Cognitive Computation
https://doi.org/10.1007/s12559-023-10154-3

An Interpretable Neuro-symbolic Model for Raven's Progressive Matrices Reasoning

Shukuo Zhao[1] · **Hongzhi You**[2] · **Ru-Yuan Zhang**[3,4] · **Bailu Si**[1] · **Zonglei Zhen**[5] · **Xiaohong Wan**[5,6] · **Da-Hui Wang**[1,5,7]

Received: 1 October 2022 / Accepted: 5 May 2023

Abstract
Raven's Progressive Matrices (RPM) have been widely used as standard intelligence tests for human participants. Humans solve RPM problems in a hierarchical manner, perceiving conceptual features at different levels and inferring the latent rules governing the matrix using cognitive maps. Although the latest AI algorithms can surpass human performance, little effort has been made to build a model that solves RPM problems in a human-like hierarchical manner. We built a human-like hierarchical neuro-symbolic model to solve RPM problems. The proposed model consists of a semantic-VAE (sVAE) perceptual module and a cognitive map reasoning back-end (CMRB). The supervised sVAE extracts the hierarchical visual features of RPMs by perceiving the structural organization of RPMs through a convolutional neural network and disentangles objects into semantically understandable features. Based on these semantic features, the CMRB predicts the semantic features of objects in the missing field using cognitive maps generated by supervised learning or manually designed. The answer image was generated by sVAE using the semantic features predicted by CMRB. The proposed model achieved state-of-the-art performance on three benchmarks datasets—RAVEN, I-RAVEN, and RAVEN-fair—generalizes well to RPMs containing objects with untrained feature dimensions, mimics human cognitive processes when solving RPM problems, achieves interpretability of their hierarchical processes, and can also be applied to some real-world situations that require abstract visual reasoning.

Keywords Raven's Progressive Matrices · Semantic-variable autoencoders · Cognitive maps · Visual abstract reasoning · Artificial intelligence

✉ Da-Hui Wang
wangdh@bnu.edu.cn

1 School of Systems Science, Beijing Normal University, 19th Xinjiekouwai Street, Haidian, Beijing 100875, China

2 School of Life Science and Technology, University of Electronic Science and Technology of China, No. 2006, Xiyuan Avenue, Sichuan, Chengdu 611731, China

3 Institute of Psychology, Shanghai Jiao Tong University, No. 80, Dongchuan Street, Minhang, Shanghai 200030, China

4 Behavioral Science and Shanghai Mental Health Center, Shanghai Jiao Tong University, No. 80, Dongchuan Street, Minhang, Shanghai 200030, China

5 State Key Laboratory of Cognitive Neuroscience and Learning, Beijing Normal University, 19th Xinjiekouwai Street, Haidian, Beijing 100875, China

6 IDG/McGovern Institute for Brain Research, Beijing Normal University, 19th Xinjiekouwai Street, Haidian, Beijing 100875, China

7 Beijing Key Laboratory of Brain Imaging and Connectomics, Beijing Normal University, Beijing 100875, China

Introduction

The ability to generalize is critical for humans to reason flexibly and quickly [1, 2]. A commonly studied neural mechanism for generalization is how humans build cognitive maps in the entorhinal cortex using grid cells that factorize representations of hippocampal place cells [2, 4] for spatial generalization [1, 3]. Researchers have found that similar principles apply in nonspatial domains [2]. For example, in the semantic domain, when faced with the simple problem "woman-man, queen-?", people first factorize the "queen" into gender and status representations, and then generalize in the gender dimension [2, 5]. Because this (semantic) cognitive map reasoning process is the foundation of abstract reasoning, we built a general model to mimic the human reasoning process when solving RAVEN problems.

Raven's Progressive Matrices (RPM) [6] have been widely used as standard intelligence tests for human participants [7, 8]. Each test takes the form of a 3×3 matrix

Published online: 25 May 2023

Table 1 Average accuracy of different algorithms

Methods	RAVEN	I-RAVEN	RAVEN-fair	Acc
PrAE [21]	65.0	77.0	88.3*	76.8
ALANS [22]	79.6	65.4	-	72.5
Rel-base [19]	91.7	92.1*	92.8*	92.2
MRNet [17]	96.6	85.1*	88.4	90.0
SCL [20]	91.6	95	91.7*	92.8
SRAN [16]	55.7*	60.8	71.3*	62.6
sVAE-CMRB	**97.7**	**98.2**	**98.8**	**98.2**
designed CM	**97.7**	**98.1**	**98.9**	**98.2**

[a]An asterisk (*) indicates that we tested the model by us; other unmarked values were taken from the literature

with the bottom right panel missing. Participants are asked to identify the underlying rules that govern the progression of the matrix and to apply the rules to complete the missing entry of the matrix. Human participants have relied on conceptual semantic abstraction and analogical reasoning to solve RPM problems [9, 10]. Semantic abstraction can occur in Gestalt hierarchical perception [11, 12], and analogical reasoning can easily generalize learned knowledge to new situations [13, 14].

Many efforts have been made to solve RPM problems using artificial intelligence (AI) algorithms [10]. A digital dataset "RAVEN" was proposed for AI algorithms and then became a benchmark problem for abstract spatiotemporal reasoning [15]. Seven configurations (Center, 2*2 Grid, 3*3 Grid, Left–Right, Up-Down, Out-InCenter, and Out-InGrid; see Figs. 4 or 5 in [15]) of 3×3 matrix problems that have similar progressions in RPM were automatically generated by the algorithm in the RAVEN dataset; objects were arranged differently according to the configurations. AI algorithms must identify the rules in the rows of the matrices and complete the 9th cell of the matrices in all configurations [15]. The subsequent "I-Raven" and "Raven-FAIR" datasets improved the algorithm's ability to generate false candidates for the missing panel in the RAVEN dataset [16, 17].

AI algorithms can even surpass human performance with adequate training on these datasets (see Tables 1 and 2) [10, 18]. Based on the supervised learning method, the Rel-base and Rel-air models used convolutional neural networks (CNNs) to identify the attributes of matrices and to find the relationships between panels [19]. The SCL used scattering transformation in shared attribute and relation modules to acquire attributes and rules [20]. The MRNet used a multi-scale feature extraction module and a relational network to determine the relevance between different panels in different rows [17]. The PrAE model used neural networks and a symbolic reasoning backend to identify visual features and find the most plausible attribute values [21]. The ALANS learner, which is another type of neurosymbolic model, used algebraic methods to represent the relationships of attributes extracted from encoders [22].

Although the latest AI algorithms can surpass human performance, little effort has been made to build a model that solves RPM problems in a human-like hierarchical manner, translating the physical attributes of RPM into semantic conceptual features and inferring the latent rules governing the modification of matrices using cognitive maps or mental models. In this study, we developed a model to solve RPM problems in a human-like manner. The proposed model showed clear progress in describing how and why answers are generated and predicted what the answer image looks like for the missing cell with no candidate answers, as predicted by Gestalt psychology [23]. The proposed model may report incorrect answers, but errors can be human-like and semantically understandable.

Table 2 Average accuracy of VAE-related methods on RAVEN

VAE methods	I-RAVEN([a])
LoGe [27]	62.9
C_0 [29]	60.8[a]
Ours	98.2

[a]C_0 was tested on RAVEN-fair

Related Works

Psychological Model for Solving RPM Problems

Psychologists have developed models to describe the psychological stages and cognitive skills involved in the RPM problem-solving process. In one psychology-based model, five psychological steps were required to solve RPM problems. Hierarchical perception (the first step) determined which parts to consider as a whole and perceived each segment separately (i.e., as edges and as a whole). In the second step, the model compared and found consistencies and differences in different hierarchies across panels. If the comparison failed, the model reorganized the perception in the first step. In the remaining three steps, the model reasoned about the critical consistencies and differences to find the answers [11]. Psychology-based models were less complex but more semantically explicable than performance-based AI algorithms.

Variational Autoencoders

Autoencoders (AEs) are neural networks that consist of three parts: encoder, bottleneck, and decoder [24, 25]. The encoder part takes the input data and compresses the data into a low-dimensional latent space represented by the bottleneck, and then, the decoder recovers the data from the compressed latent representation. Variational AE (VAE) has the same structure as AE but views the recovery process as an inference problem. With regularized latent dimensions, VAE can generate new data points [26]. VAE has been used to solve Raven-related problems for feature disentanglement [27, 28] and answer image generation [27, 29] or Gestalt image completion [23]. Answer generation without candidates is more complex and human-like than other AI algorithms. Although VAE algorithms can sometimes generate reasonable answers and images, overall performance has not been good [23, 27, 29].

The hyperparameter β was introduced in β-VAE to balance the reconstruction and regularization losses [30], which enhanced the disentanglement of the latent variables and improved the performance. Although β-VAE was efficient for data disentanglement, we do not know the semantic meaning of the latent dimensions produced by β-VAE, the conceptual features they encode, or how the latent dimension affects image regeneration. In this paper, we introduce a semantic version of VAE (sVAE) that can decompose figures into semantic dimensions through supervised learning (see Fig. 2). sVAE can generate learned images and novel images according to semantic descriptions by changing the semantic features in the bottleneck (morphing images).

Cognitive Maps

Tolman first proposed cognitive maps as a systematic organization of knowledge that spans all domains of behavior [1]. Later, the hippocampus was recognized as the neural substrate of cognitive maps [3]. Computationally, cognitive maps represent generalizable action transitions between different state spaces that can be modeled by reinforcement learning algorithms such as successor representation [31, 32], linearly solvable Markov decision processes [33], cognitive graph models such as the clone-structured cognitive graph model (CSCG) [34], and neural-inspired grid cell models that perform path integration [1, 2]. Recently, a unified framework called the Tolman-Eichenbaum machine provided a model that shows the role of the hippocampus in spatial and nonspatial generalization and the principles underlying many entorhinal and hippocampal cell types [4]. The cognitive maps proposed by Whittington were independent of specific locations and could be generalized across different maps. When entering a new environment, the encoding of spatial knowledge by grid cells binds with the encoding of places by place cells to form a complete cognitive map from which we can infer the relationship between any two nodes in the map. This process can be generalized to nonspatially structured relationships such as family trees [4]. RPM problems contain similar general "feature maps" that can be generalized across different RPM problems. Humans can abstract these feature maps to form cognitive maps and use them to reason across contexts.

Purpose of This Study

We build a high-performance model with human-like perceptual and reasoning capabilities by incorporating perceptual and semantic reasoning technologies VAE and cognitive maps to mimic human behavioral reasoning pipelines involving hierarchical perception, comparison, and critical difference finding, as in psychological studies [11], and to realize human neural mechanisms of factorization (disentanglement) and cognitive map building. First, the model can generate human-like object perceptions and perceptual expectations by disentangling objects into conceptual semantic latent dimensions and generating integrated representations and expectations for objects beyond the enumeration of its part. Second, the model can reason abstractly at the conceptual or semantic level rather than at the pixel level. Third, the model can respond accurately by predicting the attributes of the missing field and generating the possible image of the answer without the candidates for the missing field of the matrices.

Methods: the Neuro-symbolic sVAE-CMRB Model

Model Overview

Experiments were performed on NVIDIA GeForce GPU platforms (driver version, 510.39.01; CUDA version, 11.6; or driver version, 460.80; CUDA version, 11.2) and a CPU (Windows 10 64-bit Intel(R) Core(TM) i7-8700 CPU @ 3.20 GHz (12 CPUs)). The proposed model (see Fig. 1) solved the problems in two human-like stages. In the first stage, hierarchical perception extracted the structural organization of the problem, the features of each panel, and the attributes of the objects in each panel. In the second stage, the model learned the rules governing the progression of the matrix and realized abstract reasoning. In the test condition (second stage), the model applied the learned rules to generate answer images with predictions or to compare attribute predictions with selections. The model consists of three modules, each of which is shown below.

Structural Organization Perception FCNN-Network

Hierarchical perception is a coarse-to-fine perceptual process in psychology. The configuration or arrangement of the items of RAVEN problem is the global or coarse information, which often be processed at first in our brain according to global first theory by Lin Chen [35]. In the model, FCNN is the first perceptual level and simplifies by assuming that human's perception at this level is determined by the type of problem, implying that the same type of problem is

