9355–9366. PMLR, 2021.

[32] Z. Shi, Q. Zhang, and A. Lipani. Stepgame: A new benchmark for robust multi-hop spatial reasoning in texts. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 11321–11329, 2022.

[33] P. Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, 46(1-2):159–216, 1990.

[34] P. Soulos, E. J. Hu, K. McCurdy, Y. Chen, R. Fernandez, P. Smolensky, and J. Gao. Differentiable tree operations promote compositional generalization. In *International Conference on Machine Learning*, pages 32499–32520. PMLR, 2023.

[35] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15 (1):1929–1958, 2014.

[36] S. Sukhbaatar, J. Weston, R. Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

[37] A. Tamkin, M. Taufeeque, and N. D. Goodman. Codebook features: Sparse and discrete interpretability for neural networks. *arXiv preprint arXiv:2310.17230*, 2023.

[38] F. Träuble, A. Goyal, N. Rahaman, M. C. Mozer, K. Kawaguchi, Y. Bengio, and B. Schölkopf. Discrete key-value bottleneck. In *International Conference on Machine Learning*, pages 34431–34455. PMLR, 2023.

[39] A. Van Den Oord, O. Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

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

[41] T. W. Webb, I. Sinha, and J. D. Cohen. Emergent symbols through binding in external memory. *arXiv preprint arXiv:2012.14601*, 2020.

[42] J. Weston, A. Bordes, S. Chopra, A. M. Rush, B. Van Merriënboer, A. Joulin, and T. Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

[43] Y.-F. Wu, M. Lee, and S. Ahn. Structured world modeling via semantic vector quantization. *arXiv preprint arXiv:2402.01203*, 2024.

[44] X. Zhuang, Q. Zhang, K. Ding, Y. Bian, X. Wang, J. Lv, H. Chen, and H. Chen. Learning invariant molecular representation in latent discrete space. *Advances in Neural Information Processing Systems*, 36, 2024.

 https://doi.org/10.52202/079017-0668

Appendix

A Experiment Details

This section provides a detailed description of our experiments on the SAR task, sys-bAbI task, sort-of-CLEVR task, and WikiText-103 task. We followed the experimental settings outlined by AID [23] to assess the decomposition capabilities of `D3`. To ensure stability and reproducibility, we ran all experiments, except for the WikiText-103 task, using 10 different random seeds[3]. For the WikiText-103 task, we experimented with a single seed of 1111. Each experiment was conducted on a single 48GB NVIDIA RTX A6000 GPU and an AMD EPYC 7513 32-Core Processor.

A.1 Systematic Associative Recall task

The SAR task [23] evaluates systematic generalization in memorizing and recalling combinatorial data. It consists of a discovery phase and an inference phase. During the discovery phase, the model receives the combinatorial sequential items, each combining two symbols, $x \in X$ and $y \in Y$ where $X = X_1 \cup X_2 \cup X_3$ and $Y = Y_1 \cup Y_2$. The model is then required to predict an associated y when a specific x is presented. The SAR task uses different combination settings between training and evaluation to target systematic generalization specifically. During the training, the model learns the following combination settings: (1) X_1 and Y_1, (2) X_2 and Y_2, and (3) X_3 and Y. At evaluation, however, the model should generalize unseen combination settings, specifically X_1 and Y_2. In our study, unlike the AID paper [23], we only consider the most challenging setting of the SAR task by excluding the subset X_3.

Each combinatorial item is constructed as follows. First, symbols x and y are sampled from their respective sets X and Y, where $|X_1| = |X_2| = |Y_1| = |Y_2| = 250$. The sampled symbols are mapped into a 50-dimensional space using a word embedding method. These embedding vectors are then concatenated to construct the combinatorial item. For training, 100 randomly generated combinatorial items are sequentially provided to the model during the discovery phase. During the inference phase, the model receives only the x symbols sequentially, with the embedding vector of y set to zero. This task also provides binary flags to indicate the start of each phase. At evaluation, all possible combinations that can be formed in X_1 and Y_2 are tested.

To build the experimental environment for the SAR task, we utilize the open-source implementation[4] from the AID [23]. We train the model using the Adam optimizer with a batch size of 64 and a learning rate of $1e^{-3}$, β_1 of 0.9, and β_2 of 0.98 for training iterations of $30K$. Each experiment took approximately 3 hours per each seed.

