

A Conceptual Chronicle of Solving Raven’s Progressive Matrices Computationally

Yuan Yang, Deepayan Sanyal, Joel Michelson, James Ainooson and Maithilee Kunda*

Vanderbilt University, Nashville TN 37240, USA

Abstract

Matrix reasoning or geometric analogy problems, like those found on the widely used Raven’s Progressive Matrices test of intelligence, have been used as a challenge for machine intelligence since the early work of Evans in the 1960s. While AI research on the RPM has gone through dramatic shifts alongside other AI advances, including a dramatic rise of machine-learning-based approaches in the last five years, many of these studies have progressed in relatively siloed research lines, making it difficult to compare different approaches and judge progress in the field as a whole. This paper intends to provide a framework for understanding the different lines of work in this research field. In particular, we reviewed 50+ computational models for solving RPM or RPM-like problems and collated them into a linear conceptual framework to help researchers navigate across these diverse research paradigms. We also provide instructions on other resources such as problem/data sets and necessary background knowledge of RPM.

Keywords

Raven’s Progressive Matrices, Visual Abstract Reasoning, Analogical Reasoning

1. Introduction

Raven’s Progressive Matrices (RPM) is a very popular human intelligence test because of its easy administration, interpretability, and non-verbal item format. As shown in Fig. 1, RPM was designed to be multiple choice problems, where the correct choice completes the matrix such that the variations are consistent across rows and columns of the matrix. The first RPM problem set was published nearly a century ago by Raven [1] to study the genetic determinant of human intelligence. While our understanding of the original specifics about test development and target mental constructs have evolved over time, it is still used as a measure of fluid intelligence and even of general intelligence [2]. Later studies [3] showed that RPM indeed occupies a central position among other intellectual ability tests, in that a person’s RPM scores show high correlations with other tests in various different ability domains.

RPM is a masterpiece of item-writing “art”, integrating multiple complexity factors [5, 6] into a single framework. Within this framework, adjusting these factors would produce a large

The 8th International Workshop on Artificial Intelligence and Cognition, June 15–17, 2022, Örebro, Sweden


*Corresponding author.

✉ yuan.yang@vanderbilt.edu (Y. Yang); deepayan.sanyal@vanderbilt.edu (D. Sanyal); joel.p.michelson@vanderbilt.edu (J. Michelson); james.ainooson@vanderbilt.edu (J. Ainooson); mkunda@vanderbilt.edu (M. Kunda)

🌐 <https://my.vanderbilt.edu/mkunda/> (M. Kunda)

🆔 0000-0003-3827-3189 (D. Sanyal)

© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

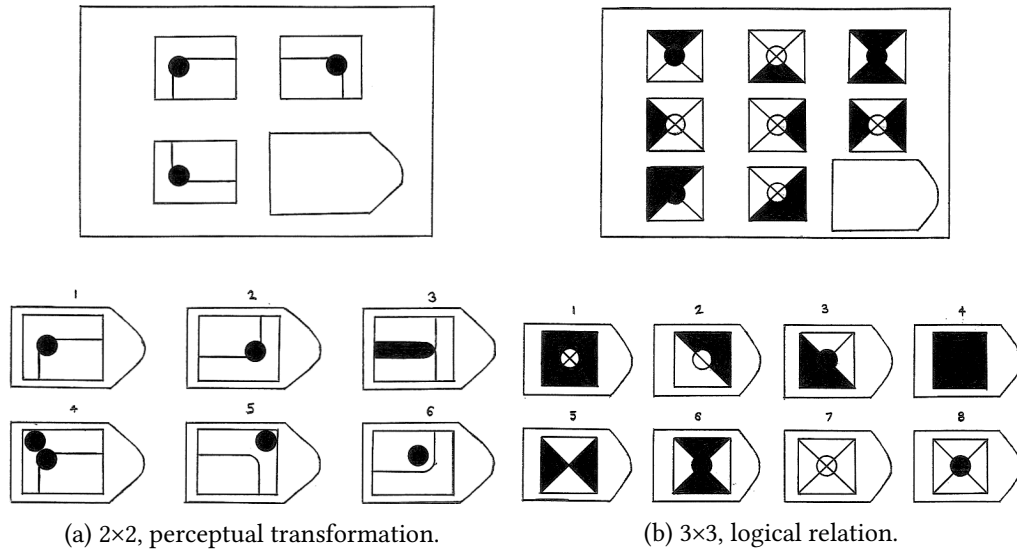


Figure 1: Two example RPMs [4] showing two formats in original RPM— 2×2 and 3×3 —and also illustrating two common ways to interpret/construct RPMs—perceptual transformation and logical relation.

(even infinite) collection of items, which possesses great variability in appearance and a wide range of difficulty. Despite this complexity, RPM items are generally intelligible and solvable to human subjects (up to a reasonable distribution of difficulty). This type of task contrasts sharply with two types of tasks that are commonly considered in machine intelligence. The first type is represented by game playing, where the rules and regulated search space restrict the variability and openness, but the symbolic reasoning is complicated and even challenging for the capacity of human’s working memory; and the second is represented by image classification, which is highly variable but trivial to human’s intellectual ability. Therefore, the openness, variability and moderate difficulty of RPM and its close relation to general intelligence unsurprisingly make RPM a ideal propellant for research on machine intelligence.

Given the above understanding of why researchers seek computational solutions to RPM, we now move on to the technical details. We keep the prerequisite knowledge at a minimal level, requiring no prior knowledge of RPM or solving RPM, and unfold our discussion in a manner that reveals the philosophy behind technical development in simplest language. As the first stop of this journey, we refer our readers to Section A in the appendix for available problem/data sets of RPM. Since the 3×3 format (Fig. 1b) is the most common one among these problem/data sets, we will assume that the format is 3×3 in the following discussion. If the reader is already familiar with RPM problem/datasets, she can directly move on to the next section, where we introduce a linear framework—a conceptual chronicle—that explains the technical development of computational solutions to RPM. The development is divided into four stages with each stage characterized by a distinct high-level approach. In particular, the last stage features the structural evolution of machine learning models for solving RPM, which is further divided into 4 types of models, by which we wish to show the process from the first attempt to solve RPM by machine learning to the ultimate goal of building the visual abstract

reasoning ability through machine learning. As an important complement to the structural evolution of learning models, we give a full description of training techniques that have been highly adapted for solving RPM in Section B in the appendix.

To clarify the terminology, we use RPM to refer to both the original RPM and automatically generated RPM-like items unless they have to be differentiated; in an RPM, various terms have been used to refer to the images in the matrix, such as panel, figure, entry, cell, frame, etc.; in this paper, we use panel, and, thus, refer to the images of given matrix entries as context panels and the images of answer choices as choice panels.

2. A Conceptual Chronicle of Technical Development

To cover all the relevant but heterogeneous works and present them in an understandable way, we decided to take a linear framework, in which we adapted the technical development into a conceptual chronicle of different stages of computational solutions of RPM. This pseudo-chronicle is roughly, though not strictly, chronological, but well serves the purpose of being comprehensive and intelligible. We will use the acronyms of computational models for simplicity and please refer to Table 1 in the appendix for their full names.