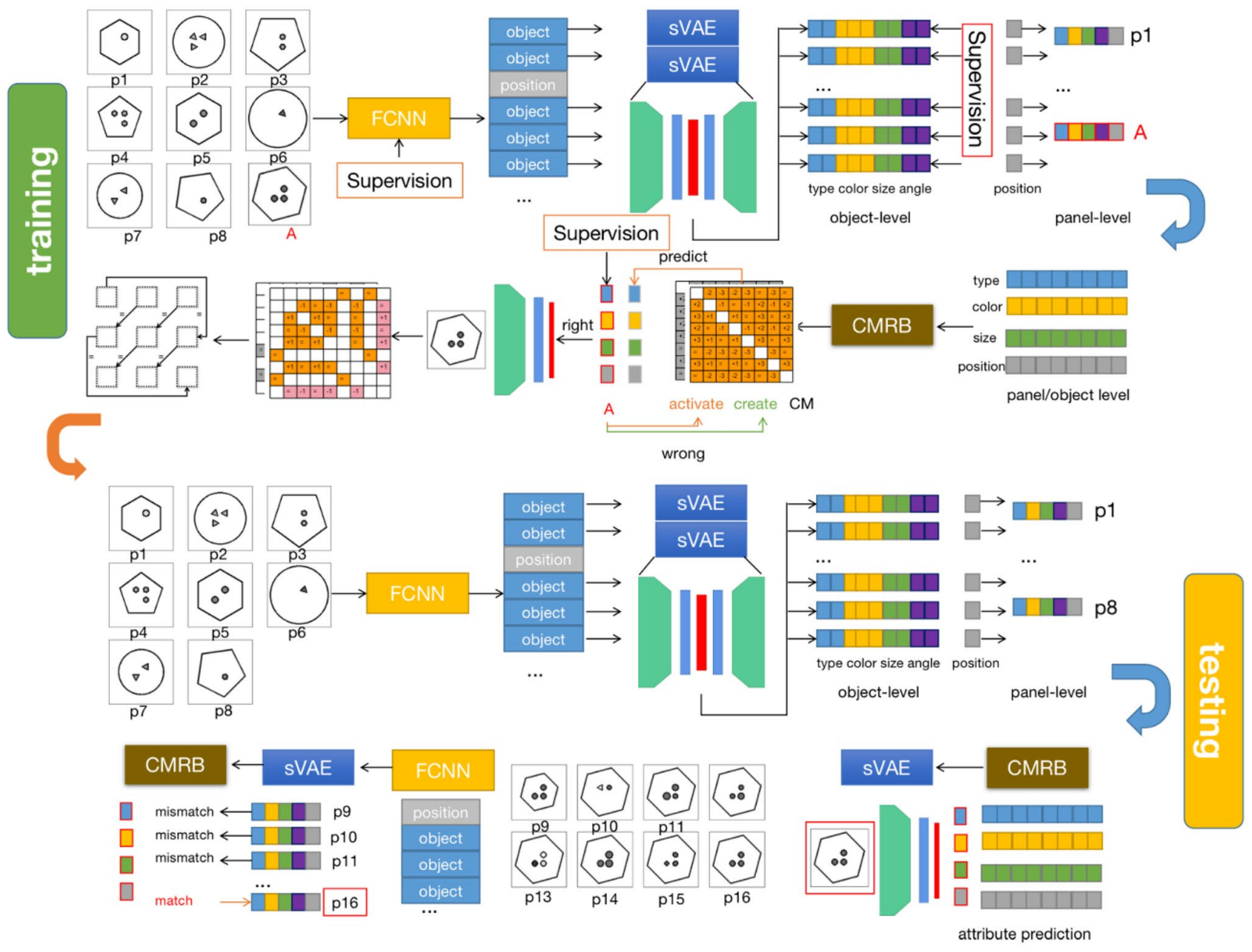

Fig. 1 Overview of the proposed model framework. Training condition (top): Given an RPM problem, the FCNN first processes the eight problem panels and the answer panel; recognizes the structural organization of the problem; extracts object position information (in gray); and segments the image into objects (in blue). The sVAE then processes the object segments and disentangles them into semantic features. CMRB analyzes position information (in gray) and semantic features at the panel or object level (including type (in blue), color (in yellow), size (in green), and angle (in purple) information) for the eight problem panels and the answer panel, abstracts relationships, generates predictions, and learns cognitive maps. The supervisions are applied to the model as indicated. Testing condition (bottom): In the test condition, without providing the answer panels to the model, the trained FCNN and sVAE modules extract detailed semantic features of objects and positions in the first eight panels, and the CMRB predicts features in the missing panels. sVAE further constructs possible answer images based on these predictions. To make a selection, FCNN and sVAE process panels 8–16. CMRB compares the semantic features of panels 8–16 with the generated predictions and selects the best matching panel as the answer. The outputs are circled in red squares

perceived roughly in the same way. A convolutional neural network with four layers (16, 32, 64, 128 latent dimensions, kernel size 5 * 5, step 1, padding 1) and a feedforward layer (16384 nodes) was jointly trained (FCNN) to recognize the type of RPM problem. This structural organization information was critical because the framework of semantic description and cognitive maps was different for different problem types. The FCNN network took the image of the first panel (size: 160 * 160) of the problem as input and output one-hot encoded structural organizations corresponding to one of the seven types of RPM problems (Center, 0; 2*2 grid, 1; 3*3 grid, 2; Left–Right, 3; Up-Down, 4; Out-InCenter, 5; and OutInGrid, 6). We used 959 problems per configuration to train the FCNN module (cross-validation). The FCNN learned quickly, reaching 100% accuracy after seeing 959 * 7(=6713) problems in 50 training epochs. According to the resulting structural organization of the RPM problem, the image of each panel can be segmented into individual objects whose semantic features can be extracted by the sVAE framework described in the following subsection. The structural organization information also determined the maximum number of object-level features to consider in

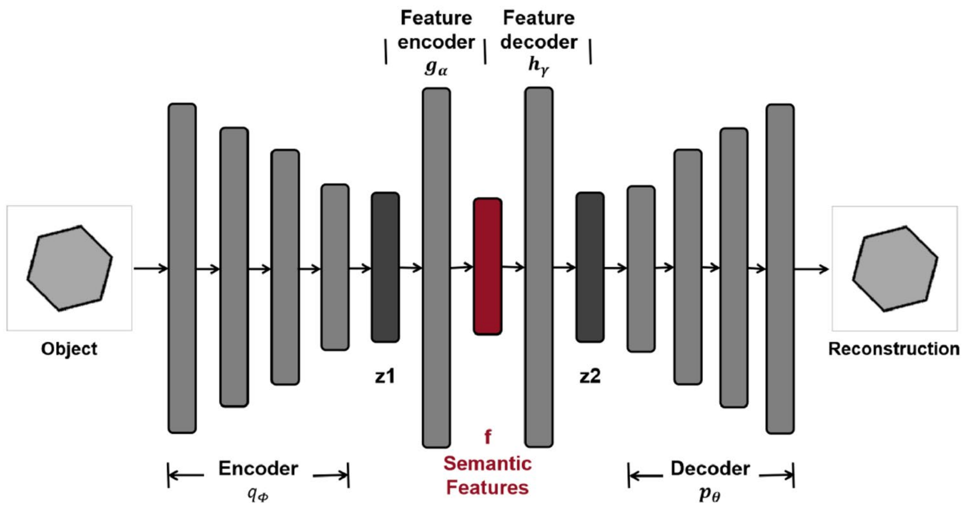

Fig. 2 Schematic of the semantic VAE module. Each bar represents the output of one layer of the algorithm. The encoder of sVAE (4 layers) decomposes the images of objects into low-dimensional latent features (z1). The latent features (z1) are fully connected to the semantic layer (f). The semantic feature layer (f) has 29 dimensions, representing the semantic features of objects (5 types, 10 colors, 6 sizes, 8 angles). The semantic feature layer fully projects onto latent features (z2). The decoder (4 layers) regenerates the image according to the latent features (z2)

a problem in CMRB (different configurations had different numbers of objects).

Semantic VAE Feature Extraction Module

The encoder of sVAE sampled the data (X) and generated a joint distribution of X and its latent variables Z1. Z1 was input to the feature encoder to produce human understandable semantic features (f) of the sample data (X). The latter was further input to the decoder of the semantic feature layer to produce a new set of latent variables Z2 and the decoder reconstructs X from Z2. The reconstruction loss is the difference between the data X and the reconstructed X. VAE assumes that Z follows a given distribution. Any violation of this distribution leads to a regularization loss. The relative weight between the regularization loss and the reconstruction loss is modulated by the hyperparameter β in β-VAE [30]. The β-VAE is efficient at disentangling data, but the latent variables in the β-VAE had no explicit semantic or conceptual meanings for the input data; thus, it was difficult for us to understand the reasoning process of the model. To overcome this shortcoming, we introduced semantic feature layers into VAE, i.e., projected the latent variables (Z1) onto a feature encoder and then onto a semantic feature layer (f) whose activity is semantically understandable in human-defined dimensions (i.e., successively one-hot encoded shape (five dimensions), color (ten dimensions), size (six dimensions), and angle (eight dimensions)). The 29 semantic nodes in the semantic layer correspond to the 29 human-defined dimensions listed above, and the activation of each node represents how well the input image fits within that dimension. Taking a node as an example, if the image being semantically analyzed fits precisely into that dimension, the activation of that node converges to 1. If it does not, it converges to 0. The supervision to the semantic layer is the same as the one-hot encoded label of the input data obtained from the corresponding problem XML file (see Appendix 3 for a more detailed description). The semantic feature layer was then projected onto feature decoder layer and then onto another latent variable (Z2) similar to Z1. With this modification, the model's decoder continued to use the latent variable (Z2) to reconstruct the input image as a β-VAE (see Fig. 2). Therefore, the loss of the sVAE model can be decomposed into four parts: the object reconstruction loss, the latent variable reconstruction loss, the supervised loss (difference between the semantic features and the labels), and the regularization loss (divergence between the distribution of the latent variables and the assumed distribution), as in Eq. 1:

$$L = E_{q\phi}(z1|x)[\log\ p_\theta(x|z_2)] + E_{g\alpha}(f|z_1)[\log h_\gamma(z_2|f)] + Smooth\ l_1(f,\ labels) - \beta \times D_{KL}(q_\phi(z_1|x)|p(z_1)) \quad (1)$$

where β is the weight of the regularization loss relative to other losses. In the proposed model, β was set to 10, as in other typical β-VAE models. ϕ, θ, α, and γ were the parameters for the encoder (4-layer CNN encoder with 40, 64, 128, and 256 latent dimensions; a kernel size of 3 * 3; a stride of 2; and a padding of 1), the decoder (4-layer transpose CNN with 256, 128, 64, and 40 latent dimensions; a kernel size of 3 * 3; a stride of 2; a padding of 1; and an outpadding of 1), the feature encoder (4096 hidden units), and the feature decoder (4096 hidden units), respectively. We adjusted the hyperparameters to minimize the loss. Smooth l1 means smooth l1 loss, as in Eq. 2:

$$l_1 = \begin{cases} \sum_{i=0}^{n} 0.5 \times (y_i - f(x_i))^2, |y_i - f(x_i)| < 1 \\ \sum_{i=0}^{n} |y_i - f(x_i)| - 0.5, \text{othervise} \end{cases} \quad (2)$$

We used 500 problems per configuration to train the sVAE model (approximately 375 problems for training and 125 problems for validation; different VAEs for different configurations because the labels did not match across configurations). The sVAE model achieved 100% accuracy in predicting the labels within 10 epochs of training. sVAE can generate accurate images within 100 epochs of training. Panel features and individual object features were extracted from the semantic features after the semantic features of the objects were extracted. The panel feature was the collection of attributes of all objects in the panel. Feature of one attribute took a single value if the attribute values were the same for all objects in the panel or took a sorted (low to high) enumeration of the attribute values if the panel contained objects with different attribute values. The individual object feature listed all objects and their attribute values in the panel. If the second and third panels contained objects of the same type as the first panel, the order in which the objects were listed in these latter panels followed the order of their most similar same-type objects in the first panel. We used the individual object feature to describe how the attributes of an object changed in a row because the first object listed in the second and third panels in a row corresponded to the object in the first panel. This process mimicked the hierarchical processing of humans, who perceive at different levels and compare at different scales [11]. The panel feature and the individual object feature were the inputs to the cognitive map reasoning back-end.

The semantic feature layer in sVAE provided a semantic description of objects, and the decoder generated objects with variations in semantic features. The semantic concepts perceived by sVAE led to more efficient training with less data because they were semantically generalizable, similar to humans. The model did not need to see all objects to reason, using images of 1440 objects to learn 2400 objects (see "Result" and "Discussion" for details).

Cognitive Map Reasoning Back-end

A cognitive map, or mental model, is a type of mental representation or structured knowledge stored in long-term memory (LTM). Human use cognitive maps to acquire, encode, store, retrieve, decode, and infer the features of spatial locations (or nonspatial abstract locations). Cognitive maps are formed through experience. When people encounter a new situation without prior knowledge (cognitive maps), they experience links and relationships between all members of the scene. Some of these connections are crucial, while others are trivial. Success in similar situations depends on the critical links. With feedback and contrasts from other scenes, people gradually learn to focus only on the critical connections in similar situations. The neural representations of these critical links thus form cognitive maps and can be activated by similarly structured cues [4, 36] for similar situations. The cognitive map reasoning back-end (CMRB) used a similar logic to generate the cognitive maps for RPM problems by experiencing RPM problems in the training set according to the following steps (see Algorithm 1).

Step 1: Given the first RPM problem in the training set, the semantic features for the first eight panels and the answer panel of the problem were extracted by the sVAE module, resulting in a 1×9 vector for each attribute (type, size, color, and position). A numerical relationship (i.e., the difference between two values or whether the first plus the second equals the third) can be calculated between any two or three of these nine elements in the vector. In other words, if the value of the ath element is greater than that of the bth element by one, then the relationship between the two elements is "+ 1"; otherwise, it can be "0" or "− 2." If there are three elements (ath, bth, and cth), the types of the relations can be expressed as "$a+b=c$," with each expression corresponding to a specific relation type. The cognitive map module assigns the relationship between the ath, bth, and cth elements as "1," for example, if a, b, c follows "$a+b=c$" and the relation type number of "$a+b=c$" is "1." The numerical relationships or relationship types can be determined for every two or three elements $a \in (1,9)$, $b \in (1,9;\ b \neq a)$, and $c \in (1,9;\ c \neq b \neq a$ in three elements case). All two-two and three-three relations combine to make a 9×9 matrix or $9 \times 9 \times 9$ tensor. Figure 3 and Appendix 3 section "Cognitive Map Training Details" provide more detailed descriptions. When the problem contained 2×2 or 3×3 grids, the position information of these grids formed separate position feature maps, resulting in three types of feature maps (i.e., attribute, 2×2 position, and 3×3 position feature maps). The feature map was symmetric; thus, only the lower triangle in the feature map must be considered. The feature maps and the RPM problem that generated the maps were stored in LTM, which has a capacity of 30.

Step 2: Given a second RPM problem in the training set, 9×9 matrix (or $9 \times 9 \times 9$ tensor) feature maps CM_{new} can be computed. We compute the 8th-order principal submatrices (subtenors) of CM_{new}, denoted $CM_{new,cue}$, as the feature maps of the problem without the answer panel, and define the similarity between two maps as the number of elements in the feature map $CM_{new,cue}$ that are equal to the corresponding elements in $CM_{old,cue}$.

