VISUAL ABSTRACT REASONING IN COMPUTATIONAL IMAGERY

By

Yuan Yang

Dissertation
Submitted to the Faculty of the
Graduate School of Vanderbilt University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY
in

Computer Science

May 10, 2024

Nashville, Tennessee

Approved:

Maithilee Kunda, Ph.D.

Douglas H. Fisher, Ph.D.

Matthew S. Berger, Ph.D.

Soheil Kolouri, Ph.D.

Bin Li, Ph.D.

Copyright © 2024 Yuan Yang
All Rights Reserved

To my wife, my daughter, my mother, my father, my grandmother, my grandfather, and my cat Xigua.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Dr. Maithilee Kunda, for the guidance, encouragement, and inspiration that she gave me selflessly in the last six years. She sets an example for me to be a computer scientist and AI researcher. I remember that in our first meeting, she told that she was here (at Vanderbilt) to answer big questions. This has been inspiring me in all my research. She also taught me that paying attention to details and patience were crucial for scientific research. In everyday life, she is a nice person to students, colleagues, and even strangers, always caring about others.

I would also like to thank my dissertation committee members, Prof. Fisher, Prof. Berger, Prof. Kolouri, and Prof. Li, and all the instructors of the classes I took in the computer science department and the math department at Vanderbilt. I am lucky and grateful to meet these people in my life. They have shown me the highest level of education quality that I have ever experienced.

I would also like to thank my master advisor, Prof. Xiaoan Li, who helped me start my research career and gave me adequate research freedom in my master study. He helped shape my preference for interdisciplinary AI research that deeply influenced my dissertation research. I would also like to thank Ms. Weina Xu, my middle school instructor, who taught me to be a responsible person for myself and others.

The past and current members of the Lab for Artificial Intelligence and Visual Analogical Systems: Deepayan, Tengyu, Joel, James, Shiyao, Effat, Caoimhe, and many others, created a perfect and friendly atmosphere for living and studying, in which I stayed happily for six years.

Last but not least, without the funding support from Vanderbilt University and the NSF grants: NSF Award #1730044, NSF Award #2033413, NSF Award #1936070, and NSF Award #2034013, I would not be able to finish my dissertation research.

TABLE OF CONTENTS

Page

LIST OF TABLES . . . . **viii**

LIST OF FIGURES . . . . **x**

1 AI's AI Research Proposal **1**

2 Research Questions **8**

- 2.1 Problem Statement 8
- 2.2 Research Question 1: From Psychometrics to Artificial Intelligence 11
- 2.3 Research Question 2: The Interplay between Perceptual and Conceptual Processing in Visual Abstract Reasoning 11
- 2.4 Research Question 3: Computational Imagery 12
- 2.5 Research Question 4: From Generative to Productive 13
- 2.6 Summary 13

3 From Psychometrics to Artificial Intelligence: An Introduction to Raven's Progressive Matrices **15**

- 3.1 Introduction 15
- 3.2 Raven's Progressive Matrices 17
  - 3.2.1 What RPM Measures? 18
  - 3.2.2 A Brief History of RPM 20
  - 3.2.3 What RPM measures exactly? 23
- 3.3 RPM-Like Tasks 26
  - 3.3.1 RPM-like Tasks in Intelligence Tests 27
    - 3.3.1.1 Matrix Reasoning 28
    - 3.3.1.2 Figure Series 29
    - 3.3.1.3 Analogy Making 30
    - 3.3.1.4 Contrastive Classification 31
    - 3.3.1.5 Open Classification 33
    - 3.3.1.6 Summary 34
  - 3.3.2 Algorithmic Item Generation of Matrix Reasoning 35
    - 3.3.2.1 Algorithmically Generating Matrix Reasoning Items for Human Intelligence Testing 36
      - 3.3.2.1.1 Rule-Based Item Construction—Human-Based AIG 36
      - 3.3.2.1.2 Cognitive Design System Approach—Combination of Cognitive Modeling and Psychometrics 39
      - 3.3.2.1.3 MatrixDeveloper—4-by-4 Matrices 40
      - 3.3.2.1.4 GeomGen—Perceptual Organization 40
      - 3.3.2.1.5 Sandia Matrix Generation Software—High-Fidelity SPM Generator 42
      - 3.3.2.1.6 CSP Generator—First-Order Logic Representation 43
      - 3.3.2.1.7 IMak Package—Open Source 45
    - 3.3.2.2 Algorithmically Generating Matrix Reasoning Items for AI Testing . . . 47
      - 3.3.2.2.1 Procedurally Generated Matrices 47
      - 3.3.2.2.2 Relational and Analogical Visual rEasoNing 48