3. Stage 1: Imagery-Based Approach

Visual mental imagery, in human cognition, refers to the creation, inspection, and manipulation of visual knowledge representations that do not match concurrent perceptual inputs, and that serve some functional purpose in solving tasks [7]. Using imagery to solve RPM problems—which evidence from psychology and neuroscience suggests does occur (see review in [4])—a person might inspect objects in the matrix, compare them by mentally superimposing one on another, mentally transform the objects, and mentally estimate perceptual similarity. Computational imagery-based systems [8, 4, 9] represent RPM panels by raster images, systematically apply predefined pixel-level operations on the images (e.g., affine transformations and set operations) and calculate pixel-level similarities between the images (e.g., Jaccard index and Hausdorff distance). These systems have proven to be effective for solving the original RPM [10, 11].

4. Stage 2: Logical Reasoning

The imagery-based approach provides an “in-place” solution, i.e., solving a visual reasoning problem “visually” without auxiliary devices or further abstraction of problem representation. While this approach has been found to be successful on RPM problems, this approach is restricted by the predefined pixel-operations and similarity metrics, and the computational cost of them. Logical reasoning, as a task-general and efficient tool in the early development of AI, thus can also be attempted on RPM. In these efforts [12, 13, 14], predicate and propositional representations (or other symbolic representations) are predefined to describe the objects and rules in RPM problem sets. These efforts were initially focused on modeling human’s cognitive behaviors of solving RPM, by comparing different symbolic representations and control mechanisms, and, at

the same time, laid foundations for later problem-solving computational models. In fact, the logical reasoning approach predates the imagery-based approach in Stage 1 [15], but “logically” postdates it in our conceptual chronicle as it requires a higher level of problem representation.

5. Stage 3: Probabilistic Reasoning

Despite being more general, logical reasoning has eluded the uncertainty in perception, e.g., how the symbols in logical reasoning are determined from panel images. Probabilistic reasoning is a natural upgrade of pure logical reasoning, where perceptual uncertainty is modeled through conditional probability distributions given a panel image. This upgrade leads us to the neural-symbolic paradigm, in which a neural perception frontend extracts the distributions over a predefined set of objects from each panel and a symbolic reasoning backend performs probability calculation according to a predefined set of rules. The result of the probability calculation indicates the probability of every possible outcome of the reasoning. Different implementations of frontend and backend have been used to construct probabilistic reasoning models, such as ALANS2, PrAE and NVSA [16, 17, 18]. Although both probabilistic reasoning and logical reasoning work on the basis of predefined sets of objects and rules, probabilistic reasoning has entered the realm of **data-driven approaches** (because the neural perception frontend requires training data), leaving imagery-based and logical reasoning in the realm of **knowledge-based approaches**.

6. Stage 4: Learning Approach

To reduce the reliance on explicit prior knowledge about RPM, the learning approach has been exploited as recent developments in deep learning. This section subdivides the learning approach into four types that together show structural evolution of learning models for solving RPM.

6.1. Learner Type 1

A natural solution to reduce the reliance on the predefined objects and rules is similar to the upgrade from the logical reasoning to the probabilistic reasoning. That is, we can approximate the conditional distribution over the possible rules given the matrix panels. There exist different ways to organize the matrix panels to compute this conditional distribution, which all depend on the **parallelism heuristic** of the matrix structure, i.e., the geometric parallelism implies abstract conceptual parallelism in rows and columns. For example, the Pairwise-ADV and Triple-ADV models [19, 20] compute the rule distribution of binary variables given two and three adjacent panels, respectively. A binary variable indicates whether a specific rule applies to the adjacent panels, for example, whether the objects in the

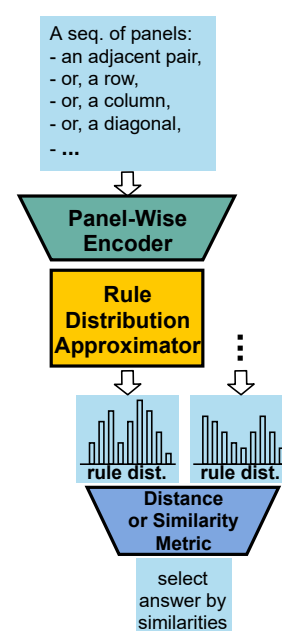


Figure 2: Type 1

panels are of the same color. Another example is DeepIQ [21], in which the rule distribution given two adjacent panels is of an ordered categorical variable (rather than binary), indicating, for example, that the objects in the two panels differ by 3 units in their sizes. According to the parallelism heuristic, the rule distributions in parallel rows or columns should be the same or similar; thus, probability metrics, such as KL-divergence and Euclidean distance, are used to measure the similarity; an answer choice is then chosen when it gives a third row/column whose rule distribution is most similar to the ones of context rows/columns.

These models can be abstracted into Learner Type 1, as shown in Fig. 2. Note that a panel-wise encoder is used to process each input panel individually, which is similar to the perception frontend in probabilistic reasoning. But, unlike the perception frontend, the panel-wise encoder does not necessarily output distributions over the predefined objects. The panel-wise encoder is to represent any latent space as long as the rule distribution can be approximated from this space. After the panel-wise encoder encodes each panel in an input sequence, the embeddings of these panels are further aggregated by the rule distribution approximator into a rule distribution; the distributions of different sequences are finally compared to select the answer choice. The panel-wise encoder and rule distribution approximator can be implemented as proper neural network modules, such as CNN, ResNet and MLP. In practice, these two modules are jointly trained given the ground-truth rule labels of panel sequences.

6.2. Learner Type 2

Unlike the approaches in Stage 1, 2 and 3, Type-1 learners have avoided composing computational streams that explicitly rely on the predefined objects and rules. But it still relies on the ground-truth rule labels and the parallelism heuristic. Therefore, we introduce the Learner Type 2, as shown in Fig. 3, which is free of the reliance. This type converts an RPM into a classification problem, where the class labels are the correctness of choice panels: if only one choice panel is included in the input, it is a binary classification problem; if multiple choice panels are included, it is a multi-class problem.

Readers might have noticed a small difference between Fig. 2 and Fig. 3—the panel-wise encoder has been changed to an encoder (not necessarily panel-wise). As the name indicates, the encoder takes as input multiple panels and may encode relations among these panels into its output. Therefore, the encoder in Type-2 learners is not only responsible for perceptual processing but also conceptual processing.

Conceptual processing in RPM generally involves reasoning about the relations among matrix panels, and thus if one wishes to explicitly separate these two types of processing, one would have a first module (like the panel-wise encoder in Type 1) that attends to each panel individually and a second module (like the rule distribution approximator in Type 1) to aggregate the outputs of the first module. This design choice is an important one for building any computational models for visual abstract reasoning. By changing the name to “encoder”, we implies that Learner Type 2 does not necessarily require an explicit separation of perceptual and conceptual processing.