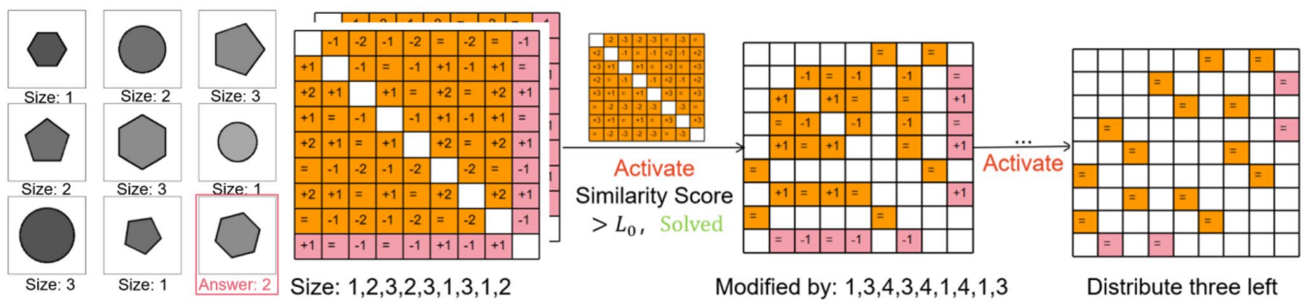

Fig. 3 Schematics of the cognitive maps. Left: The algorithm forms a feature map for an RPM problem with size attributes 1,2,3,2,3,1,3,1,2 (panels 1–9; panel 9 is the answer panel). As described in the text, the feature map (orange and pink tables in the figure) is a 9×9 matrix. The (a,b) position in the feature map represents the numerical relationship of the size attribute between panels a and b. ("=" means equal, "+1" means the size of panel a is greater than panel b by 1). Middle: The algorithm encounters a similar instance (similarity score > L_0), creates a temporal cognitive map (keeping only the effective elements of the old cognitive map that have the same value as the feature map of the new problem), and solves the problem using the temporal map. Right: After multiple activations and updates, the final cognitive map describes the "Distribute Three Left" relation

Algorithm 1 Acquire and apply cognitive maps

```
Require: feature vector $d = x_1, x_2 ... x_9$//The attribute values of the first 8 panels and answer panel.
Ensure: CM index, p//The cognitive map index and the attribute prediction
1: for $d$ in training set do//Enumerate feature vectors of training problems
2:     $CM_{new,9\times9}, CM_{new,9\times9\times9} \leftarrow generate\ relationmap(d)$
3:     if $id > 2$ then//Later than the second problem
4:         $CM_{cue,8\times8} \leftarrow CM_{new,9\times9}[:8,:8]$
5:         $CM_{cue,8\times8\times8} \leftarrow CM_{new,9\times9\times9}[:8,:8,:8]$//Construct cues
6:         for $CM_{old}$ in $LTM$ do
7:             $L2 \leftarrow compare(CM_{cue,8\times8}, CM_{old,9\times9})$
8:             $L3 \leftarrow compare(CM_{cue,8\times8\times8}, CM_{old,9\times9\times9})$//Compute similarity
9:             if $L2 > L_0$ or $L3 > L_1$ then//Activate similar cognitive maps
10:                $CM_{temp} \leftarrow same(CM_{new}, CM_{old})$//Find same relationships
11:                $s_1, p \leftarrow solve(CM_{temp}, d[0:8])$//Solve the new problem
12:                $s_2, p \leftarrow solve(CM_{temp}, d\ old[0:8])$//Solve the old problem
13:                if $s_1 == 1$ and $s_2 == 1$ then
14:                    $update(CM_{old}, CM_{temp})$//Update the cognitive map
15:                    $count(1) += 1$//Count the frequency of use
16:                    break//Break loop
17:                end if
18:            end if
19:        end for
20:    end if
21:    if $s_1 == 0$ and $s_2 == 0$ then
22:        $save(CM_{old,9\times9}[argmin(count)], CM_{temp,9\times9})$//save or replace
23:        $save(CM_{old,9\times9\times9}[argmin(count\ 1)], CM_{temp,9\times9\times9})$
24:    end if
25: end for
```

Step 2.1.1: If the similarity between $CM_{new,cue}$ and all $CM_{old,cue}$ in LTM was less than the similarity threshold L_0 (L_1), the feature maps CM_{new} were novel for CMRB (see Appendix Tables 7, 8, and 9 for the choice of L_0 and L_1).

Step 2.1.2: If the number of feature maps reached the capacity of the LTM, CM_{new} and the problem replaced the least active feature map and its associated problem. Otherwise, CM_{new} and the problem were saved directly to the LTM.

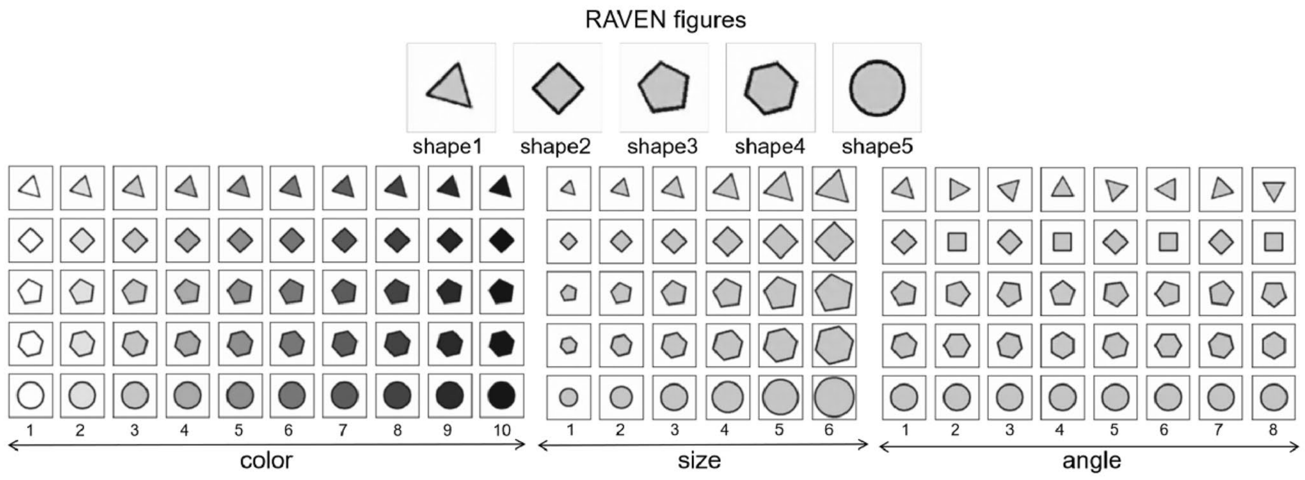

Fig. 4 Example of generated objects. sVAE can generate objects with semantic descriptions (latent semantic dimensions) by assigning values to the semantic features type, color, size, and angle. This figure shows five types of objects (top) with ten colors (bottom left), six sizes (bottom center), and eight angles (bottom right) generated by sVAE

Step 2.2: If the similarity between $CM_{old,cue}$ and $CM_{new,cue}$ was greater than the similarity threshold L_0 (L_1), a similar old feature map CM_{old} was activated. Then, a temporal feature map CM_{temp} was created in memory. The elements in CM_{temp} were set to the value of the corresponding elements in CM_{old} if an element in CM_{new} was the same as the corresponding

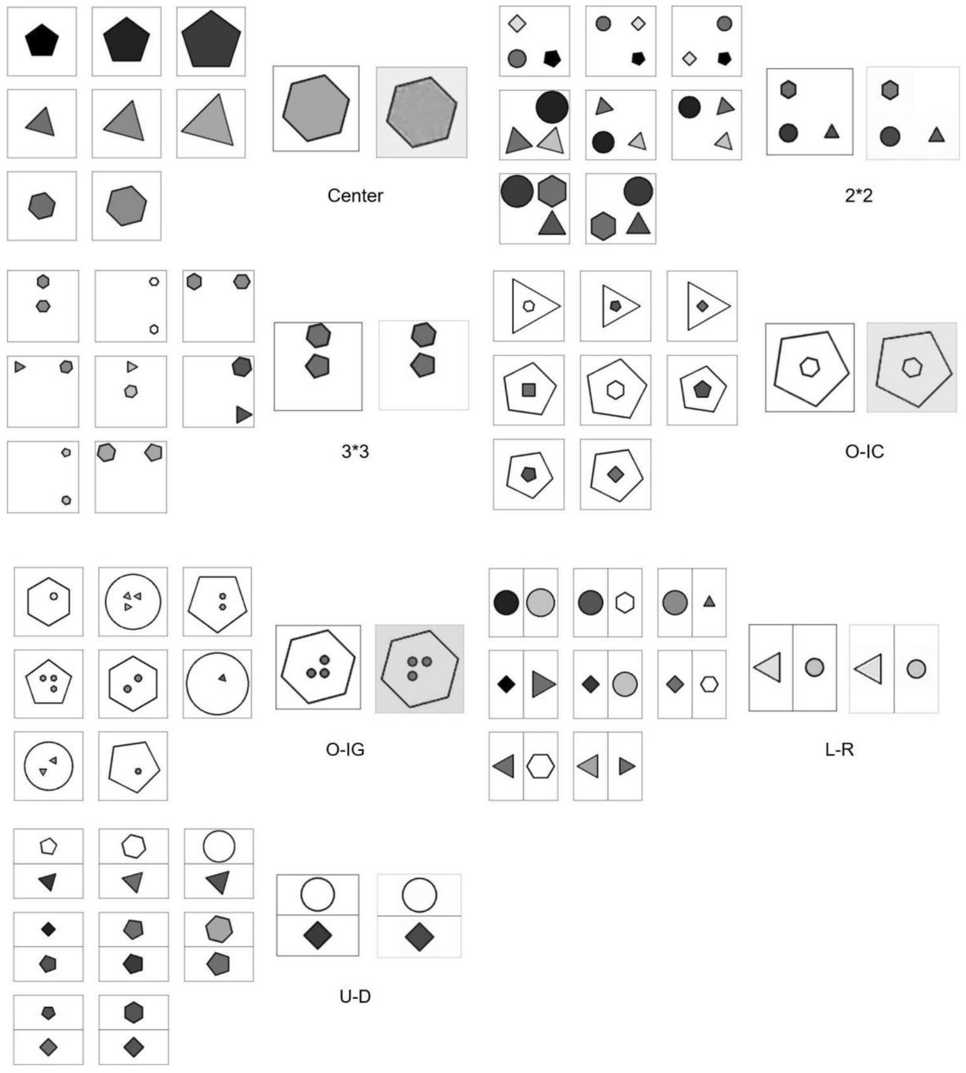

Fig. 5 Sample answer image generated by sVAE. One problem for each configuration (as labeled) is shown. The left side of each configuration shows the problem, and the right side shows the answer image provided by RPM dataset (left) and the answer image generated by model (right)

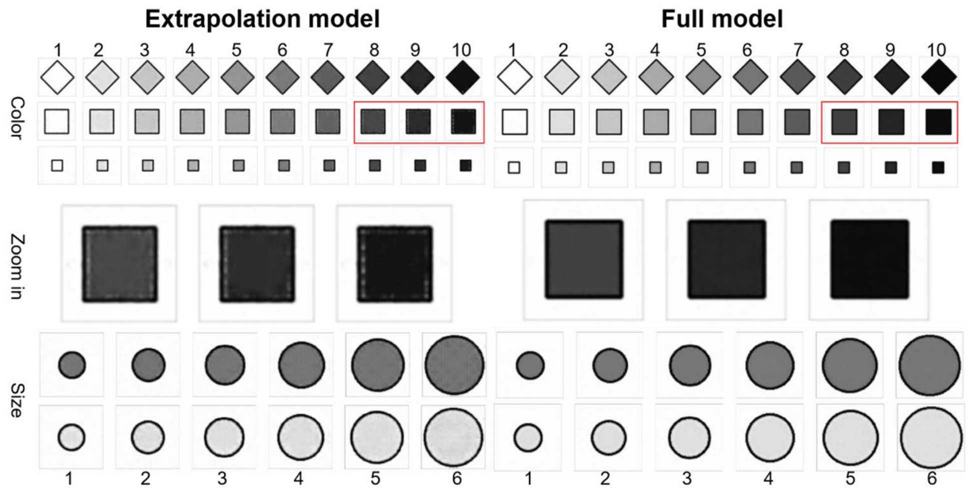

Fig. 6 The extrapolation models (left) trained on partial data of the training set without squares of color 5-10 (top) or circles of size 4-6 (bottom) produce similar images compared to the full model (right) trained on all data of the training set. The numbers above and below the images indicate the values of the color and size attributes, respectively. Details produced by the full model are better (see zoomed images in the middle row of the figure)

element in CM_{old} (and to zero if not). CM_{temp} encoded the collective cognitive map memory in CM_{new} and CM_{old}.

Step 2.2.1: Predict the values of the attributes of the objects in the missing panel of the new problem and the CM_{old}-related problem stored in LTM using the nonzero elements in the 9th row of CM_{temp}. If more than half of the predictions were correct, CM_{old} in LTM was updated by CM_{temp}. We examined the next problem in the training set.

Step 2.2.2: If more than half of the predictions were wrong for all CM_{temp}s, which were created by CM_{old}s in LTM, CM_{new} was new to CMRB. CM_{new} and the problem were saved to the LTM or as a replacement for the least active feature maps and problems, depending on the number of feature maps in the LTM. We examined the next problem in the training set.

Step 3: When all the problems in the training set had been examined, the CMRB completed the acquisition of cognitive maps of the RPM problems and stored the maps in the LTM.

The CMRB can then apply the cognitive maps to solve RPM problems. Given an RPM problem in the test set, the sVAE module extracts the semantic features of the first eight panels of the problem. CMRB then computes the similarity between CM_{cue} and the feature maps in LTM. The feature maps CM_{pre} in LTM that fully match CM_{cue} can predict the attributes of the missing panel using the elements of the 9th row of the maps. The sVAE module then generated the image of the missing panel using the predicted values of the attributes.