3.3.2.2.3 Context-Blind Issue 52
3.3.2.3 Summary 54
3.4 Computational Models for Solving RPM and RPM-Like Tasks 55
3.4.1 Stage 1: Imagery-Based Approach 58
3.4.2 Stage 2: Logical Reasoning 60
3.4.2.1 Rule Matching 61
3.4.2.2 Rule Induction 62
3.4.3 Stage 2.5: Neuro-Symbolic Reasoning 63
3.4.4 Stage 3: Learning Approach 63
3.4.4.1 Type 1 64
3.4.4.2 Type 2 65
3.4.4.3 Type 3 67
3.4.4.4 Type 4 71
3.4.4.4.1 Reasoning Kernel 1: CNN 73
3.4.4.4.2 Reasoning Kernel 2: LSTM 73
3.4.4.4.3 Reasoning Kernel 3: Self-Attention 74
3.4.4.4.4 Reasoning Kernel 4: Multi-Head Rule Detector 75
3.4.5 Stage 4: Data Manipulation 75
3.4.5.1 Auxiliary Training 76
3.4.5.2 Disentangled and Generative Representations 76
3.4.5.3 Contrastive learning and Manipulating Data 77
3.4.5.3.1 Intra-Item Contrasting: Row/Column Contrasting 77
3.4.5.3.2 Intra-Item Contrasting: Matrix Contrasting 78
3.4.5.3.3 Inter-Item Contrasting: Single-Label Contrasting 79
3.4.5.4 Other Dimensions of Manipulating Data 79
3.4.6 Summary 80
3.5 Discussion 80
3.5.1 The Validity of AI Testing 81
3.5.2 The Implication of Human Intelligence for Building AI systems 85

4 The Interplay between Perceptual and Conceptual Processing in Visual Abstract Reasoning 87

4.1 Introduction 87
4.2 Two Paradigms of Visual Abstract Reasoning 88
4.3 From the Paradigm to Neural Architecture 93
4.3.1 CPCNet Neural Architecture 95
4.3.2 CPCNet Model for Solving RPM-Like Datasets 96
4.4 Implementation of CPCNet and Training 97
4.5 Experiments of CPCNet on the RAVEN Family 99
4.6 Ablation Studies of CPCNet on RAVEN 101
4.6.1 Varying the number of CPC layers 101
4.6.2 Ablating the consistency computation 102
4.7 The Gap Between Grid and Non-Grid Configurations of RAVEN 103
4.7.1 What Possibly Causes the Gap 104
4.7.2 AB-RAVEN 107
4.8 Experiments of CPCNet on PGM 113
4.8.1 Trivial Generalization Regimes 117
4.8.2 Nontrivial Generalization Regimes 118
4.9 An Extended View of CPCNet : Deep Non-Monotonic Reasoning 119
4.10 Concluding Remarks 121

5 Computational Imagery 122

5.1 Introduction 122
5.2 ASTI+: an Imagery-Based Model for Solving RPM 123

5.2.1 Model Details 124
5.2.1.1 Input and Output Representations 124
5.2.1.2 Similarity Metrics 125
5.2.1.3 Transformations 126
5.2.1.4 Analogies 127
5.2.1.5 General Integration Strategy 129
5.2.1.6 Specific Integration Strategies: When and What to Maximize 131
5.2.2 From ASTI to ASTI+ 133
5.2.2.1 Analogy 133
5.2.2.2 Transformation 133
5.2.3 Integration Strategy 133
5.2.3.1 Option-Usage Strategy 133
5.2.4 Experimental Studies of the ASTI+ Model 134
5.2.5 Summary 138
5.3 Soft Jaccard Index 140
5.4 An Analogy-Making View 142
5.4.1 Intuitions Behind the Proposed Approach 143
5.4.2 Correspondence Finding 145
5.4.3 Experimental Studies 148
5.4.4 Summary 150