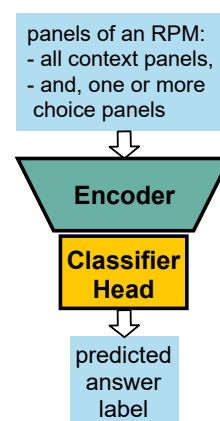


Figure 3: Type 2

Hoshen and Werman [22] implemented a Type-2 model using a CNN encoder and an MLP classifier, and tested it on simple figural series and RPM-like problems. This CNN+MLP model has been constantly used as a baseline in later works. Learner Type 2 can also be implemented in many different ways, such as the Wild-ResNet+MLP model [23] and the ResNet+MLP model [24], respectively representing the binary and multi-class versions of Learner Type 2. Hoshen and Werman [22] also proposed the generative counterpart of the CNN+MLP model, by replacing the MLP classifier with a deconvolutional network to generate the predicted answer panel (no choice panel in the input in this case). We upgrade Type 2 to Type 2+ to include this generative case, as shown in Fig. 4. Interestingly, this type can now be considered as a prototype of Learner Type 4, which we will discuss later. But before Learner Type 4, we are going to take a detour to see how we can reach the same destination differently.

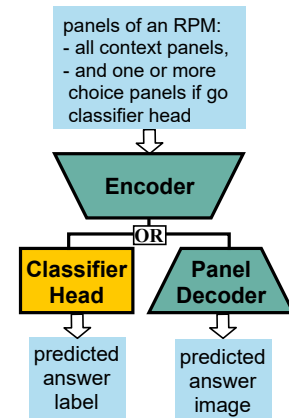


Figure 4: Type 2+

6.3. Learner Type 3

By following the paradigm of image classification, Learner Type 2 eliminates the reliance on the ground-truth rule labels and the parallelism heuristic. But an unnecessary cost in this is that it does not impose separation of perceptual and conceptual processing, which is usually considered beneficial for visual abstract reasoning. This observation leads us to Learner Type 3, as shown in Fig. 5. Note that, given the same input and output, one could certainly consider Learner Type 3 as a special case of Learner Type 2, by regarding everything before the classifier head as a single module. But models based on this more detailed specification generally perform better than the typical models of learner Type 2. Thus, we separate it from Learner Type 2.

After the panel-wise encoder encodes each panel, these panel embeddings go through a combinatorial process, in which subsets of these panel embeddings are selected and fed into next module subset by subset. In Fig. 5, we use two trapezoids of opposite orientations for the panel-wise encoder and this combinatorial process to indicate that the amount of information is compressed and decompressed (i.e., the number of combinations is more than what are combined). As the name “combinatorial heuristics” indicates, Learner Type 3 explicitly relies on some heuristics to take combinations, which include but not limited to the aforementioned parallelism heuristic. Essentially, these heuristics inform the learner of which panel embeddings, together as a group, would manifest a rule. Each such group is individually processed by the same Relation Encoder to produce a relation embedding for this group. At last, all relation embeddings are aggregated for classification.

A typical example of Learner Type 3 is the WReN model [23]. WReN takes as input all context panel plus one choice panel (thus solving binary classification). The panel-wise encoder is a

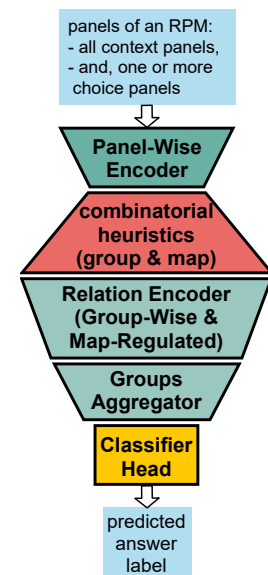


Figure 5: Type 3

small CNN (plus tagging the panel embeddings with one-hot position vectors indicating the panels' positions in the matrix). For combinatorial heuristics, WReN considers all binary rules (i.e., relations between every two panels). Note that WReN does not use the parallelism heuristic, which is commonly used in other models; but the position-tagged panel embedding make this less of a problem, because the relation encoder can trivially determine the non-adjacent panels from position-tags and output a specific rule-embedding for them. The Groups Aggregator in WReN is simply a summation.

Following WReN, a series of models of Learner Type 3 have been created, using different panel-wise encoders, combinatorial heuristics, relation encoders and groups aggregators. For example, LEN [25] considers only ternary rules for combinatorial heuristics, i.e., groups every three panels together for relation encoding, and applies gating variables to each groups in aggregator (unsurprisingly, the experiment results showed that all gating variables except the ones of rows and columns were zeroed); MXGNet [26] also considers ternary rules, uses CNN or R-CNN panel-wise encoder, relies on parallelism heuristic for combinatorial heuristics (instead of gating variables), and employs a graph-learning-based relation encoder that models the 3 panels in a group as a graph and compute the graph embedding as the relation embedding.

Different from the previous Type-3 learners, multi-layer RN [27, 28, 29] extends the relation encoding in WReN into a multi-layer format. This is, the relation embeddings of each group are not aggregated into a single embedding for classification, but into multiple embeddings, which are further fed into another combinatorial module and relation encoder. Therefore, one could visualize multi-layer RN as a Type-3 learner, repeating the middle three modules as many times as needed, during which the combinatorial heuristics are defined according to task specifics. One would expect the higher-order relations (if any) to be detected in this model.

The SRAN model [30] adopts a more complicated encoding scheme by using multiple encoders and multiple relation encoders, where panels of two rows/columns (6 in total) are encoded panel-wise, 3-panels-wise, 6-panels-wise by three different encoders, and the resulting panel-embeddings, 3-panel-embeddings and 6-panel-embeddings are sequentially integrated by three relation encoders into a single rule embedding, representing the rule of these two rows/columns. The encoding scheme of SRAN, though complicated, does not deviate too much from Learner Type 3. But, in stead of using rule embeddings to solve the RPM as an classification problem, SRAN directly uses similarity metrics of rule embeddings to select the answer, as in Learner Type 1, which is also a common practice (just a different way to present the same supervising signal). Thus, it gives us a more complete Learner Type 3+, as shown in Fig. 6.

MRNet [31] is another Type-3 learner using multiple panel encoders and multiple relation encoders, corresponding to three different resolutions defined by different layers' output in a CNN panel-wise encoder. The computational streams for each resolution separately follow the parallelism heuristic, and are aggregated at the end for classification.