The mechanism of the CMRB module is specified in Algorithm 1, and the functions in the algorithm are conceptually generalizable. The Compare function took two cognitive maps (matrices) as input and output similarity scores (number of identical elements, as in step 2). The same function created CM_{temp} with CM_{old} and CM_{new} as in step 2.2. The Solve function took a cognitive map and an incomplete feature vector (the features of the first eight problem panels) as input and applied transitions in the cognitive map, which are defined as a 9×9 matrix or $9\times9\times9$ tensor, as shown in Fig. 3, to solve the missing features. The Update function updates CM_{old} with CM_{temp}. The Save function saved the learned CM_{temp} to long-term memory or as a replacement for the least active feature maps.

Result

We evaluated the performance of the neuro-symbolic sVAE-CMRB model on the RAVEN, I-RAVEN, and RAVEN-fair datasets (see Appendices for details), compared its performance with other baseline models, and analyzed its errors. Additional results are provided to illustrate the interpretability and generalizability of the model.

Baselines

First, we compared the performance of the proposed model with (1) the neuro-symbolic models PrAE and ALANS; (2) the best-performing algorithms Rel-air, MRNet, SCL, and SRAN; and (3) the VAE-based methods LoGe (VQ-VAE) and C0 (VAE). For comparison purposes, in most cases, we used the performance published in the literature. However, many models were not tested on all three datasets; thus, we enriched the results of some models by performing

Table 3 Extrapolation performance of sVAE on untrained objects, all objects, and RPM problems

Models	Untrained dimensions	Perception accuracy	RAVEN accuracy
Extrapolation	0.9781	0.9957	0.977
Extrapolation +240 images	1	1	0.989
Full training	1	1	0.989

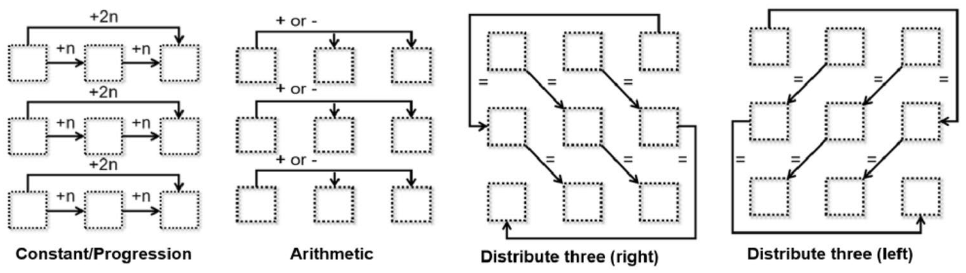

Fig. 7 Four typical cognitive maps learned by CMRB. Each map represents a progressive relationship. The nine squares in a map represent the nine panels of the RPM, and the annotated arrows indicate the numerical relationship between the features of different panels at the panel or object level

experiments with open codes on the datasets not tested by the authors, as indicated by asterisks in Table 1. Second, we compared the performance of the proposed sVAE model with the traditional β-VAE model and the performance of the CMRB with manually designed cognitive maps (see "Result" and Appendices). Finally, we compared the performance of the proposed model with perfect performance as a baseline to show where it went wrong.

Result for the Semantic-VAE Module

The sVAE module achieved interpretability by generating accurate, fully disentangled semantic interpretations for all items and by generating accurate figure representations for all semantic descriptions. The model performed better at disentangling and generating objects than the baseline β-VAE model (figure reconstruction error: 0.00077 vs. 0.00252; disentanglement score: 1.000 vs. 0.961). Perceptual accuracy in generating semantic interpretations was 1.0 for all configurations except the O-IG (out-in-grid) configuration, whose accuracy was 0.99. sVAE generated clear and meaningful objects (Fig. 4) and answer images (Fig. 5) with the semantic description generated by CMRB.

Its generalizability was evident in regard to generating semantic interpretations (for objects) or figures (for descriptions) for similar but unlearned items. We conducted an extrapolation experiment to determine whether the algorithm can understand and generalize Raven's key semantic dimensions. Size, color, and type are three semantic dimensions in the Raven context that are crucial to defining laws and relationships. In particular, the semantical understanding of size and color can be transferred from one type of object to another. In the extrapolation experiment, we excluded all squares with colors 5–10 (or circles with sizes 4–6) from the training set containing images of objects from the 500 RPM problems. After training with this ablated training set, the model never saw any squares with colors 5–10 (or circles with sizes 4–6) but could accurately predict the semantic color features (5–10) of the squares (or size features (4–6) of the circles) and could generate accurate images for squares with colors 5–10 (or circles with sizes 4–6; Fig. 6), suggesting that the sVAE did not simply complete the task by memorizing the learned items but rather "understood" the semantic dimensions of color and size beyond the scenes it had acquired (object types) to new scenes (squares or circles). This process follows the definition of semantic knowledge [37], and this organic understanding is generalizable to similar untrained objects. We fed 240 images of untrained objects (one image per object) to further train the extrapolationally trained sVAE. After training, the performance of the further trained sVAE was equal to the performance of the originally trained sVAE on 500 problems (Table 3).

Results for the Cognitive Map Module

CMRB is the first reasoning backend that provides interpretable, transparent descriptions of the strategies induced by CMRB in the form of cognitive maps. Figure 7 shows 4 typical learned cognitive maps (misleading cognitive maps lead to interpretable errors, see the "Mistakes" section). The cognitive maps implied reasoning that could be transferred to any other problem with similar reasoning, regardless of apparent differences at the pixel or attribute level. These maps were compatible to new logics, where these problems were easily recognized as novel and new cognitive maps were formed. CMRB learned the cognitive maps and achieved high accuracy in predicting the attributes of the missing matrices. The resulting performance was comparable to the baseline of hand-designed cognitive maps (Table 1). The prediction accuracy and the number of learned cognitive maps varied with the similarity threshold L_0 (see Appendices for training and parameter details).

Table 4 Average accuracy for 7 configurations in the RAVEN, I-RAVEN, and RAVEN-fair datasets

Config/datasets	Center	2*2	3*3	O-IC	O-IG	L-R	U-D	Average
RAVEN	0.992	0.9879	0.9931	0.9945	0.876	0.9963	0.9968	0.9767
I-RAVEN	0.9944	0.9742	0.9945	0.9968	0.9147	0.9987	0.9982	0.9816
RAVEN-fair	0.9932	0.9921	0.9954	0.9958	0.9436	0.9975	0.9979	0.9879

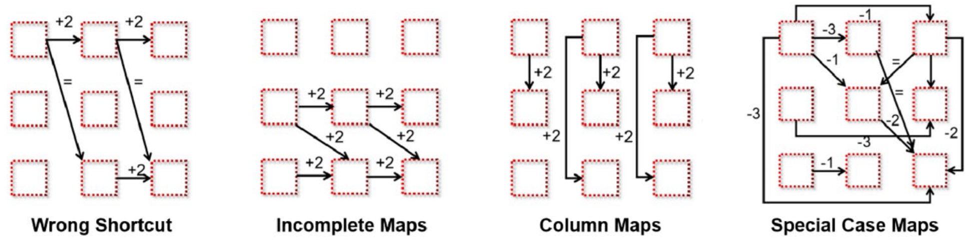

Fig. 8 Four examples of cognitive maps, as in Fig. 7, that are generated by the CMRB and that lead to incorrect predictions. The first map relies on a false short-cut relationship between panels 1 and 8. The second map ignores critical information from the first three panels. The third map uses column wise relationships that are not defined in the RAVEN datasets. The fourth map shows a special case of far-fetched generalizations between two unrelated problems. In this case, knowledge from one problem should not be transferred to the other

Primary Results

The model performed well on all three datasets, achieving good performance in all configurations (Table 4). The proposed model produced the highest overall accuracy among the baseline competitors (Tables 1 and 2; 97.7% on the RAVEN dataset, 98.2% on I-RAVEN, and 98.8% on RAVEN-fair).

Mistakes

The algorithms rarely made mistakes. Perception was nearly perfect unless the image was too small (e.g., in the O-IG configuration). The accumulation of perceptual errors caused a decrease in accuracy in O-IG cases. The cognitive reasoning backend was accurate and powerful, solving 98% of the problems. Occasionally, CMRB recognized some relationships that were not considered by the RAVEN dataset. For example, CMRB might take a shortcut (e.g., the attribute of the 8th panel equals that of the 2nd panel) that was not definitive, ignore critical information in the graph, use unstable column information to make judgments, or make a false generalization between unrelated problems (see Fig. 8). These unrecognized relationships captured by these biased cognitive maps can lead to incorrect predictions. However, humans can make similar errors, particularly when they are less familiar with the premise of RPMs. Errors were reduced with training, particularly after we introduced forgetting (deleting cognitive maps not used for a long time) and prioritized cognitive maps that solved more problems.

Discussion

Inspired by how human solve RPM problems, we used the sVAE-CMRB model to solve RPM problems in an interpretable and human-like way. The sVAE operated in a neural network, while the CMRB operated in a symbolic manner. The model perceived RPM problems in a hierarchical, human-like manner and performed human-like symbolic reasoning, produced understandable intermediate results, and solved RPM problems in a blank-filling manner with high accuracy.

Models' Innovations

Many machine learning models used sophisticatedly designed modules to increase contrast [20], enable shared rule processing [22], or develop higher level convolutions

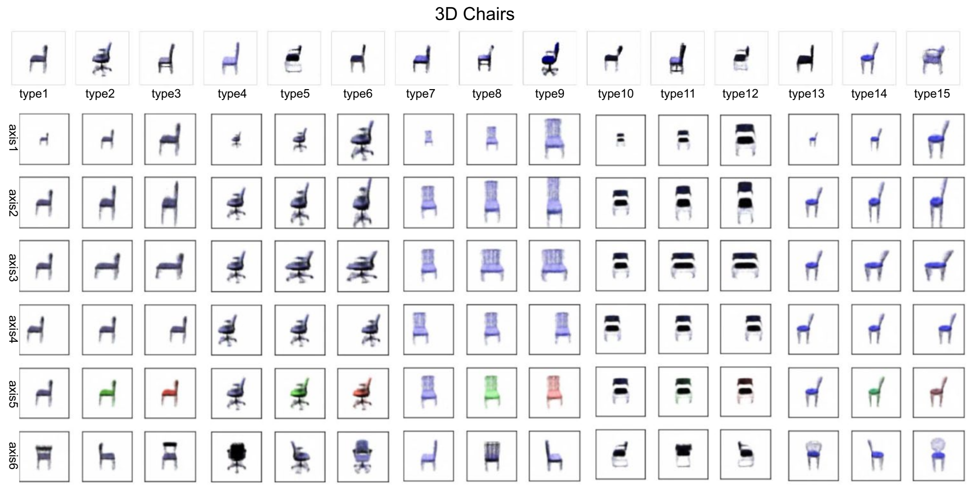

Fig. 9 3D chair generation. The sVAE model learns to describe chairs in terms of the understandable semantic features of type (top row), size (Axis 1), height (Axis 2), width (Axis 3), position (Axis 4), color (Axis 5), and angle (Axis 6) and generates clear and meaningful images with one-hot encodings of the semantic features as its latent semantic units. The model can use these axes to apply transfigurations or create chair images

Table 5 Figure reconstruction errors of sVAE and β-VAE on five datasets

Model	RAVEN	3DChairs	3DFace	celebA	lfw
sVAE	0.00077	0.00933	0.00274	0.01467	0.01408
β-VAE	0.00252	0.00755	0.09461	0.02078	0.11825

Table 6 Disentanglement scores of sVAE and β-VAE on the RAVEN and 3D Chair datasets

Model	RAVEN	3DChairs
sVAE	1	1
β-VAE	0.961	0.969

[21]. However, these models tend to perceive objects differently in nonsemantic dimensions and rely on statistical correlations at the pixel or representation level to solve RPMs. In 2022, Ma et al. proposed "parsimony" as a general principle for how we should build machine learning algorithms [38]. The principle encouraged machines to abstract parsimonious latent dimensions among mass data and build representations on top of them. In this study, we exploited the priority of the human brain and put efforts into building a model that can abstract perceptual dimensions or rules, "perceive" objects, and "communicate" or "generalize" strategies in a human-like way, suggesting a possible way to realize human-like intelligent abstract reasoning. Considering that the human brain has evolved so many years and is sophistical to many challenging problems, we think this kind of brain-inspired model can achieve great success in many specific hard problems in the future.

The sVAE was shown to be a good successor to VAE. With simple supervised constraints, the model performed fine disentanglement and produced clearer images than other VAE-based algorithms on the same datasets [27, 29]. Compared to the β-VAE model [30], the sVAE module had semantic latent features that improved the model's explicability while enhancing its generalization (shown in Appendix Fig. 9). The model's generalizability led to more efficient training and was sufficient to achieve high perceptual accuracy and to solve 90% of the problems with a small training dataset (only 1440 object images). The reported model achieved high performance within 300 problems (see Appendix Table 7), and results suggest that we have made significant progress in variational autoencoders.

Cognitive maps describe people's mental models of how to solve RPM problems [1, 4] and contain conceptual rules that can bind with specific problems. Due to the abstract nature of cognitive maps, reasoning with cognitive maps was different from reasoning with specific rules and did not enumerate all possible rules to solve a problem [21]. Rather, this reasoning applied a general rule to a specific attribute value, which is similar to the human problem-solving process. To our knowledge, the proposed model is the first to apply cognitive maps to the RAVEN, iRAVEN, and RAVEN fair datasets. CMRB achieved high accuracy. As insights for machine learning intelligence, we believe that cognitive maps are essential for achieving human-like abstraction and generalization. The machine learning community should apply the cognitive map to solve reasoning or inference problems. A large-scale application of cognitive maps is likely to result in generalization and other intelligent behaviors, which suggests a possible direction for abstract reasoning in the future (see "Model Innovations" section).