6 Future Work 151

6.1 Goal 1 151
6.2 Goal 2 152
6.3 Goal 3 154
6.4 Summary 156

References 157

LIST OF TABLES

Table Page

1.1 A conversation with GPT-4 to ask it to produce a research proposal of AI research. 2
1.2 A conversation with GPT-4 to test its ability to combine existing knowledge in new ways. 4
1.3 A conversation with GPT-4 to test its ability to establish understanding of new experience. Adapted from the pq-system in Hofstadter (1979) 5
1.4 Ten logic puzzles that GPT-4 (version: Mar 14, 2023) failed. 6

3.1 The technical details of generator programs in the reviewed works. 37
3.2 Computational Models for Solving RPM and RPM-like Tasks 57

4.1 The implementation of the entry encoder F_E. 98
4.2 The implementation of the perceptual and conceptual processing $h_1^{(i)}$,$g_1^{(i)}$,$h_2^{(i)}$ and, $g_2^{(i)}$. . . . 98
4.3 The implementation of the consistency computation $q^{(i)}$. 98
4.4 The implementation of the classification head p_j. 98
4.5 Accuracies on the original RAVEN. We report without-auxiliary-training accuracies if possible. Data source for each row: (Zheng et al., 2019), (Wang et al., 2020), (Zhang et al., 2019b), (Zhuo and Kankanhalli, 2021), (Sahu et al., 2023), (Zhang et al., 2019a), (Hahne et al., 2019), (Zhuo et al., 2021), (Zhang et al., 2021), (Zhang et al., 2022), (Benny et al., 2021), (Hersche et al., 2023), (Wu et al., 2021), (Xu et al., 2023), and (Spratley et al., 2020). 100
4.6 Accuracies on the I-RAVEN. We report without-auxiliary-training accuracies of each model if possible. Data source for each row: (Hu et al., 2021), (Sahu et al., 2023), (Zhuo et al., 2021), (Zhang et al., 2021), (Zhang et al., 2022), (Hersche et al., 2023), (Xu et al., 2023), (Wu et al., 2021). 100
4.7 Accuracies on RAVEN-FAIR. We report without-auxiliary-training accuracies of each model if possible. Data source for each row: (Sahu et al., 2023), (Benny et al., 2021). . . 101
4.8 Ablation Accuracies of CPCNet on RAVEN by Varying L. 101
4.9 Ablation Accuracies of CPCNet on RAVEN by Weakening Consistency in Different Ways. 102
4.10 Testing accuracy of the prototypical CPCNet for each abstract concept and attribute in the 2x2Grid configuration of RAVEN. 108
4.11 Testing accuracy of the prototypical CPCNet for each abstract concept and attribute in the 3x3Grid configuration of RAVEN. 108
4.12 Testing accuracy of the prototypical CPCNet for each abstract concept and attribute in the Out-InGrid configuration of RAVEN. 108
4.13 The Training Item Number for Each Abstract rule and Attribute in the 2x2Grid Configuration of RAVEN. 108
4.14 The Training Item Number for Each Abstract rule and Attribute in the 3x3Grid Configuration of RAVEN. 109
4.15 The Training Item Number for Each Abstract rule and Attribute in the Out-In2x2Grid Configuration of RAVEN. 109
4.16 The Item Number for Each Abstract rule and Attribute in the Center Configuration of RAVEN. 109
4.17 The Item Number for Each Abstract rule and Attribute in the Out-InCenter Configuration of RAVEN. 109
4.18 The Item Number for Each Abstract rule and Attribute in the Left-Right Configuration of RAVEN. 109
4.19 The Item Number for Each Abstract rule and Attribute in the Up-Down Configuration of RAVEN. 110
4.20 Numbers of RAVEN (I-RAVEN and RAVEN-FAIR) Training items containing each combination of abstract rules and attributes. 110