A.2 Systematic bAbI task

The sys-bAbI task [23] is a variant of the bAbI task [42] designed to evaluate systematic generalization in text understanding and reasoning. It consists of 20 distinct sub-tasks, each comprising stories, relevant queries, and corresponding answers. The sys-bAbI task requires the models to remember the stories and predict corresponding answers to the queries. Unlike the original bAbI task, the sys-bAbI task evaluates the models with two aspects: (a) in-distribution (*w/o sys diff*) and (b) with the systematic difference (*w/ sys diff*) where each sub-task includes unseen words during training. Therefore, the models should learn task-independent text understanding to solve the sys-bAbI task.

The bAbI dataset includes various versions, such as `en-10k` and `en-valid-10k`. The sys-bAbI task uses the `en-valid-10k` version, which is already divided into training, validation, and test datasets. To create the experimental environment for the sys-bAbI task, we use the open-source implementation[5] provided by the AID.

[3] We used the following seed values: {0, 1111, 2222, 3333, 4444, 5555, 6666, 7777, 8888, 9999}

[4] https://github.com/taewonpark/AID/tree/main/SARtask

[5] https://github.com/taewonpark/AID/tree/main/bAbItask

We use the open-source implementation of the baseline models, TPR-RNN[6] [28] and FWM[7] [30]. Following the experimental settings of baseline models, we use different configurations for each model. We train the TPR-RNN with D3 using an embedding size of 179 and the Adam optimizer with a batch size of 128 and a learning rate of $1e^{-3}$, β_1 of 0.9, and β_2 of 0.99 for 100 training epochs. For FWM with D3, we use an embedding size of 256 and the Adam optimizer with a batch size of 64 and a learning rate of $1e^{-3}$, β_1 of 0.9, and β_2 of 0.98 for training iterations of $60K$. Furthermore, following the AID, we use the reconstruction loss for the bAbI task, introduced in Park et al. [22], in our experiments on the sys-bAbI task. Each experiment took approximately 7 hours per seed for the TPR-RNN with D3 and 8 hours per seed for the FWM with D3.

A.3 Sort-of-CLEVR task

The sort-of-CLEVR task [26] evaluates compositional generalization in visual relational reasoning. It consists of scene images, queries, and corresponding answers. This task requires the models to understand the properties of individual objects (*Unary*) or the relationships between multiple objects (*Binary* or *Ternary*) within visual scene images and predict the correct answers to the queries. Therefore, the model should capture relationships within each object and between objects to solve this task.

Each scene image, with a size of 75×75 pixels, includes 6 distinct objects in 6 different colors (red, blue, green, orange, yellow, or gray) and 2 different shapes (square or circle). This scene image is encoded by a visual encoder. The encoded visual feature is then concatenated with the embedding vector of the query. These concatenated features are provided to the model. Following the experimental settings of the AID [23], we use a single CNN layer with a kernel size of 15 and a stride of 15 for the visual encoder, and an embedding size of 128 for the word embedding method. Also, we use a 4-layered Transformer, where each layer shares its parameters with others, as our baseline model.

To build the experimental environment for the sort of CLEVR task, we utilize the open-source implementation[8] from Mittal et al. [20]. We train the model using the Adam optimizer with a batch size of 64 and a learning rate of $1e^{-4}$ for 100 training epochs. Each experiment took approximately 2.5 hours per each seed.

A.4 WikiText-103 task

The WikiText-103 task [19] is a language modeling dataset consisting of lengthy corpora from Wikipedia. Although the WikiText-103 task does not directly measure the systematic generalization of the models, it is used to evaluate the effectiveness and applicability of D3 on a large-scale task beyond relatively simple tasks.

The WikiText-103 task comprises 28,475 articles for training, 60 for validation, and 60 for testing. Following the experimental settings of Schlag et al. [31], we partition the articles into segments of L words. During training, the gradient is back-propagated only within spans of L words. The performance of the model is evaluated using the measure of perplexity. During evaluation, the model processes an input sequence of L words by sliding a segment over the article with a stride size of 1. Perplexity is then computed based on the last position of each segment, except for the first segment, where every position is taken into account.

To build the experimental environment for the WikiText-103 task, we utilize the open-source implementation[9] from [31]. Following the AID [23], we apply D3 to a 16-layered Linear Transformer at intervals of 4 out of the 16 layers. We train the model using the Adam optimizer with a batch size of 96, an initial learning rate of $2.5e^{-4}$, and a learning rate warmup step of 2,000 for 120 epochs. Each experiment took approximately ~3 days.