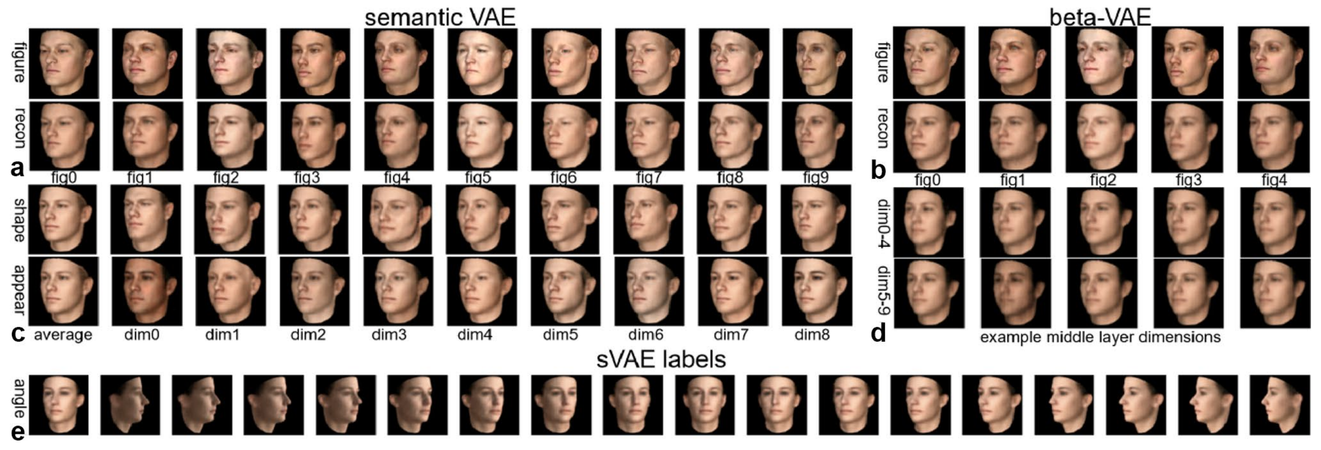

Fig. 10 sVAE describes 3D faces with semantically identifiable dimensions of shape, appearance [52, 53], and labels. sVAE also produces higher-quality figures compared to β-VAE. **a** Face images (top) and sVAE reconstructions (bottom). **b** Face images (top) and β-VAE reconstructions (bottom). **c** Shape and appearance dimensions learned by sVAE. An "average face" (first column) is constructed by setting the semantic units to the mean of 5000 random faces. For the remaining columns, we modify the "average face" by adding 3 standard deviations to one dimension (dim0 to dim8). **d** Latent dimensions learned by β-VAE (as in **c**). **e** sVAE can generate faces from different angles with one angle as input

Generalization

The proposed model can be easily generalized to real-world situations that require abstract visual reasoning. We tested the proposed sVAE model on the 3DChairs dataset [39], the 3DFace dataset [40], the celebA dataset [41], and the LFW dataset [42] (see Appendix 5 for detailed documentation of the generated datasets and labels) with the following settings:

- Encoder: five-layer convolutional-neural-network (CNN, kernel size:3) with 32, 64, 128, 256, and 512 hidden dimensions
- Feature encoder: fully connected feed-forward layer with 16384 nodes
- Semantic feature layer: fully connected feed-forward layer with equal or more nodes to label dimensions
- Feature decoder: fully connected feed-forward layer with 16384 nodes
- Decoder: five-layer transpose CNN (kernel size: 3) with 512, 256, 128, 64, 32 hidden dimensions).

The sVAE model identified meaningful semantic attributes in all datasets and outperformed the β-VAE algorithm in image reconstruction, disentangling, and semantic

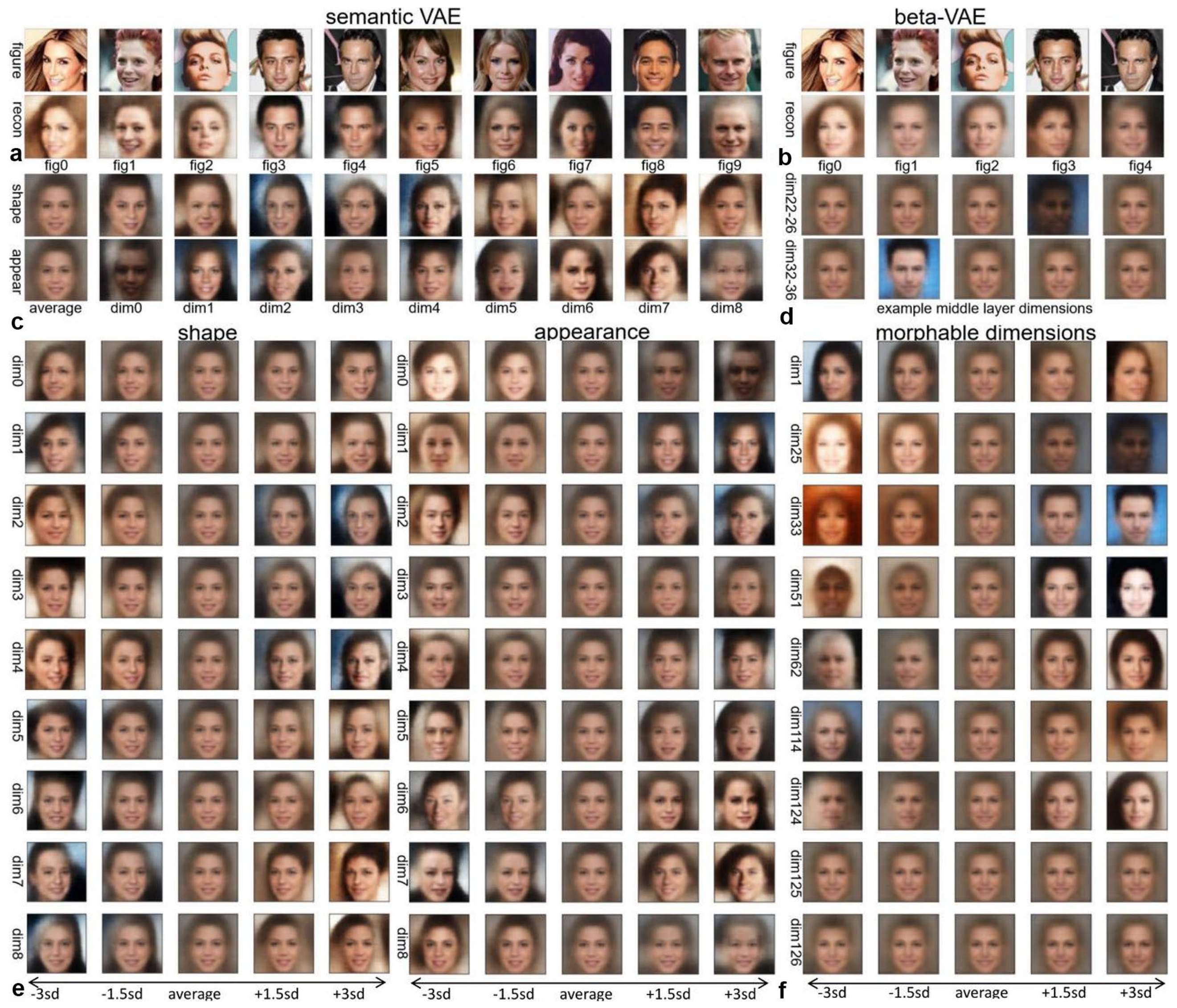

Fig. 11 sVAE describes realistic face images in the celebA dataset with semantically identifiable dimensions of shape, appearance (extracted by the active appearance model as in [52, 53]; see Appendix 5 for details), and labels. sVAE also produces high-quality images (compared to β-VAE). **a** Face images (top) and sVAE reconstructions (bottom). **b** Face images (top) and β-VAE reconstructions (bottom). **c** Shape and appearance dimensions learned by sVAE. **d** Latent dimensions learned by β-VAE (as in **c**). (E) sVAE morphs the “average face” image by decreasing or increasing the value of a semantic dimension (from -3 sd to 3 sd, where “sd” is the standard deviation of 5000 faces). Shape dimensions primarily change the shape of the face (left), while appearance dimensions primarily change the appearance (e.g., eyes and skin color, right). **f** β-VAE morphs average faces by changing the value of an example dimension in **d**

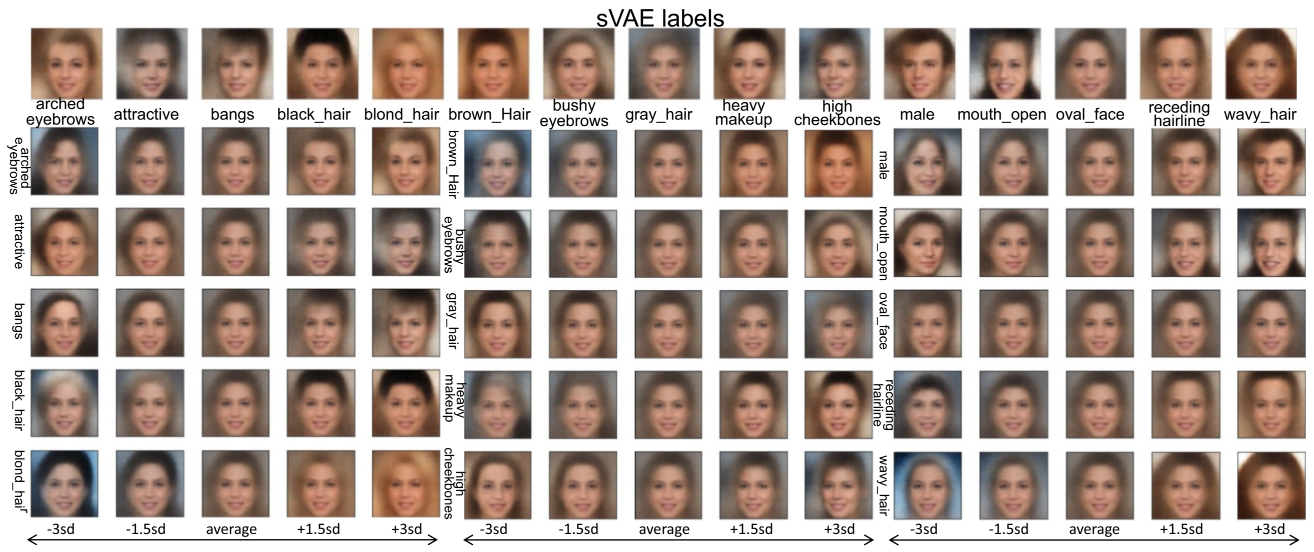

Fig. 12 CelebA provides additional semantic labels for the face images. These labels are used to train the sVAE, along with the computed shape and appearance. sVAE learns to describe images with these additional labels and morphed images using semantic features derived from the labels. Top: Images morphed from the "average face" by setting the value of a semantic feature (from additional labels) to + 3sd above the mean. Others: Images morphed from the average face by changing the value of the semantic labels (from -3 sd to + 3 sd)

morphing. Tables 5 and 6 show the sVAE and β-VAE algorithms' reconstruction errors (mean square error between reconstructed and original images) and disentanglement scores (accuracy with which differences in latent features between two images predict changes in defining attributes [30]) on the datasets. We did not measure disentanglement scores on the face datasets because there are no defining attributes. Figures 9, 10, 11, 12, 13, 14, and 15 show how we can use sVAE to semantically construct and morph chairs and faces. In these more meaningful settings of object and face, the CMRB should be able to infer relationships between objects or faces, as in the new problems we proposed in Fig. 16. This result has real-world implications for reasoning and suspect portrait generation based on eyewitness evidence [43].

In this study, we provided a method to generate the cognitive maps for RPM problems. Compared to traditional cognitive map experiments that generalize in sequential problems with spatial relations [2, 34], the proposed model was an instantiation of the cognitive map concept at the conceptual level, where the generalizable relations can be numerical. We have shown that the model can generalize to real-world stimuli of objects and faces. We believe that the model can be broadly generalized to other circumstances, such as real-world problems that have a state-space representation with certain generalizable types of relations between state-spaces of the concerned system [2]. For example, Whittington generalized cognitive maps to make family tree predictions [4], and Son generalized cognitive maps to enable flexible inference in social networks [5].

An important factor that accounts for the more abstract representation and generalization in the cognitive map is that it (the grid cell representation) factorizes lower-level representations (place cell representations) into upper-level eigenvectors [2]. This factorization can be different in conceptual state-spaces. In this study, with the development of sVAE, we have enabled a similar factorization in conceptual dimensions, factorizing an abstract state (figure) into its latent dimensions. Because sVAE can be generalized to disentangle more complex real-world stimuli such as faces and chairs, the perceptual front end should enable better generalization of cognitive maps at the conceptual level in realistic visual environments.

Neural Basis of Reasoning in RPM

The proposed model mimics the perception and reasoning processes in the human brain. Hierarchical perception takes place primarily in brain areas V1–V4 and IT. Visual areas rely on feedback projections and experience to segment objects from their environment [44]. Categorical coding of object segments can be achieved in IT [45]. For reasoning, we rely on long-term memory, and structured knowledge can take the form of cognitive maps [36]. In the brain, cognitive maps reason by jointly activating neurons that encode spatial and other features. This process implicates hippocampal, entorhinal, and frontal neurons [3, 4]. fMRI research suggests that frontal and parietal regions, visual working memory–related brain areas [46], and the caudate in the basal ganglia [47] are more activated when solving RPM.