4.21 Numbers of AB-RAVEN Training items containing each combination of abstract rules and attributes. 111
4.22 Accuracies on the AB-RAVEN, using single-choice evaluation protocol. The numbers in the second row reflect the improvement over the orginal imbalanced RAVEN dataset. . . 111
4.23 The number of items of each spatial configuration in RAVEN (I-RAVEN and RAVEN-FAIR) and AB-RAVEN. 111
4.24 The Item Number for Each Abstract rule and Attribute in the Center Configuration of AB-RAVEN. 112
4.25 The Item Number for Each Abstract rule and Attribute in the 2x2Grid Configuration of AB-RAVEN. 112
4.26 The Item Number for Each Abstract rule and Attribute in the 3x3Grid Configuration of AB-RAVEN. 112
4.27 The Item Number for Each Abstract rule and Attribute in the Out-InCenter Configuration of AB-RAVEN. 112
4.28 The Item Number for Each Abstract rule and Attribute in the Out-In2x2Grid Configuration of AB-RAVEN. 112
4.29 The Item Number for Each Abstract rule and Attribute in the Left-Right Configuration of AB-RAVEN. 113
4.30 The Item Number for Each Abstract rule and Attribute in the Up-Down Configuration of AB-RAVEN. 113
4.31 Accuracies on the Trivial Generalization Regimes of PGM. We report without-auxiliary-training accuracies of each model if possible. Data source for each row: (Zhang et al., 2019b), (Wang et al., 2020), (Zheng et al., 2019), (Zhuo and Kankanhalli, 2021), (Hu et al., 2021), (Sahu et al., 2023), (Hahne et al., 2019), (Zhuo et al., 2021), (Barrett et al., 2018), (Rahaman et al., 2021), (Rahaman et al., 2021), (Spratley et al., 2020), (Wu et al., 2021; Małkiński and Mańdziuk, 2020), (Benny et al., 2021), (Wei et al., 2023), (Jahrens and Martinetz, 2020). 117
4.32 Accuracies on the nontrivial Generalization Regimes of PGM. We report without-auxiliary-training accuracies of each model if possible. Data source for each row: (Wang et al., 2020), (Barrett et al., 2018), (Rahaman et al., 2021), (Rahaman et al., 2021), (Wu et al., 2021; Małkiński and Mańdziuk, 2020), (Benny et al., 2021), (Wei et al., 2023). 118

5.1 Details of unary, binary, and hybrid unary/binary transformations. 128
5.2 Configurations of integration strategies, analogy groups and transformation groups. . . . 135
5.3 Similarities between geometric objects in Figure 5.14. The Jaccard index is calculated using Equation (5.5). The soft Jaccard index is calculated using Equation (5.21) with α=0.03, d of one-norm and p=3. 142
5.4 Experimental Results on Geometric Analogy Problems. The second and third columns are two analogical directions. The last column shows the problems that were solved by that interpretation. 148