[6] https://github.com/APodolskiy/TPR-RNN-Torch
[7] https://github.com/ischlag/Fast-Weight-Memory-public
[8] https://github.com/sarthmit/Compositional-Attention/tree/main/Sort-of-CLEVR
[9] https://github.com/IDSIA/lmtool-fwp

https://doi.org/10.52202/079017-0668

B Hyper-parameter Settings

Table 4: Hyper-parameter settings of the D3.

	SAR task	***sys-bAbI* task**	**Sort-of-CLEVR task**	**WikiText-103 task**
D_{code}	8, 16, <u>32</u>, 64, 128	32, <u>64</u>, 128, 256	128, <u>256</u>	32, <u>64</u>
N_{code}	64			
D_{query}	$D_{\text{code}}/2$			
top-k	8			
p_{dropout}	0.1			

Table 5: Hyper-parameters of TPR-RNN.

	***sys-bAbI* task**
D_{entity} ($D_{\text{component}}$)	90
D_{relation} ($D_{\text{component}}$)	20
$N^{\text{enc}}_{\text{component}}$	5
$N^{\text{dec}}_{\text{component}}$	4

Table 6: Hyper-parameters of FWM.

	SAR task	***sys-bAbI* task**
D_{LSTM}	256	256
D_{FWM} ($D_{\text{component}}$)	32	32
N_{reads}	1	3
$N^{\text{enc}}_{\text{component}}$	3	3
$N^{\text{dec}}_{\text{component}}$	1+N_{reads}	1+N_{reads}

Table 7: Hyper-parameters of Linear Transformer.

	Sort-of-CLEVR task	**WikiText-103 task**
D_{heads} ($D_{\text{component}}$)	64	16
N_{heads}	4	8
$N^{\text{enc}}_{\text{component}}$	2 * N_{heads}	2 * N_{heads}
$N^{\text{dec}}_{\text{component}}$	N_{heads}	N_{heads}

C Additional Experiments

Table 8: The mean word error rate [%] on additional experiments of the *sys-bAbI* task for 10 seeds.

Model	D_{code}	*w/o sys diff* (↓)	*w/ sys diff* (↓)	**Gap** (↓)	# params (↓)
TPR-RNN	-	0.79 ± 0.16	8.74 ± 3.74	7.95	0.14 M
+ AID	-	0.69 ± 0.08	5.61 ± 1.78	4.92	0.32 M
+ D3	32	1.16 ± 0.25	**3.44** ± 1.78	**2.28**	**0.13** M
	64	**0.65** ± 0.25	3.50 ± 2.07	2.85	0.17 M
	128	0.68 ± 0.14	3.94 ± 2.20	3.26	0.26 M
FWM	-	0.79 ± 0.14	2.85 ± 1.61	2.06	**0.73** M
+ AID	-	**0.45** ± 0.16	**1.21** ± 0.66	**0.76**	1.23 M
+ D3 (*w/o F*)	64	0.79 ± 0.30	2.58 ± 1.12	1.79	0.75 M
	128	0.93 ± 0.20	3.82 ± 1.21	2.89	0.82 M
	256	1.04 ± 0.40	3.33 ± 1.21	2.29	0.97 M
+ D3 (*w/ F*)	32	1.20 ± 0.31	7.23 ± 4.33	6.03	0.71 M
	64	0.75 ± 0.17	1.96 ± 0.88	1.21	0.75 M
	128	0.89 ± 0.32	2.48 ± 0.67	1.59	0.84 M
	256	0.75 ± 0.23	3.09 ± 1.83	2.34	1.02 M

D Additional Comparisons

In this section, we expand our comparisons to include a broader range of state-of-the-art methods, as detailed below.

sys-bAbI task. We compare D3 to state-of-the-art methods (DAM [22] and STM [15]) on the original bAbI task. Table 9 shows that existing memory networks struggle with the sys-bAbI task, highlighting the efficacy of D3 compared to these state-of-the-art memory networks.

Table 9: The mean word error rate [%] on additional comparison of the sys-bAbI task for 10 seeds.