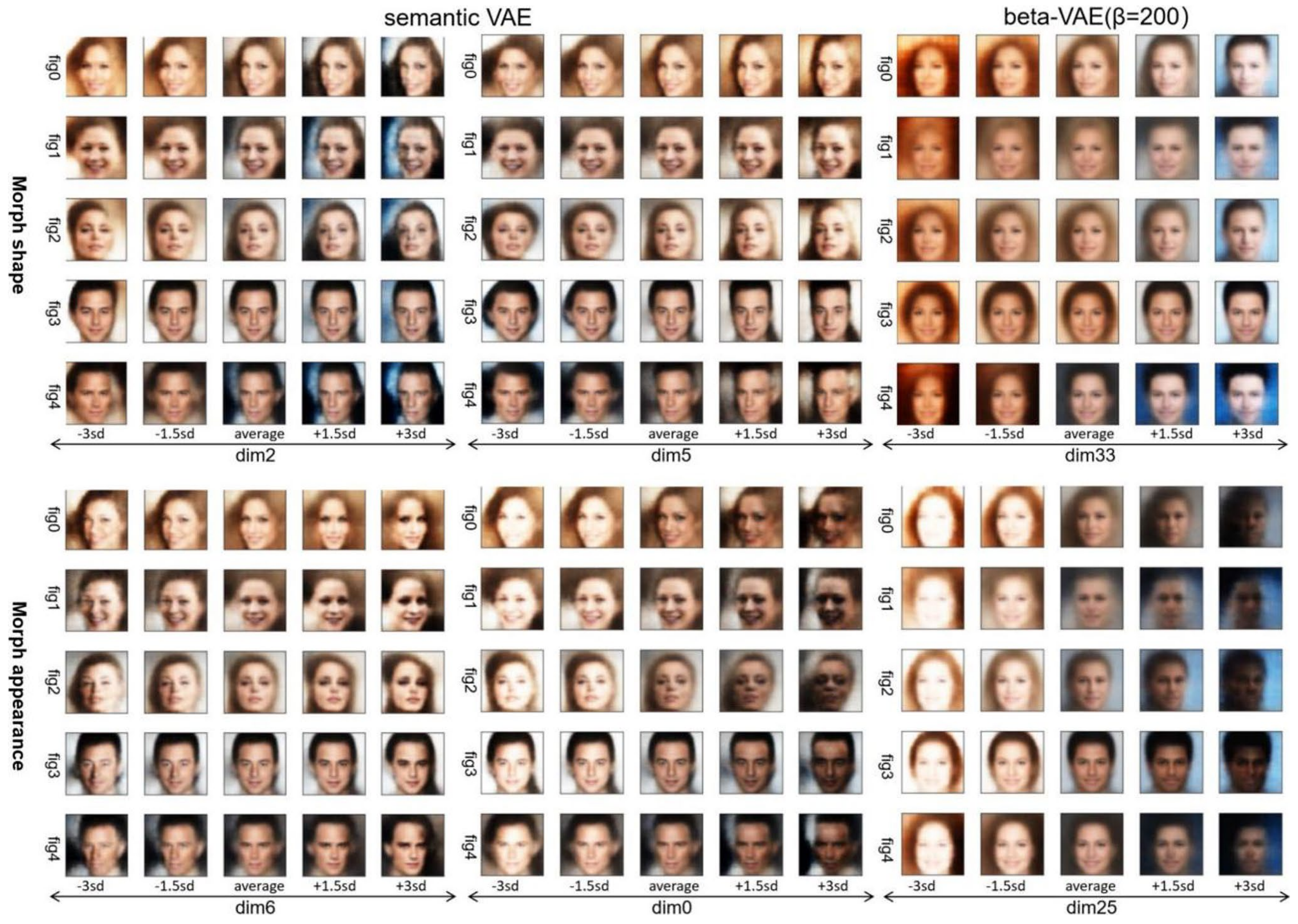

Fig. 13 sVAE produces high-quality figures that morphs the shape (top) and appearance (bottom) of sample images (left: sVAE, right: β -VAE). The annotated dimensions are changed from the disentangled value of the figure to the value plus -3 sd to 3 sd to produce morphed images. Compared to sVAE, β-VAE morphs images in a more entangled way (e.g., when introducing changes in skin color (bottom)), sometimes the gender and shape of the faces also change

Limitations and Future Work

We have introduced human-understandable features in our model, primarily represented by the neurons of the bottleneck of sVAE. These features are ad hoc because they are trained by and specific to the data in RAVEN problems. If we want to apply it to nonraven problems, we need to train them using the dataset of the new problem, which implies

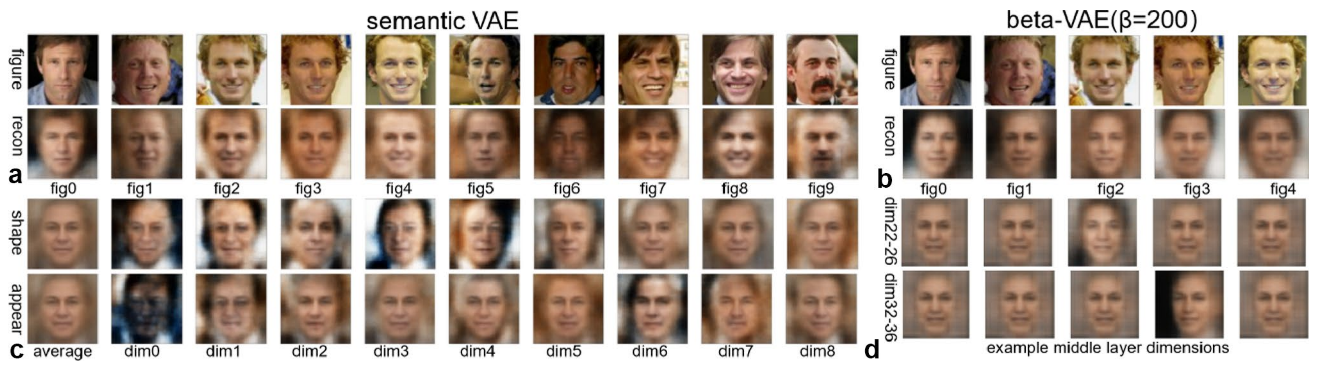

Fig. 14 As in the celebA dataset, sVAE represents realistic face images in the LFW dataset with semantically identifiable dimensions of shapes, appearances [52, 53], and labels and produces high-quality figures (compared to β-VAE). **a** Face images (top) and sVAE reconstructions (bottom). **b** Face images (top) and β-VAE reconstructions (bottom). **c** Shape and appearance dimensions learned by sVAE. **d** Latent dimensions learned by β-VAE (as in **c**)

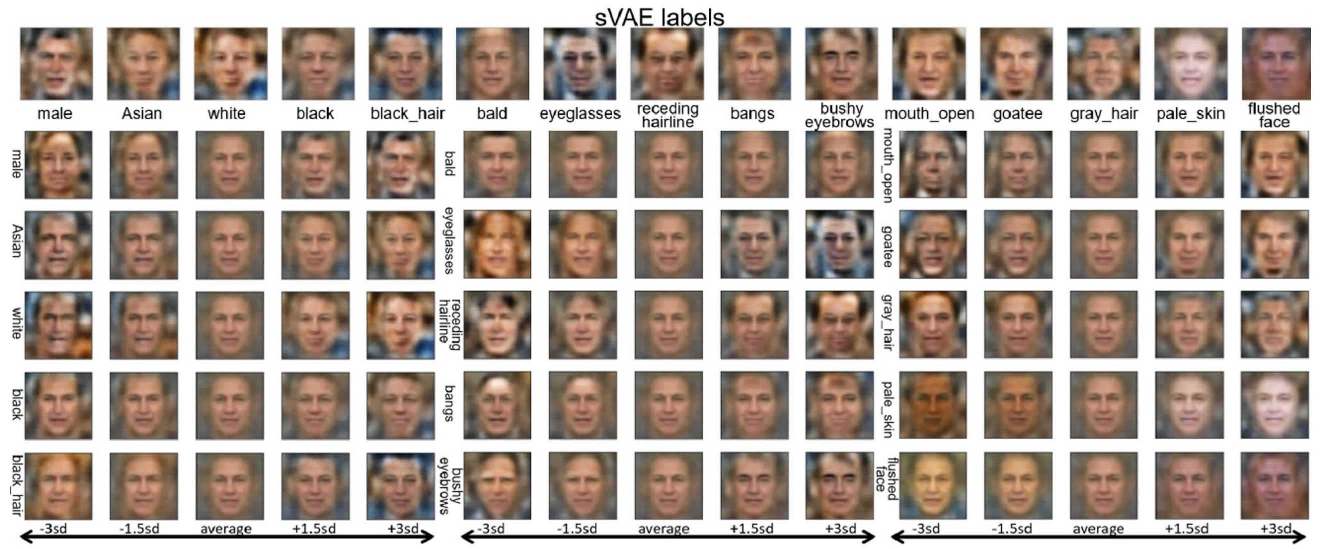

Fig. 15 LFW dataset also provides additional semantic labels for face images. These labels are used to train the sVAE along with the computed shape and appearance dimensions. sVAE learns these additional labels and morphs images according to the variation of the semantic features

that we cannot directly apply the model to different scenarios. Since sVAE features are ad hoc, we need to be aware of the features of the scenario to enable the models to characterize the set semantically. To overcome these limitations, we need to incorporate a higher-level perceptual module to extract the underlying dimensions, enabling the model to learn scene features autonomously.

Our model requires some specific human designs, such as the dimensions in the semantic feature layer of sVAE and the dimensions of the cognitive map. These designs are inspired by the hierarchical processing and semantic perception of our brain, and how humans acquire and generalize their knowledge using cognitive maps to solve problems. We ensure that the algorithm accumulated knowledge on its own during the learning experience (i.e., semantic knowledge about size and color and cognitive maps that solve the RPM problem) and was beyond the design.

The cognitive maps in this paper were conceptual. In the future, we need to build more biologically plausible models for RPM reasoning. Solving RPM problems in a human-like way provides insight into how we solve real-world reasoning problems. We must improve the encoding capability of the semantic VAE module to manage multiplexed stimuli and learn rich cognitive maps to model general reasoning under flexible real-world conditions.

The proposed model uses a small amount of metadata to train as [48, 49], which is a shortcoming since that the metadata is not available in some real-world scenarios. The lack

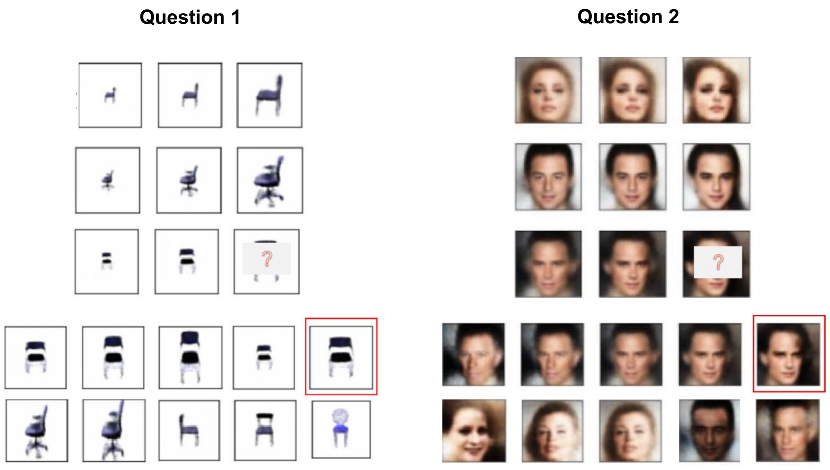

Fig. 16 RPM-like question constructed for realistic images (left: progression in the size of chairs, right: progression in the appearance of faces (the depth of the eye socket)). As in RPM, the algorithms need to abstract the rules that govern the matrix and complete the missing cell (the ninth cell) in a meaningful way. Correct answers are marked with red squares

of metadata is a big challenge for many AI algorithms, especially semantic decoupling AI. Many models have attempted to replace metadata with unsupervised approaches, such as adversarial training [50] and Monte Carlo estimation [51], and have made progress in automatic semantic decoupling. The proposed model achieves good performance under similar objectives by directly using labels and supervision, similar to the early development of children's concept learning. Eventually, we will be able to replace supervision with new algorithms based on simpler principles and achieve unsupervised semantic decoupling.

Conclusion

We built a neuro-symbolic model to mimic human cognitive processes when solving RPM problems. The sVAE module extracted the semantic features of RPM problems in a hierarchical manner, while the CMRB inferred the missing panel based on the semantic feature using cognitive maps. The proposed model achieved good performance on three benchmarks datasets (RAVEN, I-RAVEN, and RAVEN-fair). In the future, more efforts should be devoted to more biologically plausible models. We believe that a deeper understanding of how structural knowledge is stored and retrieved; how perceived dimensionality is organized in the brain; and how cognitive maps are represented and formed must benefit brain-inspired intelligent algorithms for difficult and general real-world problems.

Appendix 1. Code

The code for this study is available at https://github.com/scientific-lab/Toward_Intelligent_Semantic_Reasoning_on_Raven-s_Progressive_Matrices.

Appendix 2. Datasets, Models, and Other Resources

Datasets

We used the RAVEN [15], I-RAVEN [16], and RAVEN-fair [17] data generators to generate standard RPM problems.

The RAVEN dataset [15] (https://github.com/WellyZhang/RAVEN) uses a hierarchical generator to generate problems with different configurations, rules, and attributes. The dataset has 7 configurations: Center, 2 × 2Grid, 3 × 3Grid, L-R, U-D, O-IC, and O-IG; 4 rules: constant, progression, distribute three, and arithmetic. The objects in the problems have 6 attributes: number, position, type, size, color, and orientation. The wrong answers are generated by changing one attribute of the correct answer, which introduces an answer bias; thus, wrong answers indicate the right answer.

The I-RAVEN dataset [16] (https://github.com/husheng12345/SRAN) has the same hierarchical generator and generates wrong answers differently. Each wrong answer generated by I-RAVEN has a 50% chance of changing one of the attributes of the correct answer and differs from the correct answer in more than one attribute; thus, no answer bias exists.

The RAVEN-fair dataset [17] (https://github.com/yanivbenny/RAVEN_FAIR) also uses the same hierarchical generator and uses a different method to generate wrong answers. The algorithm generates one wrong answer at a time. After generating the first wrong answer by changing an attribute of the correct image, the algorithm randomly selects one of the generated incorrect or correct answers and changes one of the attributes of the selected image to generate a new incorrect answer. RAVEN-fair also has no answer bias.

β-VAE Module

The sVAE module was developed from the public code of the β-VAE module by Higgins [30]. The original code of the β-VAE module is available at https://github.com/AntixK/PyTorch-VAE.