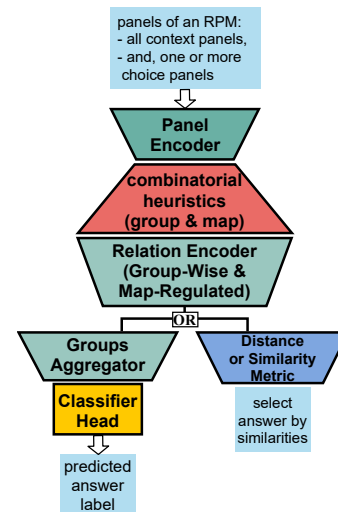


Figure 6: Type 3+

Readers might have noticed the words “group” and “map” in the diagrams of Fig. 5 and 6. By these words, we intend to call attention to a mechanism that permeates everywhere in information processing system for visual abstract reasoning, i.e., which pieces of information should be grouped together and thus to be aggregated later, and which pieces of information should be mapped¹ and thus to be processed equally². These two types of decisions are interdependent on each other; more precisely, they are better to be viewed as two aspects of the same process. These decisions have to be made repeatedly at every level in information processing. Unfortunately, there might not be a centralized or universal solution for this grouping-mapping mechanism. As one can see in these Type-3 learners, they all resort to some specific heuristics, which might not be correct in a general sense of visual abstract reasoning.

6.4. Learner Type 4

When we look back at how learners have evolved from Type-1 to Type-3, we can see a method to circumvent the difficulty of specifying a grouping-mapping mechanism and also preserve the advantages of Learner Type 3, i.e., no reliance on ground-truth rule labels and the separation of perceptual and conceptual processing. Similar to the development from Type 1 to Type 2, we use a single learnable module to replace both the combinatorial heuristics and relation encoder in Learner Type 3, without introducing any heuristic grouping-mapping. This gives us Learner Type 4, as shown in Fig. 7. Since the reasoning module contains no heuristics about grouping-mapping, its output cannot be assumed to represent the relation among certain panels, and thus cannot be processed as in Type 3/3+. Thus, supervising signals are directly applied on this output.

Now the reader might want to look back at Learner Type 2+ and understand why we said it is a prototype of Learner Type 4. Type 4 simply separates the perceptual and conceptual processing by separating the holistic encoder into a panel-wise encoder and a reasoning module. As in Learner Type 2+, the supervising signal is applied in two ways.

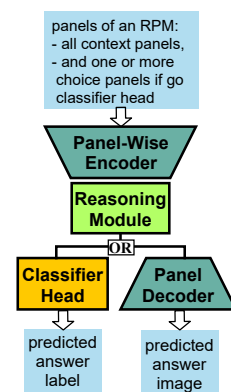


Figure 7: Type 4

Reasoning Kernel 0: CNN

The CNN module has been a basic tool to extract features from raw input, and the extracted features are not only relevant for solving specific downstream tasks, but also representing certain correlations in the input. Solving visual abstract reasoning tasks is also to process correlations among visual inputs. Therefore, CNN would have been the first choice for solving RPM. However, several early works [23, 24, 32] argued that CNN is incapable of solving RPM and thus proposed new models³. Since then, the research has gone into other directions, but having no generally good performance across multiple datasets (like PGM and RAVEN). Ironically,

¹or aligned; we use “map” to resonate with the terminology of structure-mapping theory and analogical reasoning; i.e., if two entities in the base and target domains are mapped, then they are analogous to each other.

²i.e., forcing the analogical relation between them.

³This is also why the CNN+MLP and ResNet+MLP models of Learner Type 2 have been constantly used as baselines—no outstanding performance.

Spratley et al. [33] proposed two Type-4 learners—Rel-Base and Rel-AIR—which are all CNN-based models and perform well on both PGM and RAVEN. After comparing these two models with the previous CNN models, we found that the difference is whether the conceptual and perceptual processing are separated. Taking Rel-Base as an example, its panel-wise encoder is a CNN module and its reasoning module is another CNN module; all the panel embeddings are first stacked together and then convolved with the convolution kernels in the reasoning module. But the baseline CNN-based models do not have this separation. Therefore, we conjecture that the outstanding performance of many non-CNN models is not because they are not using CNN for reasoning, but because they separate perceptual and conceptual processing.

Reasoning Kernel 1: LSTM

A typical Type-4 learner is the CNN+LSTM+MLP model [23]. This model takes as input all context panels and one or more choice panels. Each panel embedding is sequentially processed by an LSTM reasoning module, and the final state of LSTM is fed into an MLP classifier to predict the answer panel. This model is also used as a common baseline in many later works. LSTM has also been combined with other modules: Double-LSTM [34] uses two LSTM modules, which specialize in different rule types and are coordinated by an extra module trained to predict the rule type⁴; ESBN and NTM [35, 36], combining LSTM with external memory modules, can also be used as the reasoning kernel in Learner Type 4.

Reasoning Kernel 2: Self-Attention

Another commonly used reasoning kernel is the self-attention module, which is composed of a multi-head attention and a feed-forward network (with residual connections and normalization). The most typical example of this “reason kernel” is the ARNe model [37]. It extends the Type-3 learner WReN by inserting between the panel-wise encoder and the combinatorial heuristics module a self-attention module. Note that, although ARNe inherits the combinatorial heuristic of WReN, it is no longer a Type-3 learner because the self-attended embeddings no longer represent individual panels. Instead, each self-attended embedding contains information about all the matrix panels, and should better be considered summaries of the whole matrix from different angles. Therefore, the inherited combinatorial heuristics module and the following modules of WReN can be considered similar to other general classifier heads, which are semantically-agnostic of its input, simply aggregating the input and predicting the answer. With hindsight, a reasonable order should have been first testing the self-attention module with a simple classifier head rather than WReN.

A similar example is the HTR model [38], where an R-CNN panel-wise encoder is used to extract all entities in each panel and two self-attention-based sub-modules are used to move the reasoning from entity-level to panel-level and from panel-level to matrix-level. The first sub-module takes as input the entity embeddings in a single panel and sums up the self-attended entity embeddings as the panel embedding. Unlike ARNe and WReN solving RPM as binary classification, HTR solves it as multi-classification. Therefore, the output of the second sub-

⁴This reliance on ground-truth rule labels slightly deviates from our definition of Learner Type 4.

module contains 8 embeddings corresponding to the 8 choice panels. These 8 embeddings are fed into a contrastive classifier head [16] (see the appendix) to predict the answer label.

Reasoning Kernel 3: Multi-Head Relation Detector

The last reasoning kernel is closely related to the Relation Encoder of Type 3. Recall that, in Type 3, the combinatorial heuristics module groups the panel embeddings into multiple groups, and every group is processed by a singleton relation encoder to obtain a rule embedding for this group. Although this relation encoder has 1-in and 1-out, it is responsible for recognizing and encoding whichever rule the input group has. Recall that, by moving from Type-3 to Type-4, we intended to eliminate the reliance on combinatorial heuristics. A natural solution could be an “all-in-all-out” relation encoder, which takes as input all the panel embeddings of a matrix (no grouping) and outputs all the possible rules. This is analogous to image classification versus object detection. Particularly, the new relation encoder can have multiple output heads, where later supervising pressure can be applied to force each head to represent a specific rule. Therefore, we refer to this reasoning kernel as multi-head relation detector. This kernel is underrepresented and we found only one example using this kernel—the SCL model [39].

These four learner types have addressed the structural aspect of learning approach. Another equally important issue is how we train these models. Besides the standard supervised learning of the correct answers of RPM, many other techniques have been attempted to take advantage of the enriched design of RPM. Due to the page limit, We refer our readers to Section B in the appendix for more details.