Model	*w/o sys diff* (↓)	*w/ sys diff* (↓)	**Gap** (↓)
DAM	0.48 ± 0.20	5.25 ± 1.64	4.77
STM	0.49 ± 0.16	4.79 ± 1.53	3.70
TPR-RNN	0.79 ± 0.16	8.74 ± 3.74	7.95
+ AID	0.69 ± 0.08	5.61 ± 1.78	4.92
+ D3	**0.65** ± 0.25	**3.50** ± 2.07	**2.85**
FWM	0.79 ± 0.14	2.85 ± 1.61	2.06
+ AID	**0.45** ± 0.16	**1.21** ± 0.66	**0.76**
+ D3 (*w/o F*)	0.79 ± 0.30	2.58 ± 1.12	1.79
+ D3 (*w/ F*)	0.75 ± 0.17	1.96 ± 0.88	1.21

Sort-of-CLEVR task. We compare D3 to vanilla Transformer [40] and Compositional Transformer [20], designed to enhance the systematic generalization capabilities of multi-head self-attention methods. Table 10 shows that the Linear Transformer significantly degrades systematic generalization performance compared to the vanilla Transformer and the Compositional Transformer. While D3 improves the performance of the Linear Transformer from a TPR perspective, it still shows limited performance in reasoning the relationships between multiple objects (*Binary* and *Ternary*) compared to the vanilla Transformer and Compositional Transformer.

WikiText-103 task. We compared D3 to the Delta Network [31], which introduced a delta updating rule instead of the additive outer product-based updating rule in the Linear Transformer. Table 11

https://doi.org/10.52202/079017-0668

Table 10: The mean accuracy [%] on additional comparison of the sort-of-CLEVR task for 10 seeds.

Model	D_{code}	*Unary* (↑)	*Binary* (↑)	*Ternary* (↑)
Transformer	-	97.4 ± 3.5	84.3 ± 4.3	62.7 ± 3.9
Compositional Transformer	-	98.9 ± 0.2	**88.4** ± 1.4	66.5 ± 1.9
Linear Transformer	-	69.3 ± 14.8	75.5 ± 1.3	56.4 ± 4.3
+ AID	-	98.9 ± 0.2	78.6 ± 0.3	63.7 ± 1.2
+ D3 (*w/o F*)	128	73.9 ± 16.5	77.2 ± 2.2	57.3 ± 4.6
	256	73.7 ± 16.5	77.8 ± 2.5	57.9 ± 5.8
+ D3 (*w/ F*)	128	98.9 ± 0.2	79.5 ± 0.8	63.1 ± 1.9
	256	**99.0** ± 0.3	82.1 ± 2.4	**68.8** ± 1.2

indicates that although D3 improves the performance of the Linear Transformer in language modeling tasks, the choice of updating rules has a more substantial impact on performance for tasks involving the comprehension of lengthy corpora than the decomposition operation.

Table 11: Perplexity on additional comparison of the WikiText-103 task.

Model	D_{code}	*Valid* (↓)	*Test* (↓)
Delta Network	-	**35.640**	**36.659**
Linear Transformer	-	36.473	37.533
+ AID	-	36.159	37.151
+ D3 (*w/o F*)	32	36.061	37.220
	64	35.975	37.009
+ D3 (*w/ F*)	32	36.630	37.620
	64	36.220	37.128

E Additional Ablation Study

In this section, we extend our ablation studies to investigate the effects of varying the number of keys in the codebook and the impact of removing either the residual connection or the codebook from the D3 layer.

The Effect of Varying the Number of Codebook Keys. Fig. 7 shows that even with a significantly reduced number of keys, the model with D3 maintains high accuracy on the SAR task. This observation prompts the question of how consistent performance is achieved despite the reduction in codebook size. To explore this further, we examine the impact of removing the codebook or the residual connection within the D3 layer on the SAR and sys-bAbI tasks. Specifically, removing the codebook means that the `components` are generated solely by the shared feed-forward networks ($\text{layer}_{\text{residual}}$ and $\text{layer}_{\text{final}}$) while removing the residual connection implies that the `components` are derived solely from the codebook values.

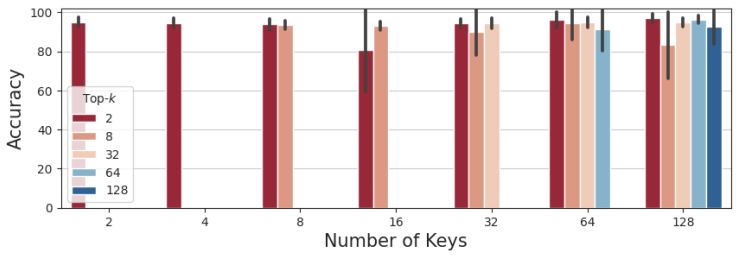

Figure 7: The mean accuracy on the SAR task for 10 seeds in the ablation study for the effect of varying N_{code} from 2 to 128 with top-k constant.

The Effect of Residual Connection. Fig. 8 shows that without the residual connection, the generalization performance of D3 dramatically degrades. This result indicates that the residual connection is crucial for effectively training the D3 layer.