LIST OF FIGURES

Figure Page

2.1 An example of RPM-like item. (Kunda et al. (2013)) 9

3.1 RPM examples of different formats and stimuli. 17
3.2 Example SPM item series. 24
3.3 A diagrammatic summary of matrix reasoning task 28
3.4 A diagrammatic summary of figure series task 30
3.5 A diagrammatic summary of analogy making task 31
3.6 A diagrammatic summary of contrastive classification task 32
3.7 A diagrammatic summary of open classification task 33
3.8 Example items created by following Hornke and Habon's AIG procedure. 38
3.9 An example of deriving nonharmonic items from harmonic items. 42
3.10 Three categories of relations commonly used in matrix reasoning (Wang and Su, 2015). . 44
3.11 Example items generated through the IMak package. Each item exemplifies a single basic rule. The correct answer is set to the first answer choice for demonstration. 46
3.12 Left: The dependencies among relations, objects, and attributes used to generate the Procedurally Generated Matrices (PGM) dataset (Barrett et al., 2018). Each path from left to right corresponds to a $[r,o,a]$ triplet representing a variation pattern in the matrices. As one can check, there are 29 paths, i.e. $[r,o,a]$ triplets, in the graph. Note that Barrett et al. (2018) did not differentiate between "shape_type" and "line_type" and referred to both of them as "type". But these two are treated as two distinct attributes in PGM's implementation. Right: The dependencies among relations, nodes, and attributes used to generate the RAVEN dataset. Note that we listed "distraction" as a rule in this graph to indicate that uniformity and orientation are distracting attributes. The paths from constant through number and position to layout are treated as a single rule in RAVEN's implementation. Therefore, there are 15 paths/rules in the graph. 48
3.13 7 hardcoded spatial configurations—center, 2x2Grid, 3x3Grid, Left-Right, Up-Down, Out-InCenter, and Out-In2x2Grid—are used to arrange objects in each matrix entry in the RAVEN dataset. Each configuration is represented by the bounding boxes that objects could occupy. The position and size of each bounding box are hardcoded in the generator program. An example matrix for each configuration is given in the first row (image obtained by running the generator code). Note that not every bounding box has to be occupied, but every object has to be in one of the bounding boxes. 49
3.14 The general A-SIG tree and 7 specific A-SIG trees used in the RAVEN dataset (image adapted from (Zhang et al., 2019a) by adding more technical details from the source code of the generator). The root node denotes the scene that the image describes. The structure nodes are the containers of different spatial structures. A structure is composed of components that could be overlaid with each other. Each component has its own layout and, more importantly, variation rules, which are independent of other components. The layout node, as its name indicated, contains the attributes specifying the number and positions of geometric objects. Entities represent geometric objects with attributes, not including number and position. 50
3.15 The spatial configurations of PGM represented in A-SIG to compare with RAVEN. There are 3 spatial configurations in PGM—line, shape, and shape-over-line—in PGM. The example matrix is given for each configuration at the bottom (images taken from PGM dataset). 51
3.16 The answer sets of three versions of RAVEN datasets depicted in graphs. Each vertex is an answer choice and two adjacent vertices differ by one attribute. 53
3.17 Learning approach Type 1 64
3.18 Learning approach Type 2 66
3.19 Learning approach Type 2+ 67
3.20 Learning approach Type 3 68

3.21 Learning approach Type 3+ 69
3.22 Learning approach Type 4 71

4.1 A is to B as C is to D. But in what ways? 87
4.2 An example item of Raven's Progressive Matrices Kunda et al. (2013). 89
4.3 Image Classification Paradigm for Solving RPM. Note that the sizes and numbers of tensors are diagrammatic, not representing the real implementation. 90
4.4 A Paradigm Simulating Human Cognition for Solving RPM. Note that the sizes and numbers of tensors are diagrammatic, not representing the real implementation. 91
4.5 CPCNet: A Feed-Forward Architecture that unrolls the paradigm in Figure 4.4. 94
4.6 7 spatial configurations of the RAVEN dataset. Each configuration is illustrated by a complete 3x3 matrix. In the **Center** configuration, each entry contains only one object located at the center. In **2x2Grid** and **3x3Grid**, objects can only be located on the grid positions in each entry. In **Left-Right** and **Up-Down**, each entry contains exactly two objects located at the fixed positions shown in the figure. **Out-InCenter** is a combination of two center configurations and **Out-InGrid** is a combination of a center configuration and a 2x2Grid configuration. 103
4.7 Non-monotonic processing of a multi-dimensional array *D*. This figure visualizes the computation of Equation (4.11), (4.12), (4.13), and (4.14). 120