7. Concluding Remarks

In this paper, we have replayed the technical development of computational solutions to RPM. The food for thought we would love to share with our readers is that, by comparing the models in different stages, we found that the technical development, on one hand, always explores new methods to solve RPM, but, on the other hand, inevitably revisits the old ideas again and again. Therefore, the most recent models are not necessarily superior to the traditional ones in nature, for example, the imagery-based approach might trigger the next cycle of technical development in future research.

References

- [1] J. C. Raven, *Mental Tests Used in Genetic Studies: The Performance of Related Individuals on Tests Mainly Educative and Mainly Reproductive*, Master’s thesis, University of London, London, United Kingdom, 1936.
- [2] J. Raven, The raven’s progressive matrices: change and stability over culture and time, *Cognitive psychology* 41 (2000) 1–48.
- [3] R. E. Snow, P. C. Kyllonen, B. Marshalek, The topography of ability and learning correlations, *Advances in the psychology of human intelligence* 2 (1984).

- [4] M. Kunda, K. McGreggor, A. K. Goel, A computational model for solving problems from the raven’s progressive matrices intelligence test using iconic visual representations, *Cognitive Systems Research* 22 (2013) 47–66.
- [5] J. Hernández-Orallo, F. Martínez-Plumed, U. Schmid, M. Siebers, D. L. Dowe, Computer models solving intelligence test problems: Progress and implications, *Artificial Intelligence* 230 (2016) 74–107.
- [6] M. Meo, M. J. Roberts, F. S. Marucci, Element salience as a predictor of item difficulty for raven’s progressive matrices, *Intelligence* 35 (2007) 359–368.
- [7] S. M. Kosslyn, W. L. Thompson, G. Ganis, *The case for mental imagery*, 1st. ed., Oxford University Press, New York, NY, 2006.
- [8] M. Kunda, K. McGreggor, A. Goel, Addressing the raven’s progressive matrices test of “general” intelligence, in: *AAAI Fall Symposium Series*, 2009, pp. 22–27.
- [9] M. Kunda, Ai, visual imagery, and a case study on the challenges posed by human intelligence tests, *Proceedings of the National Academy of Sciences* 117 (2020) 29390–29397.
- [10] Y. Yang, K. McGreggor, M. Kunda, Not quite any way you slice it: How different analogical constructions affect raven’s matrices performance, in: *Proceedings of the Eighth Annual Conference on Advances in Cognitive Systems (ACS)*, 2020.
- [11] M. Kunda, Visual problem solving in autism, psychometrics, and AI: The case of the Raven’s progressive matrices intelligence test, Ph.D. thesis, Georgia Institute of Technology, Atlanta, GA, 2013.
- [12] P. A. Carpenter, M. A. Just, P. Shell, What one intelligence test measures: a theoretical account of the processing in the raven progressive matrices test, *Psychological review* 97 (1990) 404–431.
- [13] A. Lovett, E. Tomai, K. Forbus, J. Usher, Solving geometric analogy problems through two-stage analogical mapping, *Cognitive science* 33 (2009) 1192–1231.
- [14] C. Strannegård, S. Cirillo, V. Ström, An anthropomorphic method for progressive matrix problems, *Cognitive Systems Research* 22 (2013) 35–46.
- [15] T. G. Evans, A heuristic program to solve geometric-analogy problems, in: *Proceedings of the Spring Joint Computer Conference*, ACM, New York, NY, 1964, pp. 327–338.
- [16] C. Zhang, S. Xie, B. Jia, Y. Zhu, Y. N. Wu, S.-C. Zhu, Learning algebraic representation for abstract spatial-temporal reasoning (2020). Unpublished work.
- [17] C. Zhang, B. Jia, S.-C. Zhu, Y. Zhu, Abstract spatial-temporal reasoning via probabilistic abduction and execution, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9736–9746.
- [18] M. Hersche, M. Zeqiri, L. Benini, A. Sebastian, A. Rahimi, A neuro-vector-symbolic architecture for solving raven’s progressive matrices, *arXiv preprint arXiv:2203.04571* (2022).
- [19] C. S. Mekik, R. Sun, D. Y. Dai, Deep learning of raven’s matrices, in: *Proceedings of the fifth Annual Conference on Advances in Cognitive Systems*, 2017.
- [20] C. S. Mekik, R. Sun, D. Y. Dai, Similarity-based reasoning, raven’s matrices, and general intelligence, in: *Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI’18*, AAAI Press, 2018, p. 1576–1582.
- [21] J. Mańdziuk, A. Zychowski, DeepIQ: A human-inspired ai system for solving iq test problems, in: *International Joint Conference on Neural Networks*, IEEE, 2019.

- [22] D. Hoshen, M. Werman, IQ of neural networks, arXiv preprint arXiv:1710.01692 (2017).
- [23] D. Barrett, F. Hill, A. Santoro, A. Morcos, T. Lillicrap, Measuring abstract reasoning in neural networks, in: *International Conference on Machine Learning*, PMLR, 2018, pp. 511–520.
- [24] C. Zhang, F. Gao, B. Jia, Y. Zhu, S.-C. Zhu, Raven: A dataset for relational and analogical visual reasoning, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5317–5327.
- [25] K. Zheng, Z.-J. Zha, W. Wei, Abstract reasoning with distracting features, *Advances in Neural Information Processing Systems* 32 (2019) 5842–5853.
- [26] D. Wang, M. Jamnik, P. Lio, Abstract diagrammatic reasoning with multiplex graph networks, arXiv preprint arXiv:2006.11197 (2020).
- [27] M. Jahrens, T. Martinetz, Solving raven’s progressive matrices with multi-layer relation networks, in: *International Joint Conference on Neural Networks*, IEEE, 2020.
- [28] M. Jahrens, T. Martinetz, Multi-layer relation networks for relational reasoning, in: *Proceedings of the 2nd International Conference on Applications of Intelligent Systems*, 2019, pp. 1–5.
- [29] M. Jahrens, T. Martinetz, Multi-layer relation networks, arXiv preprint arXiv:1811.01838 (2018).
- [30] S. Hu, Y. Ma, X. Liu, Y. Wei, S. Bai, Stratified rule-aware network for abstract visual reasoning, in: *Proceedings of the AAI Conference on Artificial Intelligence*, volume 35, 2021, pp. 1567–1574.
- [31] Y. Benny, N. Pekar, L. Wolf, Scale-localized abstract reasoning, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12557–12565.
- [32] C. Zhang, B. Jia, F. Gao, Y. Zhu, H. Lu, S.-C. Zhu, Learning perceptual inference by contrasting, arXiv preprint arXiv:1912.00086 (2019).
- [33] S. Spratley, K. Ehinger, T. Miller, A closer look at generalisation in raven, in: *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII* 16, Springer, 2020, pp. 601–616.
- [34] A. A. Sekh, D. P. Dogra, S. Kar, P. P. Roy, D. K. Prasad, Can we automate diagrammatic reasoning?, *Pattern Recognition* 106 (2020).
- [35] I. Sinha, T. W. Webb, J. D. Cohen, A memory-augmented neural network model of abstract rule learning, arXiv preprint arXiv:2012.07172 (2020).
- [36] T. W. Webb, I. Sinha, J. Cohen, Emergent symbols through binding in external memory, in: *International Conference on Learning Representations*, 2020.
- [37] L. Hahne, T. Lüddecke, F. Wörgötter, D. Kappel, Attention on abstract visual reasoning, arXiv preprint arXiv:1911.05990 (2019).
- [38] J. An, S. Cho, Hierarchical transformer encoder with structured representation for abstract reasoning, *IEEE Access* 8 (2020) 200229–200236.
- [39] Y. Wu, H. Dong, R. Grosse, J. Ba, The scattering compositional learner: Discovering objects, attributes, relationships in analogical reasoning, arXiv preprint arXiv:2007.04212 (2021).
- [40] H. R. Burke, Raven’s progressive matrices: A review and critical evaluation, *The Journal of Genetic Psychology* 93 (1958) 199–228.
- [41] L. E. Matzen, Z. O. Benz, K. R. Dixon, J. Posey, J. K. Kroger, A. E. Speed, Recreating raven’s: Software for systematically generating large numbers of raven-like matrix problems with