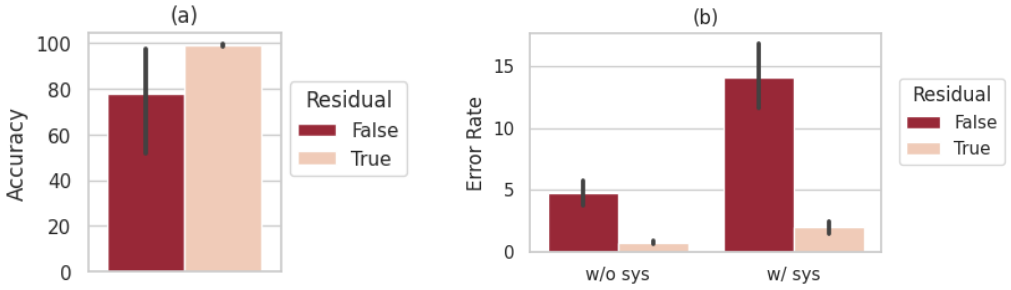

Figure 8: Ablation study for the effect of the residual connection on (a) the SAR task and (b) the sys-bAbI task for 10 seeds.

The Effect of Codebook. Table 12 shows that even without the codebook ("*w/o codebook*"), the D3 layer improves the generalization performance of the baseline model on the SAR task. This result indicates that the shared feed-forward networks significantly contribute to performance enhancement, which may explain why the model maintains robust performance even with fewer keys.

However, it is important to note that without the codebook, the D3 layer does not achieve near-perfect accuracy on the SAR task (as shown in Table 12) and fails to significantly enhance the systematic generalization of the baseline model on the sys-bAbI task (as shown in Table 13). These results demonstrate that the codebook plays a crucial role in enhancing the model's overall performance and generalization capabilities, especially in tasks requiring systematic generalization.

Furthermore, we experiment with $N_{\text{code}} = 1$ on the SAR task, where the codebook may act as a bias term. The results in Table 12) show that using a single codebook element leads to degraded generalization performance compared to the "*w/o codebook*" configuration, indicating that multiple codebook elements are essential for achieving optimal results.

Table 12: Ablation study for the effect of the codebook on the SAR task for 10 seeds.

Model	D_{code}	N_{code}	top-k	**Accuracy** (↑)
FWM	-	-	-	44.90 ± 31.5
+ D3	32	4	2	87.38 ± 11.10
+ D3		64	8	**99.27** ± 0.88
+ D3 (*w/o codebook*)		-	-	89.02 ± 4.56
+ D3	64	1	1	89.10 ± 7.99
+ D3		4	2	**94.47** ± 2.35
+ D3		64	8	94.29 ± 8.06
+ D3 (*w/o codebook*)		-	-	91.65 ± 3.66

Table 13: Ablation study for the effect of the codebook on the sys-bAbI task for 10 seeds.

Model	*w/o sys diff* (↓)	*w/ sys diff* (↓)	**Gap** (↓)
FWM	0.79 ± 0.14	2.85 ± 1.61	2.06
+ D3	**0.75** ± 0.17	**1.96** ± 0.88	**1.21**
+ D3 (*w/o codebook*)	1.19 ± 0.41	3.55± 1.04	2.36

Discussion. Our ablation study on the codebook in the SAR task (Table 12) indicates that the shared residual networks within the D3 layer significantly enhance generalization performance. However, the results from the sys-bAbI task (Table 13) suggest that while these networks improve performance, they alone struggle to generalize more structured data.

https://doi.org/10.52202/079017-0668

The ablation studies in Tables 12 and 13 demonstrate that incorporating the codebook mechanism leads to nearly 100% accuracy on the SAR task and significantly improves the systematic generalization of models on the sys-bAbI task. However, as shown in the ablation study on the residual connection (Fig. 8), the codebook alone does not achieve the same level of generalization and exhibits instability within the D3 layer.

In conclusion, our experimental results indicate that the combination of the codebook and the shared residual networks within the D3 layer is crucial for enhancing systematic generalization performance and stability. By integrating these two components, our D3 layer significantly improves the systematic generalization capabilities of TPR-based models.

F Additional Qualitative Analysis

F.1 Comparison to Baselines

We conduct an orthogonal analysis for the baseline models (FWM and AID) similar to the analysis presented in Section 4.3.1. Figs. 9 and 10 indicate that the D3 model generates more structured and orthogonal representations than the baseline models, FWM and AID, demonstrating its effectiveness.