Comparison Algorithms

We ran PrAE [21] on the RAVEN-fair dataset. The model is publicly available at https://github.com/WellyZhang/PrAE.

We ran Rel-base [19] on the I-RAVEN and RAVEN-fair datasets. The model is publicly available at https://github.com/SvenShade/Rel-AIR.

We ran MRNet [17] on the I-RAVEN dataset. The model is publicly available at https://github.com/yanivbenny/MRNet.

We ran SCL [20] on the RAVEN-fair dataset. The model is publicly available at https://github.com/dhh1995/SCL.

We ran SRAN [16] on the RAVEN and RAVEN-fair datasets. The model is publicly available at https://github.com/husheng12345/SRAN.

The model architectures and model parameters used are the same as those in the files.

Appendix 3. Training Details

FCNN Training Details

We trained the FCNN model on a CPU platform (Windows 11 64-bit; Intel(R) Core(TM) i3-10110U CPU @ 2.10 GHz (4 CPUs)), and the model contains 4 convolutional layers (16, 32, 64, and 128 latent dimensions, kernel size 5 * 5, stride 1, padding 1) and a feedforward layer (16384 nodes). We generated 959 problems per configuration to train the

FCNN module. For each problem, the model takes the first image as input and the configuration index of the problem as the label (cross-validation). The model was trained for up to 50 epochs and achieved 100% accuracy when classifying the configurations of the images.

sVAE Training Details

We trained sVAE on the institute's GPU platform (NVIDIA SMI, 460.80; driver version, 460.80; CUDA version, 11.2).

Five hundred problems were generated for each configuration to train and validate the sVAE module (approximately 375 training problems and 125 validation problems). Each problem consists of 16 NumPy array figures containing one or more objects. We segmented the figures into individual object images according to the structural organization of the figures. We used the cropped object images and 1 * 29 vectors describing the objects' meta-information (type, size, color, and angle) obtained from the corresponding problem xml file to train the sVAE module. We trained one sVAE model for each configuration, except for the O-IC and O-IG configurations, where we trained two separate sVAE models for the "in" and "out" configurations. The model was trained quickly, requiring less than 5 min and 10 epochs to obtain acceptable results. We trained the models up to the 100th epoch (or 50th epoch for the out configurations) to obtain good reconstructions.

The model used four losses to train: the object reconstruction loss, the latent variable reconstruction loss, the supervised loss (difference between the semantic features and the labels), and the regularization loss (divergence between the distribution of the latent variables and the assumed distribution). The supervised loss used the smooth l1 loss, as in Eq. 3, while the other losses used the mean squared loss (mse), as in Eq. 4:

$$l_1 = \begin{cases} \sum_{i=0}^{n} 0.5 \times (y_i - f(x_i))^2, |y_i - f(x_i)| < 1 \\ \sum_{i=0}^{n} |y_i - f(x_i)| - 0.5, \text{othervise} \end{cases} \quad (3)$$

$$L = (y_i - f(x_i))^2 \quad (4)$$

We evaluated the performance of the sVAE model on different training sample sizes (from 100 to 959 problem object segments). The model achieved top perceptual accuracy within 300 training problems and accurately answered RPM problems with human-designed cognitive maps. The reported model was trained on 500 problems (see Table 7).

The model did not need to see all objects to reason effectively. RAVEN contains 2400 objects. in this study, we used only 240 to 2400 images corresponding to 240 to 2400 objects (one image per object) to train the model. The performance is shown in Table 8.

Table 7 Model Perception and problem-solving performance for different sample sizes (number of object images)

Proportion of samples	Number of image	Perception accuracy	I-RAVEN accuracy
0.1	3445	0.9722	0.927
0.2	6891	0.9993	0.989
0.3	10,337	1.0	0.989
0.4	13,783	1.0	0.989
0.5	17,229	1.0	0.989
0.6	20,675	1.0	0.989
0.7	24,121	1.0	0.989
0.8	27,567	1.0	0.989
0.9	31,013	1.0	0.989
1.0	34,459	1.0	0.987

sVAE Image Generation

sVAE can generate clear answer images for RPM problems. To generate an answer image, the algorithm first generated object images according to object features and then arranged them according to their predicted positions. There were two levels of prediction (panel-level and object-level) for both object features and positions. We first considered object-level position predictions, which specify the attribute (feature or position) of a single object. If there was no object-level prediction, we assigned attributes according to the panel-level predictions. The panel-level predictions did not specify which value corresponded to which object. Thus, the order of assignment was randomized when we assigned values according to panel-level positions. If there were no predictions at both levels, we randomly assigned values to the attribute.

The semantic features in the sVAE altered the images in an understandable way, but this result was not possible with β-VAE (Fig. 17) [54].

Table 8 Perception and problem-solving performance of the model trained with different proportions of all 2400 objects

Proportion	No. of images	Val acc(all images)	Perception accuracy	RAVEN-fair accuracy
0.1	240	0.9258	0.0848	0.507
0.2	480	0.9609	0.2707	0.595
0.3	720	0.9961	0.5944	0.786
0.4	960	0.9844	0.6910	0.830
0.5	1200	0.9961	0.8362	0.843
0.6	1440	1.0	0.9221	0.903
0.7	1680	1.0	0.9555	0.933
0.8	1920	1.0	0.9828	0.973
0.9	2160	1.0	0.9936	0.990
1.0	2400	1.0	0.9996	0.994

Cognitive Map Training Details

The cognitive map is trained on a CPU machine (Windows 10 64-bit Intel(R) Core(TM) i7-8700 CPU @ 3.20 GHz (12 CPUs)).

There are three types of feature maps: attribute feature maps, 2×2 position feature maps, and 3×3 position feature maps. Each type of cognitive map has two subtypes: 9×9 feature maps and 9×9×9 feature maps.

To build feature maps for feature vectors containing attributes of the first eight panels and the answer panel, we found categorical numerical relationships between two (three) elements in the feature vectors. The position (a,b) ((a,b,c)) in the 9×9 (9×9×9) feature maps stores the categorical relationship between the ath element and the bth element or among the ath, bth, and cth elements in the feature vector. For attribute feature maps, the relationships in 9×9 feature maps are "+1," "-9," "=," etc., while the relationships in 9×9×9 feature maps are "a+b=c," "a−b−2=c," etc. The relationships for 2×2 location feature maps and 3×3 location feature maps are different from those for attribute feature maps. For example, "+1" defines a position relationship where all objects move to the right and the last object moves to the first. "a+b=c" means that the objects in the third panel occupy all the positions occupied by the first and second panels.

We generated 15,000 problems per configuration to train and validate the CMRB (10,000 for training and 5000 for validation). We used the "Center" configuration problems with 16 images per problem, one object per image, and three attributes per object to extract feature vectors and train the attribute feature maps. The sVAE model was used to acquire attributes from objects and construct feature vectors. A total of 30,000 feature vectors were constructed for the 3 attributes of 10,000 training problems. We provided additional candidate answers for attributes in the test set. If the predicted attribute value is not in the candidates, we allowed the algorithm to use other cognitive maps. The learned cognitive maps can be generalized to other configurations. Similarly, we used the positional information from 2×2 and 3×3 problems to train 2×2 and 3×3 positional feature maps.

We stored the acquired cognitive maps in each epoch (1000 training steps) and selected the best-performing model from the training epochs based on the validation performance. The model was trained quickly and reached 99.5% validation accuracy in the 2nd training epoch with 2000 feature vectors (approximately 700 problems). The model reached its best performance (99.7%) in the 10th training epoch with 10,000 feature vectors (approximately 3000 problems). The performance of position cognitive maps on problems with "position" or "number/position" rules in the metadata was 1.0 (the metadata information about which problem has position relations is

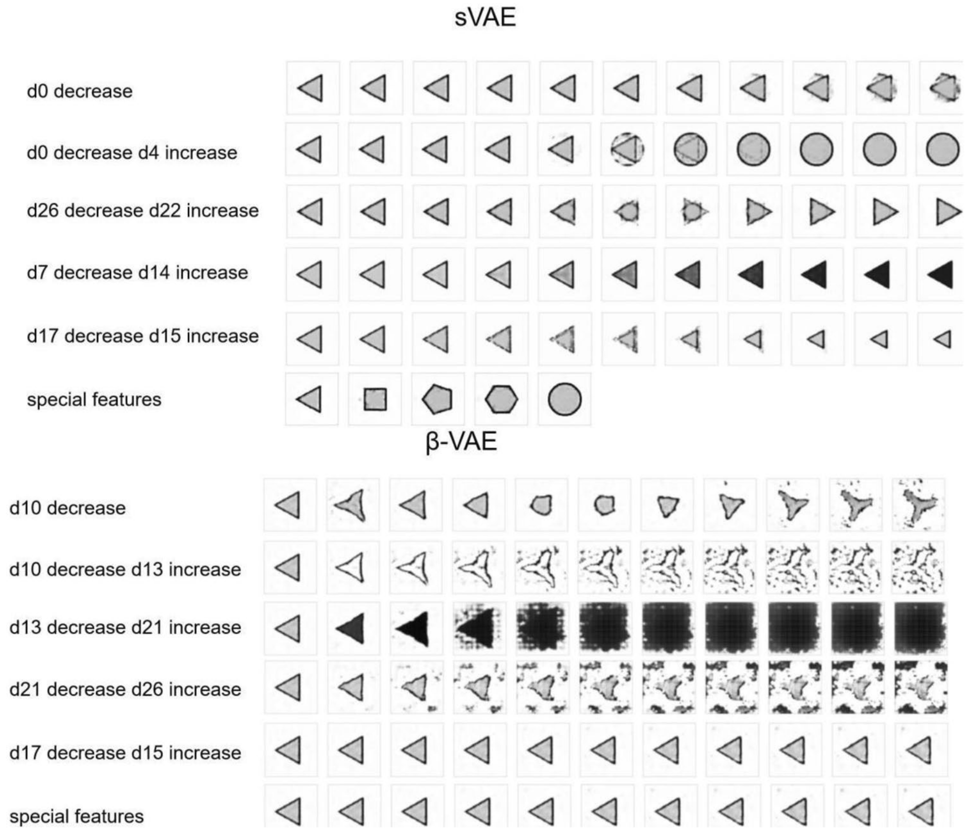
sVAE
d0 decrease
d0 decrease d4 increase
d26 decrease d22 increase
d7 decrease d14 increase
d17 decrease d15 increase
special features
β-VAE
d10 decrease
d10 decrease d13 increase
d13 decrease d21 increase
d21 decrease d26 increase
d17 decrease d15 increase
special features

Fig. 17 When we changed the latent variables in the bottleneck of β-VAE, the change in the generated image was usually unpredictable. The attributes (shape, angle, and size) tended to change together. Many dimensions did not change the image at all. Conversely, when we changed the semantic features in sVAE, the change in the generated images was predictable. For example, we can change the shape of a triangle from a triangle to a circle (second row). Alternatively, we can change its color from light to dark (fourth row). We can also create objects with these semantic features (last row)

Table 9 Performance of attribute feature maps at different similarity thresholds

	$L_1=1$	$L_1=2$	$L_1=3$
$L_0=4$	0.9737(7+6)	*0.9953(7+8)*	0.9768
$L_0=5$	0.9737(7+6)	0.9853(7+6)	0.9752
$L_0=6$	0.9761(7+6)	0.9944(7+7)	0.9841
$L_0=7$	0.9761(7+6)	0.9944(7+7)	0.9805

The italics highlight the selected best performing parameter

not available to the algorithms during training). The threshold similarity score L_0 for 9×9 feature maps and L_1 for 9×9×9 feature maps are parameters for CMRB. The validation performance of the model at different threshold similarity scores is shown in Tables 7, 8, and 9. The parameters that led to the best performance with the least number of cognitive maps (numbers in parentheses) were selected (italics).

LTM has a capacity of 30 for each subtype of feature map and can store a maximum of 180 cognitive maps in total. Finally, the best model generated 49 cognitive maps. 7 (9×9) + 8 (9×9×9) attribute feature maps, 7 + 17 2×2 position feature maps, and 6 + 4 3×3 position feature maps. Many cognitive maps capture the underlying rules in RPM problems. The algorithms also discover some rules that are not considered in RPM problems. These maps can solve some problems efficiently but can also lead to predictions that differ from the data generator (Tables 11 and 12).

Cognitive maps reflect how algorithms see RPM problems. In some cases, they see RPM problems differently than humans. For example, the algorithms sometimes use the plus operation in the position feature maps to solve cases where three different positions have the same value. This view is logically correct, but humans rarely see the problem this way. The algorithm usually discovers relationships between nonadjacent panels, which is unusual for humans This unusual behavior is similar to alphaGo, which has produced some unusual strategies in the game of GO.

Model Testing

Using parameters from the trained FCNN and sVAE modules and cognitive maps from the CMRB module, we tested the algorithms on 70,000 new problems (10,000 problems per configuration), and the resulting performance is reported in the main text (Fig. 18).

Table 10 Performance of 2×2 position feature maps at different similarity thresholds

	$L_1=1$	$L_1=2$
$L_0=2$	1 (14+16)	1 (11+24)
$L_0=3$	*1 (7+17)*	1 (7+25)
$L_0=4$	1 (7+18)	1 (8+25)
$L_0=5$	1 (7+18)	1 (7+25)

The italics highlight the selected best performing parameter

Table 11 Performance of 3×3 position feature maps at different similarity thresholds

	$L_1=1$	$L_1=2$	$L_1=3$
$L_0=4$	1 (6+4)	*1 (6+4)*	1 (6+3)
$L_0=5$	1 (6+4)	1 (6+4)	1 (6+3)
$L_0=6$	1 (6+4)	1 (6+4)	1 (6+3)
$L_0=7$	1 (6+4)	1 (6+4)	1 (6+3)

The italics highlight the selected best performing parameter

Appendix 4. Designed Cognitive Maps

We can draw a cognitive map by hand based on our understanding of the problem (i.e., Fig. 8) and represent it mathematically: (1) we draw the structure and define nine variables (×1,×2,…×9) corresponding to the 9 positions in 3 by 3 matrices; and (2) we draw the edges and describe the edges (relations between the 9 variables) in mathematical terms and functions, i.e., ×2 = ×1 + n, ×3 = ×2 + n (n equals -9 to 9).