- normed properties, *Behavior research methods* 42 (2010) 525–541.
- [42] S. van Steenkiste, F. Locatello, J. Schmidhuber, O. Bachem, Are disentangled representations helpful for abstract visual reasoning?, *arXiv preprint arXiv:1905.12506* (2019).
- [43] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, A. Lerchner, beta-VAE: Learning basic visual concepts with a constrained variational framework, in: *International Conference on Learning Representations*, 2017.
- [44] H. Kim, A. Mnih, Disentangling by factorising, in: *International Conference on Machine Learning*, PMLR, 2018, pp. 2649–2658.
- [45] Y. Yang, D. Sanyal, J. Michelson, J. Ainooson, M. Kunda, Automatic item generation of figural analogy problems: A review and outlook, in: *The Ninth Annual Conference on Advances in Cognitive Systems*, 2022.
- [46] R. Primi, Complexity of geometric inductive reasoning tasks: Contribution to the understanding of fluid intelligence, *Intelligence* 30 (2001) 41–70.
- [47] N. Pekar, Y. Benny, L. Wolf, Generating correct answers for progressive matrices intelligence tests, *Advances in Neural Information Processing Systems* 33 (2020) 7390–7400.
- [48] Y. Kim, J. Shin, E. Yang, S. J. Hwang, Few-shot visual reasoning with meta-analogical contrastive learning, *Advances in Neural Information Processing Systems* 33 (2020) 16846–16856.
- [49] M. Małkiński, J. Mańdziuk, Multi-label contrastive learning for abstract visual reasoning, *arXiv preprint arXiv:2012.01944* (2020).
- [50] X. Steenbrugge, S. Leroux, T. Verbelen, B. Dhoedt, Improving generalization for abstract reasoning tasks using disentangled feature representations, *arXiv preprint arXiv:1811.04784* (2018).
- [51] S. Eslami, N. Heess, T. Weber, Y. Tassa, D. Szepesvari, G. E. Hinton, et al., Attend, infer, repeat: Fast scene understanding with generative models, *Advances in Neural Information Processing Systems* 29 (2016).
- [52] T. Zhuo, M. Kankanhalli, Solving raven’s progressive matrices with neural networks, *arXiv preprint arXiv:2002.01646* (2020).
- [53] N. Q. W. Kiat, D. Wang, M. Jamnik, Pairwise relations discriminator for unsupervised raven’s progressive matrices, *arXiv preprint arXiv:2011.01306* (2020).
- [54] F. Hill, A. Santoro, D. G. Barrett, A. S. Morcos, T. Lillicrap, Learning to make analogies by contrasting abstract relational structure, in: *International Conference on Learning Representations*, 2019.
- [55] T. Zhuo, M. Kankanhalli, Effective abstract reasoning with dual-contrast network, in: *International Conference on Learning Representations*, 2021.
- [56] T. L. Hayes, C. Kanan, Selective replay enhances learning in online continual analogical reasoning, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3502–3512.
- [57] K. McGregor, M. Kunda, A. Goel, Fractals and ravens, *Artificial Intelligence* 215 (2014) 1–23.
- [58] N. Rahaman, M. W. Gondal, S. Joshi, P. Gehler, Y. Bengio, F. Locatello, B. Schölkopf, Dynamic inference with neural interpreters, *Advances in Neural Information Processing Systems* 34 (2021).
- [59] S. Yu, S. Mo, S. Ahn, J. Shin, Abstract reasoning via logic-guided generation, *arXiv preprint*

arXiv:2107.10493 (2021).

- [60] T. Hua, M. Kunda, Modeling gestalt visual reasoning on the raven’s progressive matrices intelligence test using generative image inpainting techniques, in: Proceedings of the Eighth Annual Conference on Advances in Cognitive Systems, 2020.
- [61] S. Shegheva, A computational model for solving raven’s progressive matrices intelligence test, Master’s thesis, Georgia Institute of Technology, Atlanta, GA, 2018.

A. Problem/Data Sets

This section introduces three original RPM problem sets, eight RPM-like datasets, which are automatically generated based on the original ones, and two other relevant problem types.

The original RPM test has three versions: Standard Progressive Matrices (SPM), Colored Progressive Matrices (CPM), and Advanced Progressive Matrices (APM). SPM was first published in 1930s and contains 60 items, evenly divided into 5 subsets (with 12 items each) according to variation patterns and item difficulty. SPM has been the most widely studied version, but its discriminative power for the high-ability and low-ability groups is relatively weak. Therefore, APM and CPM were constructed for the high-ability and low-ability groups in 1940s by adapting SPM [40]. Particularly, the first two subsets of SPM formed the basis of CPM with an extra transitional subset inserted in between, and thus CPM contains 36 items; the last three subsets of SPM formed the basis of APM with an extra transitional subset placed before them, and thus APM contains 48 items. The most recent edition of RPM has gone through many updates as more and more normative data are available, and parallel forms have also been developed.

Most of the automatically-generated RPM-like datasets can be traced back to the work of Carpenter et al. [12], who enumerated five variation rules (or, the relation among panels) in APM, which could be mapped into two more generalized categories: (a) pixel-level algebraic rules and (b) object-level algebraic rules. Objects here refer to pixels that are grouped together according to some contextual information, and algebraic rules refer to algebraic operations that are defined on a set, for example, arithmetic operations on pixel RGB values or object numbers, and logical operations on pixels or objects. By applying all the five rules and specifying object types, many RPM-like datasets of different sizes were automatically generated: Sandia [41] (840 items, generator available), RAVEN/I-RAVEN/RAVEN-FAIR (70000 items, generator available) [24, 30, 31] and PGM [23] (1.2M items). These 5 datasets all consist of 3×3 RPMs and are mostly used in computational studies. Datasets based on reduced dimensions or rule sets were also used for special purposes. For example, van Steenkiste et al. [42] constructed two datasets based on the object types in dSprites [43] and 3dshapes [44], and Sinha et al. [35] constructed a dataset based on the Unicode images.