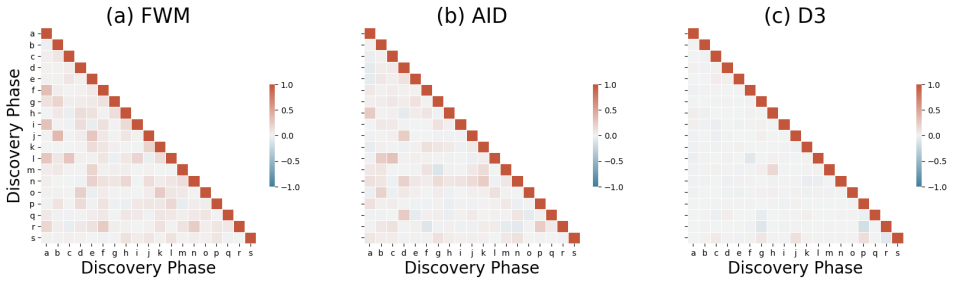

Figure 9: The heatmap displays the cosine similarity between the *roles* during the discovery phase for the SAR task.

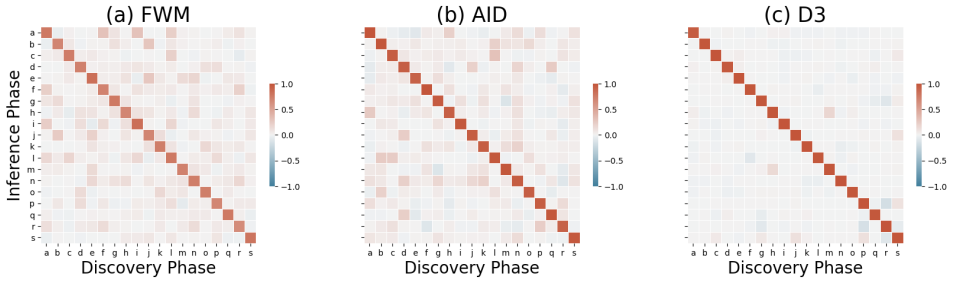

Figure 10: The heatmap displays the cosine similarity between the *roles* (**x-axis**) during the discovery phase and the *unbinding operators* (**y-axis**) during the inference phase for the SAR task.

F.2 Qualitative Analysis for Different Seeds

Additionally, we present the results of the qualitative analysis for different seeds in the SAR task.

F.2.1 N_{code}: 64, D_{code}: 32, top-k: 8, seed: 3333

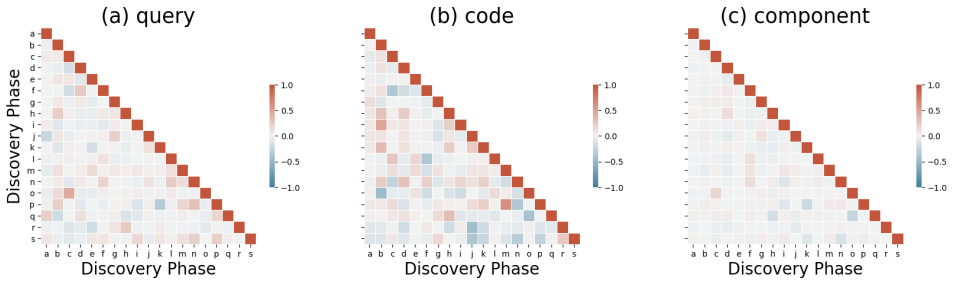

Figure 11: The heatmap displays the cosine similarity between the generated representations during the discovery phase for the SAR task. We explore the similarity across different types of representations: (a) `queries` of *roles*, (b) `codes` of *roles*, and (c) the *roles* themselves.

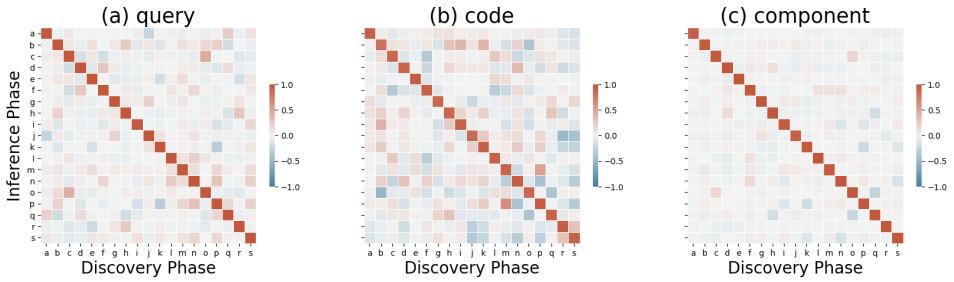

Figure 12: The heatmap displays the cosine similarity between the generated representations during the discovery phase (represented on the **x-axis**) and the inference phase (represented on the **y-axis**) for the SAR task. We explore the similarity across different types of representations: (a) `queries` of *roles* and *unbinding operators*, (b) `codes` of *roles* and *unbinding operators*, and (c) the *roles* and *unbinding operators* themselves.