When given a new RPM problem, we fed its first 8 attribute values into the first 8 defined variables and observed if the mathematical equations were satisfied. If they were satisfied, we computed the 9th variable based on its relationships to the other variables. The performance of the hand-designed cognitive maps for 7 configurations in the RAVEN, I-RAVEN, and RAVEN-fair datasets is shown in Table 10.

Appendix 5. Generalization Experiment Datasets and Details

In the 3D-chairs dataset [39] (https://www.di.ens.fr/willow/research/seeing3Dchairs/), we selected 20 types of chairs, 21 images per chair (image size, cropped to center 592; resize 256,256; channel no, 3) from left, right, front, and back angles (4–6 images per angle, marginally different from each other), and applied four types of transfigurations (zoom, stretch, shift, and color change with 5, 5, 3, and 5 dimensions) to the images, creating a dataset of 157,500 images (7875 per chair type) and labels (documenting types, transfigurations, and angles).

For 3D face datasets [40] (https://faces.dmi.unibas.ch/bfm/bfm2019.html), we used the Basel face model to construct three-dimensional faces with 53,490 3D vertices. Each vertex had three position indices (x, y, z) describing the topological corresponding position of the vertex and three color indices (r, g, b) describing its texture. A new face was generated by randomly sampling from Gaussian distributions and determining the weight of the first 199 shape principal components (the principal component of the position indices) and the first 199 appearance principal components (the principal component of the color indices). We created a dataset of 25,000 images (image size, resize 256×256; channel number, 3) and labels by taking a photo at the 60° left

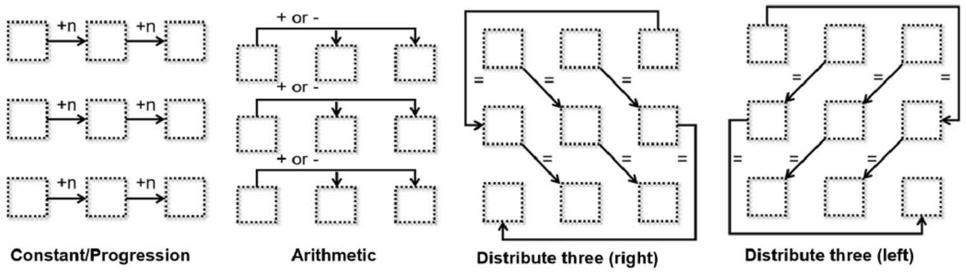

Fig. 18 We can also draw cognitive maps by hand and assign relationships between panels as shown above

Table 12 mean accuracy for the model with designed cognitive maps on RAVEN, I-RAVEN, and RAVEN-Fair datasets

Config/datasets	Center	2*2	3*3	O-IC	O-IG	L-R	U-D	average
RAVEN	0.9883	0.9912	0.9927	0.9916	0.8954	0.9899	0.9876	0.9767
I-RAVEN	0.9896	0.9885	0.9908	0.9943	0.9195	0.9934	0.9929	0.9813
RAVEN-fair	0.9921	0.9950	0.9957	0.9946	0.9531	0.9956	0.9942	0.9886

viewing angle for each constructed face and using the weight of the first 25 shape and appearance components as labels.

The CelebA dataset [41] (http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html) contains 202,599 images (image size, crop to center 148 resize 128 × 128; channel number, 3; random horizontal flip) labeled with 40 binary attributes and ten landmark locations. We also trained an active appearance model (AAM, https://www.menpo.org/menpofit/aam.html) to place 68 landmark point markers carrying shape information of faces to acquire additional brain-like shape and appearance labels. With the acquired shape information from the landmark locations, we morphed the landmarks to match the average landmark locations to produce images carrying shape-free appearance information and projected shape information and shape-free appearance onto 24 principal components. The scores of the images on these shape and appearance principal components, along with the 50 semantic labels from the dataset, were used to train the model, resulting in 98-dimensional labels.

The LFW dataset [42] (http://vis-www.cs.umass.edu/lfw/) contains 13,233 images (image size, crop to center 148 resize 128 × 128; channel number, 3; random horizontal flip) with 73-dimensional numerical labels. We generated an additional 48-dimensional brain-like shape and appearance labels using the active appearance model as the CelebA dataset.

Funding This study was funded by NSFC (grant number 32171094).

Data Availability The datasets generated and/or analyzed during the current study are available in the GitHub repository: https://github.com/scientific-lab/Toward_Intelligent_Semantic_Reasoning_on_Raven-s_Progressive_Matrices. Other public resources related to this work are listed in the Appendices.

Declarations

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Edward CT. Cognitive maps in rats and men. Psychol Rev. 1948;55(4):189–208.
2. Whittington JC, McCaffary D, Bakermans JJ, Behrens TE. How to build a cognitive map. Nat Neurosci. 2022;25(10):1257–72.
3. O'keefe J, Nadel L. The hippocampus as a cognitive map. Oxford university press; 1978.
4. Whittington JC, Muller JC, Mark TH, Chen S, Barry G, Burgess N, Behrens TE. The Tolman-Eichenbaum machine: unifying space and relational memory through generalization in the hippocampal formation. Cell. 2020;183(5):1249–63.
5. Son JY, Bhandari A, FeldmanHall O. Cognitive maps of social features enable flexible inference in social networks. Proc Natl Acad Sci. 2021;39:128–118.
6. Raven JC. Raven's progressive matrices: western psychological services Los Angeles CA. 1938.
7. Bilker WB, Hansen JA, Brensinger CM, Richard J, Gur RE, Gur RC. Development of abbreviated nine-item forms of the raven's standard progressive matrices test. Assessment. 2012;19(3):354–69.
8. Raven JC, Court JH. Raven's progressive matrices and vocabulary scales, vol. 759. Oxford: Oxford pyschologists Press; 1998.
9. Mitchell M. Abstraction and analogy-making in artificial intelligence. Annals of the New York Academy of Sciences. 2021;1505(1):79–101.
10. Małkiński M, Mańdziuk J. Deep Learning Methods for Abstract Visual Reasoning: A Survey on Raven's Progressive Matrices. arXiv preprint arXiv:2201.12382; 2022.
11. Lovett A, Forbus K. Modeling visual problem solving as analogical reasoning. Psychol Rev. 2017;124(1):60.
12. Lovett A, Forbus K, Usher J. A structure-mapping model of Raven's Progressive Matrices. Proc Ann Meeting Cognit Sci Soc. 2010;32.
13. Spearman C. General Intelligence. Objectively Determined and Measured. 1961.
14. Dai WZ, Xu QL, Yu Y, Zhou ZH. Tunneling neural perception and logic reasoning through abductive learning. arXiv preprint arXiv:1802.01173; 2018.
15. Zhang C, Gao F, Jia B, Zhu Y, Zhu SC. Raven: a dataset for relational and analogical visual reasoning. Proc IEEE/CVF Conf Comput Vis Pattern Recognit. 2019;5317–27.
16. Hu S, Ma Y, Liu X, Wei Y, Bai S. Stratified rule-aware network for abstract visual reasoning. Proc AAAI Conf Artif Intell. 2021;35(2):1567–74.

17. Benny Y, Pekar N, Wolf L. Scale-localized abstract reasoning. Proc IEEE/CVF Conf Comput Vision Pattern Recognit. 2021;12557–65.
18. Zhang C, Jia B, Gao F, Zhu Y, Lu H, Zhu SC. Learning perceptual inference by contrasting. Adv Neural Inf Proc Syst. 2019;32.
19. Spratley S, Ehinger K, Miller T. A closer look at generalisation in raven. Eur Conf Comput Vision Springer. 2020;601–16.
20. Wu Y, Dong H, Grosse R, Ba J. The scattering compositional learner: discovering objects, attributes, relationships in analogical reasoning. arXiv preprint arXiv:2007.04212; 2020.
21. Zhang C, Jia B, Zhu SC, Zhu Y. Abstract spatial-temporal reasoning via probabilistic abduction and execution. Proc IEEE/CVF Conf Comput Vision Pattern Recognit. 2021;9736–46.
22. Zhang C, Xie S, Jia B, Wu YN, Zhu SC, Zhu Y, Learning algebraic representation for systematic generalization in abstract reasoning. In Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings,. Part XXXIX. Cham: Springer Nature Switzerland; 2022. p. 692–709.
23. Hua T, Kunda M. Modeling Gestalt visual reasoning on raven's progressive matrices using generative image Inpainting Techniques. CogSci. 2020;2:7.
24. Bourlard H, Kamp Y. Auto-association by multilayer perceptrons and singular value decomposition. Biol Cybernet. 1988;59(4):291–4.
25. Kramer MA. Nonlinear principal component analysis using autoassociative neural networks. AIChE J. 1991;37(2):233–43.
26. Kingma DP, Welling M. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114; 2013.
27. Yu S, Mo S, Ahn S, Shin J. Abstract reasoning via logic-guided generation. ICML Workshop Self-Supervised Learning for Reasoning and Perception. 2021.
28. Van Steenkiste S, Locatello F, Schmidhuber J, Bachem O. Are disentangled representations helpful for abstract visual reasoning? Adv Neural Inf Proc Syst. 2019.
29. Pekar N, Benny Y, Wolf L. Generating correct answers for progressive matrices intelligence tests. Adv Neural Inf Proc Syst. 2020;7390–400.
30. Higgins I, Matthey L, Pal A, Burgess C, Glorot X, Botvinick M, et al. beta-vae: Learning basic visual concepts with a constrained variational framework. Int Conf Learning Represent. 2017.
31. Momennejad I, Russek EM, Cheong JH, Botvinick MM, Daw ND, Gershman SJ. The successor representation in human reinforcement learning. Nat Human Behaviour. 2017;680–92.
32. Dayan P. Improving generalization for temporal difference learning: the successor representation. Neural Comput. 1993;613–24.
33. Todorov E. Linearly-solvable markov decision problems. Adv Neural Inf Proc Syst. 2006.
34. George D, Rikhye RV, Gothoskar N, Guntupalli JS, Dedieu A, La´zaro-Gredilla M. Clone-structured graph representations enable flexible learning and vicarious evaluation of cognitive maps. Nat Commun. 2021;12(1):1–17.
35. Chen L. The topological approach to perceptual organization. Visual Cognit. 2005;12(4):553–637.
36. Steinberg J, Sompolinsky H. Associative memory of structured knowledge bioRxiv. 2022.
37. Patterson K, Nestor PJ, Rogers TT. Where do you know what you know? The representation of semantic knowledge in the human brain. Nat Rev Neurosci. 2017;8(12):976–87.
38. Ma Y, Tsao D, Shum HY. On the principles of parsimony and self-consistency for the emergence of intelligence. Front Inf Technol Electr Eng. 2022;23(9):1298–323.
39. Aubry M, Maturana D, Efros AA, Russell BC, Sivic J. Seeing 3d chairs: exemplar part-based 2d–3d alignment using a large dataset of cad models. Proc IEEE Conf Comput Vision Pattern Recognit. 2014;3762–9.
40. Paysan P, Knothe R, Amberg B, Romdhani S, Vetter TA. 3D face model for pose and illumination invariant face recognition. 2009 sixth IEEE Int Conf Adv Vid Signal Based Surv. 2009;296–301.
41. Liu Z, Luo P, Wang X, Tang X. Deep learning face attributes in the wild. Proc IEEE Int Conf Comput Vision. 2015;3730–8.
42. Huang GB, Mattar M, Berg T, Learned-Miller E. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Workshop Faces "Real-Life" Images: Detect Align Recognit. 2008.
43. Almudhahka NY, Nixon MS, Hare JS. Semantic face signatures: Recognizing and retrieving faces by verbal descriptions. IEEE Transact Inf Forensics Sec. 2017;13(3):706–16.
44. Tschechne S, Neumann H. Hierarchical representation of shapes in visual cortex—from localized features to figural shape segregation. Front Comput Neurosc. 2014;93.
45. Sato T, Uchida G, Lescroart MD, Kitazono J, Okada M, Tanifuji M. Object representation in inferior temporal cortex is organized hierarchically in a mosaic-like structure. J Neurosci. 2013;33(42):16642–56.
46. Prabhakaran V, Smith JA, Desmond JE, Glover GH, Gabrieli JD. Neural substrates of fluid reasoning: an fmri study of neocortical activation during performance of the raven's progressive matrices test. Cognit Psychol. 1997;33(1):43–63.
47. Melrose RJ, Poulin RM, Stern CE. An fmri investigation of the role of the basal ganglia in reasoning. Brain Res. 2007;146–58.
48. Hersche M, Zeqiri M, Benini L, Sebastian A, Rahimi A. A neurovector-symbolic architecture for solving raven's progressive matrices. arXiv preprint arXiv:2203.04571; 2022.
49. Wang D, Jamnik M, Lio P. Abstract diagrammatic reasoning with multiplex graph networks. arXiv preprint arXiv:2006.11197; 2020.
50. Kim H, Mnih A. Disentangling by factorizing. Int Conf Machine Learning. 2018;2649–58.
51. Chen RT, Li X, Grosse RB, Duvenaud DK. Isolating sources of disentanglement in variational autoencoders. Adv Neural Inf Proc Syst. 2018;31.
52. Chang L, Tsao DY. The code for facial identity in the primate brain. Cell. 2017;169(6):1013–28.
53. Chang L, Egger B, Vetter T, Tsao DY. Explaining face representation in the primate brain using different computational models. Curr Biol. 2021;31(13):2785–95.
54. Zhang Z, Song Y, Qi H. Age progression/regression by conditional adversarial autoencoder. Proc IEEE Conf Comput Vision Pattern Recognit. 2017;5810–8.