The reader might have a question about object type—how object types are determined from the contextual information. This question is closely related to perceptual organization in visual abstract reasoning, which has been less studied on the computational side of RPM [45] but thoroughly studied in cognitive psychology [46], which is beyond the scope of this paper.

Another type of RPM-like problem—geometric analogy problems (GAP)—features the analogical aspect of RPM items, i.e., two rows or columns of a matrix form an analogy. This type of problems is represented by 20 GAPs (find images in [13]) published in the 1942 edition of

the Psychological Test for College Freshmen of the American Council on Education. Each GAP is explicitly organized in a proportional analogy $A:B::C:D$, i.e., A is to B as C is to D , which resembles a 2×2 RPM in Fig. 1a. A subtle difference is that RPM can have algebraic relations in both horizontal and vertical directions while GAP, if organized into a matrix, has algebraic relations in one direction and the relations in the other direction are determined analogically (i.e., forming a meaningful analogy). Another type of RPM-like problem—figural series [22, 34]—features the inductive nature of RPM item, which follows the format of the number series problems in many ability tests.

B. Stage 4.5: Training

We have addressed the structural aspect of the learning approach in Stage 4. In this section, we discuss how these learning models are trained. All the learners in Stage 4 can be trained through the supervising signal of the correct answer of RPM as in most supervised learning tasks. But different from them, the special design of RPM provides more information that can be used in training. In this section, we will focus on how the learning models are trained differently given this enrichment of useful information.

B.1. Auxiliary Training

For the learners of Type 2, 3 and 4, an extra classifier head can be attached to exactly where the existing classifier head is attached to predict the meta-target of the input RPM, which is a multi-hot vector indicating rules and objects in this RPM. These meta-targets are available in automatically-generated datasets, such as PGM and RAVEN. Therefore, the learner can be trained on the answer labels and meta-targets simultaneously. And the training on meta-targets is referred to as auxiliary training.

Intuitively, this extra supervising signal can boost the accuracy of the answer-label classifier head. Auxiliary training was first tried by the WReN model on the PGM dataset and indeed showed a approximately 10% boost (in IID generalization regime). The contribution of auxiliary training was verified by a high correlation between the two classifier heads [23]. Similar observations on PGM were found in other studies [47, 37]. For example, the ARNe model would not even converge without auxiliary training.

However, the effect of auxiliary training is inconclusive. For example, Benny et al. [31] showed that auxiliary training on PGM could only increase the accuracy of 1-rule problems but decrease the accuracy of multi-rule problems (and thus degrade the overall accuracy). Besides depending on rules, the effect also differs between datasets. It has been reported that the auxiliary training would generally decrease the performance on the RAVEN dataset [24, 32, 25, 26], with one exception [48], which used a special contrastive loss and will be discussed later. However, Małkiński and Mańdziuk [49] also showed contradictory results that, when the meta-target is encoded in a sparse manner (the above experiments are all dense-encoding), the auxiliary training can increase the performance on RAVEN. Therefore, we can only say that the effect of auxiliary training is jointly determined by model, loss function, dataset and meta-target encoding.

B.2. Disentangled and Generative Representations

The probabilistic reasoning approaches in Stage 3 have been frequently using neural networks, such as autoencoder and CNN, in their perception frontends to construct representations of panel image with explicit symbolic meanings. And, as we mentioned in Learner Type 1, these symbolic meanings of encoders' output were not guaranteed. Therefore, instead of representations with symbolic meanings, disentangled and generative representations are used in Stage 4. For example, the Type-1 learner, DeepIQ [21], uses the encoder of an variational auto-encoder (VAE), which is pretrained on panel images of the Sandia dataset and kept frozen when the rule approximator is trained subsequently.

Several advantages of disentangled and generative representations in RPM have been reported, such as data efficiency [42], robustness to distracting attributes [25] and better OOD generalization [50]. Disentangled and generative representations of panel images are usually obtained through VAE or its variants. For Type-3 learners, β -VAE, FactorVAE, β -TCVAE and DIP-VAE were pretrained on panel images and the frozen encoders were combined with WReN [50, 42]; a reduced version of MRNet was jointly trained with a VAE to simultaneously predict the answer label and generate the answer image (thus we call it generative-MRNet) [47]. For Type-4 learners, the VAE is usually jointly trained with the reasoning module, for example, the aforementioned ESN model [35], and the LoGe model [18] using vq-VAE as its encoder and decoder. Another special example of Type-4 is the Rel-AIR model [33] integrating into its encoder an Attend-Infer-Repeat model [51], which can be thought of as iterative VAE.

B.3. Contrastive learning and Manipulating Data

In addition to supervised learning, contrastive learning has also been popular for solving RPM. We need to point out that the techniques of contrastive learning have been highly adapted to fully employ the structural and analogical characteristics of RPM and thus might not strictly follow the paradigms of contrastive learning. Particularly, the special format of RPM provides more options to manipulate data, such as decomposing matrices into rows and columns and regrouping them, and regrouping answer choices and even RPM problems, and various supervising signals can be applied to contrast the decomposed and regrouped data. In this subsection, we discuss all the contrasting mechanisms for solving RPM.

Intra-Item Contrasting: Row/Column Contrasting

The minimum structure that can be contrasted is rows/columns of matrices. This type of contrasting was first attempted in the MCPT model [52], where 8 choice panels are inserted into the matrix to obtain 10 rows/columns (2 context rows/columns and 8 choice rows/columns). The context rows/columns are assigned pseudo-label 1 and choice rows/columns are assigned pseudo-label 0; and this newly constructed pseudo-dataset of row/columns is learned by a Type-2 learner, assuming that only one "mis-assigned" pseudo-label of the correct choice will not affect the final result. To solve RPM, the choice row/column with the highest predicted output (between 0 and 1) is selected.

The intuition behind MCPT is to capture any characteristic that distinguishes between the correct and incorrect choices when they are embedded into the third row/column. In particular, it

checks whether the third row/column has a meaningful variation that is similar to **any** context row/column in the dataset. The PRD model [53] enhanced this type of single-row/column contrasting by including the parallelism heuristic. As in standard contrastive learning, positive and negative pairs are constructed from rows/columns, where the first two rows/columns in an RPM matrix make a positive pair. The negative pair could be constructed in multiple ways, such as rows/columns from different RPMs, randomly-shuffled rows/columns of the same RPM, or filling the third row/column with a random non-choice panel. In PRD, a Type-2 learner is used to learn the difference (or a similarity metric) between two paired rows/columns. To solve an RPM, the choice row/column that is most similar to the first two rows/columns is selected. Compared to the single-row/column contrasting, the double-row/column contrasting is more common, which could be found in many other works. For example, the aforementioned generative-MRNet [47] contrasts the choice rows/columns formed by the generated answer panel with the choice rows/columns formed by the given choice panels.