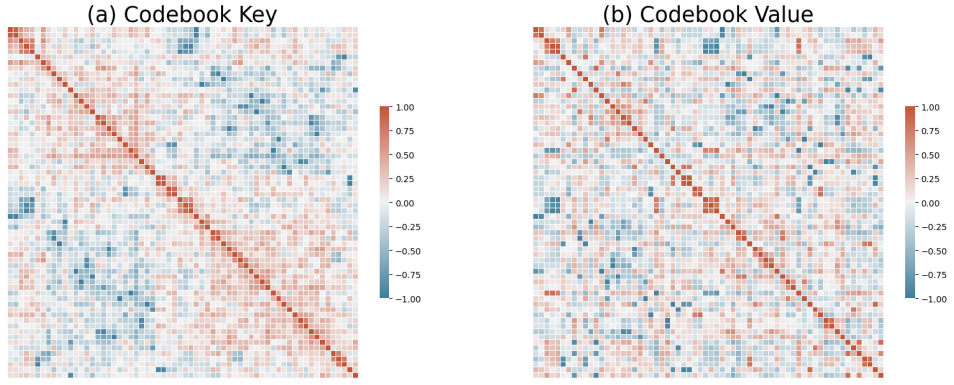

Figure 13: The heatmap visualizes the cosine similarity of the learned codebook features for the SAR task. There are two parts to each heatmap: (a) the similarity among codebook keys, denoted as $\{\mathsf{k}_i\}_{i=1}^{N\text{code}}$, and (b) the similarity among codebook values, denoted as $\{\mathsf{v}_i\}_{i=1}^{N\text{code}}$. For better visualization, the heatmap values are reordered to reflect the cluster of similar codebook keys.

https://doi.org/10.52202/079017-0668

F.2.2 N_{code}: 64, D_{code}: 32, top-k: 8, seed: 4444

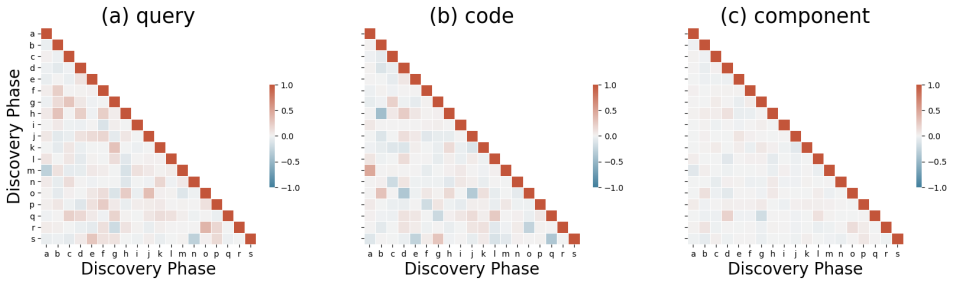

Figure 14: The heatmap displays the cosine similarity between the generated representations during the discovery phase for the SAR task. We explore the similarity across different types of representations: (a) `queries` of *roles*, (b) `codes` of *roles*, and (c) the *roles* themselves.

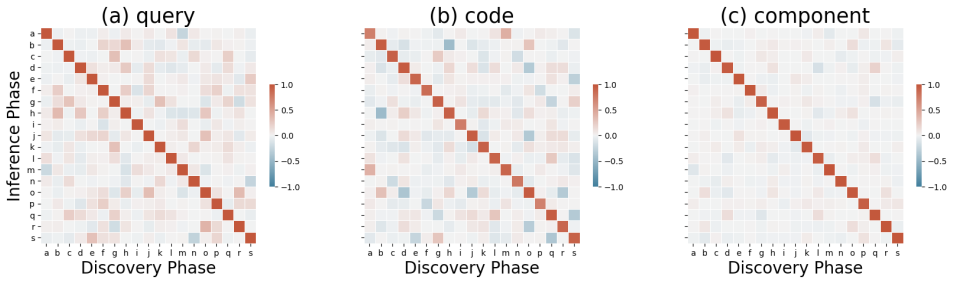

Figure 15: The heatmap displays the cosine similarity between the generated representations during the discovery phase (represented on the **x-axis**) and the inference phase (represented on the **y-axis**) for the SAR task. We explore the similarity across different types of representations: (a) `queries` of *roles* and *unbinding operators*, (b) `codes` of *roles* and *unbinding operators*, and (c) the *roles* and *unbinding operators* themselves.

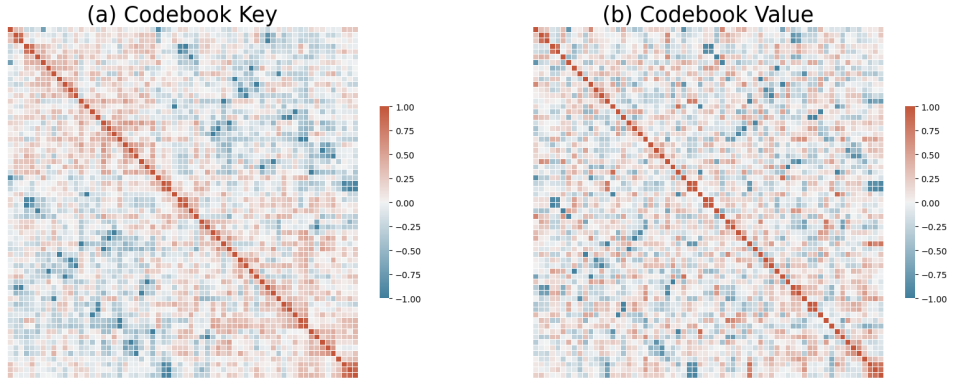

Figure 16: The heatmap visualizes the cosine similarity of the learned codebook features for the SAR task. There are two parts to each heatmap: (a) the similarity among codebook keys, denoted as $\{\mathsf{k}_i\}_{i=1}^{N\text{code}}$, and (b) the similarity among codebook values, denoted as $\{\mathsf{v}_i\}_{i=1}^{N\text{code}}$. For better visualization, the heatmap values are reordered to reflect the cluster of similar codebook keys.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: This paper includes the paper's contributions and scope in the abstract and introduction, as follows. This paper tackles the decomposition problem inherent in the TPR-based approaches. To address this, this paper proposes a discrete dictionary-based decomposition (D3) layer designed to enhance the decomposition capabilities of the TPR-based models.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: This paper discusses the limitations of the work performed by the authors in Section 5, as follows. The model introduced in this paper requires additional computational overhead and configuration search when the proposed model is integrated into the existing baseline models.

   Guidelines:

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

https://doi.org/10.52202/079017-0668

3. **Theory Assumptions and Proofs**

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

   Answer: [NA]

   Justification: This paper does not include theoretical results.

   Guidelines:

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

4. **Experimental Result Reproducibility**

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

   Answer: [Yes]

   Justification: This paper discloses all the information needed to reproduce the experimental results. This paper explains the mechanism of the proposed model and how it is applied to existing baseline models in Section 3 and presents the experiment details and hyperparameter settings in Appendices A and B.

   Guidelines:

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

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

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

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

   Answer: [Yes]

   Justification: This paper provides supplementary materials to reproduce all experimental results of the proposed method, including source codes about our model implementation, data processing, scripts for execution, etc.

   Guidelines:

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

6. **Experimental Setting/Details**

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

   Answer: [Yes]

   Justification: This paper presents our experiment details and hyper-parameter settings in Appendices A and B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

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

   Answer: [Yes]

   Justification: This paper reports the mean and standard deviation values in the experimental results conducted using fixed 10 different random seeds.

https://doi.org/10.52202/079017-0668

Guidelines:

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

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: This paper provides the computer resources used in our experiments and the time it took to learn each task in Appendix A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: The research conducted in this paper conforms to the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader Impacts**

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

Answer: [NA]

Justification: The authors do not foresee a negative societal impact on the work presented in this paper beyond the general effects of ML advancements.

Guidelines:

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

11. **Safeguards**

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

Answer: [NA]

Justification: This paper does not have a high risk for misuse.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: This paper cites the original paper that produced the code package or dataset, and includes URLs in Appendix A.

Guidelines:

- The answer NA means that the paper does not use existing assets.

https://doi.org/10.52202/079017-0668

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

13. **New Assets**

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

    Answer: [Yes]

    Justification: This paper provides supplementary materials with source code, license, and README.md files. The README.md files cite the code packages utilized in this paper and provide all the instructions to reproduce the experimental results.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

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

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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

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

    Answer: [NA]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

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

https://doi.org/10.52202/079017-0668