The rationale of moving from single-row/column to double-row/column contrasting was also exemplified by the LABC training/testing regime [54], which makes it more accurate and complete through the meta-targets used in auxiliary training. Different from the single-row/column and double-row/column contrasting, where the effect of contrasting is applied through extra contrastive loss functions, LABC, as a training/testing regime, requires models to learn adapted datasets, which will force the model to contrast the rows/columns. In particular, an RPM is adapted by muting some digits of its meta-target and regenerating the incorrect choice panels (based on given context panels). Since meta-targets use multi-hots to indicate the rules and geometric objects that are used to generate RPM items, the newly-generated choice panels are partially correct. This way, the model will have to compare such choice rows/columns to the context rows/columns to find the correct answer, instead of only seeking meaningful variations in the choice row/column as in the single-row/column. LABC makes this idea more systematic by introducing the concepts of semantically and perceptually plausible choice corresponding to muting different subsets of meta-target digits and using distracting objects and rules.

Intra-Item Contrasting: Matrix Contrasting

Instead of contrasting rows/columns, we can also contrasting the matrices completed by each choice panel. This is essentially contrasting the choices relative to the context panels. A Type-2 learner, CoPINet [32], is the first model performing such contrasting. The contrasting in CoPINet is two-fold—contrastive representation and contrastive loss. First, the embeddings of the matrices completed by each choice panel are aggregated into a “central” embedding and their differences to the “central” embedding are used in the following processing. Second, given the interweaving of these matrix embeddings, it naturally leads to a contrastive loss function that incorporates both correctly completed and incorrectly completed matrices and increases the gap between their predicted values. This contrasting could be easily embedded into models of parallel computation streams, for example, the aforementioned HTR model [38].

We need to point out that row/column contrasting and matrix contrasting are not exclusive. For example, the DCNet model [55] first uses row/col contrasting to compute the matrix embeddings and then uses the matrix contrasting to predict the answer.

Inter-Item Contrasting: Single-Label Contrasting

The above contrasting has been restricted within a single RPM item. Now, we describe the inter-item contrasting. The ACL and Meta-ACL [48] are the first two inter-item contrasting models. The relation between ACL and Meta-ACL is similar to that between single-row/column and double-row/column contrasting. Given an RPM, let X be its incomplete context matrix (regarding the missing panel as an empty image), X_i an incomplete matrix obtained by replacing the i -th panel with a white-noise image, and X' an incomplete matrix obtained by randomly reordering the panels of X . The ACL model contrasts the positive pair (X, X_i) with the negative pair (X, X') . The Meta-ACL resorts to meta-targets to compose positive and negative pairs. In particular, two incomplete matrices of two distinct items of the same meta-target form a positive pair (X_S, X_T) , and the corresponding negative pair is (X_S, X'_S) . In both ACL and Meta-ACL, the contrasting effect is applied through an extra standard contrastive loss function.

The MLCL model [49] formalizes the idea of Meta-ACL in a multi-label setting by regarding multi-hot meta-targets as multi-labels. Therefore, instead of requiring positive pairs to have exactly the same meta-targets, MLCL regards pairs of intersecting meta-targets as positive pairs (to some degree). Different from Meta-ACL, the completed matrices are used. In particular, the correctly completed matrices are used for inter-item contrasting, and the intra-item contrasting between the correctly completed matrix and its corresponding incorrectly completed matrices is performed as in CoPINet. These two types of contrasting losses are jointly optimized.

Other Dimensions of Manipulating Data

Besides contrasting, there are also other dimensions of manipulating data. For example, the FRAR model [25] utilizes a reinforcement learning teacher model to select items from an RPM item back to train a student model, where the items in the bank are characterized by their meta-targets and the reward is the increase in accuracy of the student model. The models solving RPM have also been examined in the setting of continual learning. For example, RAVEN can be divided into 7 batches according to its 7 spatial configurations and the models are trained with different methods to mitigate forgetting when sequentially learning the 7 batches in different orders [56].

C. A Summary of Computational Models/Methods

Table 1
Computational Models/Methods for Solving RPM

Acronym	Full Name	Article
ASTI	Affine and Set Transformation Induction	[11]
ASTI+	Affine and Set Transformation Induction Plus	[10]
-	Fractal Model	[57]
-	FAIRMAN	[12]
-	BETTERMAN	[12]
CogSketch+SME	CogSketch and Structural Mapping Engine	[13]
-	Anthropomorphic Solver	[14]
-	ANALOGY	[15]
ALANS2	ALgebra-Aware Neuro-Semi-Symbolic	[16]
PrAE	Probabilistic Abduction and Execution	[17]
NVSA	Neural-Vector-Symbolic Architecture	[18]
Pairwise-ADV [*]	Pairwise Attribute Difference Vector	[19]
Triple-ADV [*]	Triple Attribute Difference Vector	[20]
DeepIQ	Deep IQ	[21]
CNN+MLP	-	[22]
CNN+decoder [*]	-	[22]
ResNet+MLP	-	[23]
Wild-ResNet+MLP	-	[23]
WReN	Wild Relation Network	[23]
LEN	Logic Embedding Network	[25]
MXGNet	Multiplex Graph Network	[26]
multi-layer RN	multi-layer Relation Network	[29, 28, 27]
SRAN	Stratified Rule-Aware Network	[30]
MRNet	Multi-Scale Relation Network	[31]
Rel-Base	Basic Relational Reasoning	[33]
Rel-AIR	Attend-Infer-Repeat Relational Reasoning	[33]
CNN+LSTM+MLP	-	[23]
Double-LSTM	-	[34]
ESBN	Emergent Symbol Binding Network	[35]
NTM	Neural Turing Machine	[35]
ARNe	Attention Relation Network	[37]
HTR [*]	Hierarchical Transformer Reasoning	[38]
NI	Neural Interpreter	[58]
SCL	Scattering Compositional Learner	[39]
4 VAE+WReN	4 variants of VAE plus WReN	[50, 42]
generative-MRNet [*]	-	[47]
LoGe	Logic-Guided Generation	[59]
MCPT	Multi-label Classification with Pseudo Target	[52]
PRD	Pairwise Relations Discriminator	[53]
LABC	Learning Analogies by Contrasting	[54]
CoPINet	Contrastive Perceptual Inference Network	[32]
DCNet	Dual-Contrast Network	[55]
ACL	Analogical Contrastive Learning	[48]
Meta-ACL	Meta Analogical Contrastive Learning	[48]
MLCL	Multi-Label Contrastive Learning	[49]
FRAR	Feature Robust Abstract Reasoning	[25]
-	Continual Learning	[56]
DRT	Dynamic Residual Tree	[24]
-	GAN	[60]
-	Structural Affinity Method	[61]
PGM	Procedurally Generated Matrices	[23]
RAVEN	Relational and Analogical Visual Reasoning	[24]

^{*} No Acronym was given in the original article. So, we created a name to clearly refer to it in this paper.