5.1 Sample problems similar to those from the Standard version of the Raven's Progressive Matrices (RPM) test (Raven et al., 1998). Real RPM problems are not shown in order to protect the confidentiality of the test. 124
5.2 Illustration of input to our model for the 2×2 example problem in Figure 5.1, where m_{ij} is the matrix entry in row i and column j, and O_{k} is the k-th answer option. 124
5.3 Illustrations of affine transformations used in our model. 126
5.4 Illustrations of set transformations used in our model: (a) Given an analogy *A:B::C:?* and an unary set transformation T, the output image is $O{=}T(C|A,B)$, where C is the input, and B and C are parameters of T; (b) Given an analogy *A:B:C::D:E:?* and a binary set transformation T, the output image is $O{=}T(A,B|C)$ when T is applied on *A:B:C*, where A and B are the inputs, and C is a parameter of T, or $O{=}T(D,E|O')$ when T is applied on *D:E:?*, where O' is an option of the RPM problem. 126
5.5 Illustrations of simple analogies in RPM problems. Simple analogies reflect how a matrix layout is naturally perceived as rows or columns. Particularly, given 2×2 matrices in (a) and (b), the row analogy is *A:B::C:?* and the column analogy is *A:C::B:?*; similarly, given 3×3 matrix in (c) and (d), the row analogies include *A:B:C::G:H:?* and *D:E:F::G:H:?*, and the column analogies include *A:D:G::C:F:?* and *B:E:H::C:F:?*. 127
5.6 Illustrations of recursive analogies in 3×3 RPM problems: (a) are (b) are trio analogies and (c) through (j) are pair analogies. 129
5.7 Expanded matrices to generate analogies: (a) through (c) are expanded from the 2×2 matrix in Figure 5.5, and (d) through (g) are expanded from the 3×3 matrix in Figure 5.5. 130
5.8 The dependencies of scores: The dashed lines denote partial dependence. Given the relations in an analogy, MAT relies on the entries that are not related to the missing entries while O relies on the entries that are related to the missing entries. 131
5.9 Performance of each strategy on the standard RPM test: (a) and (b) show numbers of problems correctly solved by each strategy in every set (A—E) and the entire test; (c) and (d) visualize MAT and O scores of each strategy's answer to each problem as disks of various sizes and colors, where red disks indicates incorrect answers and blue disks indicate correct answers. Note that the "signed" O score in (c) and (d) is only to distinguish visually between correct and incorrect answers, and the real scores always fall in [0,1]. . . 136
5.10 Scatter plots of each strategy's answer to each problem in the standard RPM test drawn with respect to the MAT and O scores. 137
5.11 Bar charts of numbers of problems correctly solved by M-prudent strategy using different analogy groups and transformation groups. (This figure should be viewed in color.) . . . 138

5.12 Comparison between ASTI+ and human subjects. The blue horizontal lines denote the performance of the configurations used in our experiments. The green curves represent the percentiles of human data. 139
5.13 A simple example of geometric analogy problems (Lovett et al., 2009). 140
5.14 Geometric objects. Each cell in a grid denotes a pixel. 142
5.15 Correspondences between geometric objects. The correspondences between *A* and *B* and between *C* and *D* are shape mappings; the mappings between *A* and *C* and between *B* and *D* are derived from inside/outside spatial relations. 144
5.16 The typical problems solved by each interpretation. 148

6.1 Images generated by a Stable Diffusion model given the prompt “photograph of an astronaut riding a horse”. 152
6.2 Images generated by a Stable Diffusion model given the prompt “An astronaut riding a porcupine”. 152
6.3 Images generated by a Stable Diffusion model given the prompt “A horse riding an astronaut”. 153

CHAPTER 1

AI's AI Research Proposal

Most research projects start with a research proposal. In our case of AI research, it would be interesting to consider how AI systems would compose such a research proposal. When this dissertation is written, people are enjoying the feast of large language models (LLMs), which was brought about by a newly-launched model—GPT-4. The company—OpenAI—that created this model claimed that GPT-4 "exhibits human-level performance on various professional and academic benchmarks", for example, Uniform Bar Exam, SAT, GRE, and AP tests. Moreover, based on its predecessor ChatGPT 3.5—a pure language model, GPT-4 was augmented to a multi-modal model, which takes both visual and text inputs. Besides basic multi-model tasks, such as VQA datasets and visual naive physics, a research preview even shows that it is capable of capturing the humor in its visual input, which has been considered unique to human thinking. In short, it seems that GPT is more and more like a human assistant.

Therefore, to start my dissertation research, I asked GPT-4 to produce several research proposals and evaluated it on this writing task that requires not only rigorous and critical reasoning but also creative thinking. Table 1.1 shows my prompt and its response.

Given such a simple prompt, it produced a complete, meaningful, well-structured, and properly-worded research proposal. Considering my effort to type this short prompt, such a research proposal is fairly satisfactory. If the length of GPT-4's response was not limited by its computational resources, it would probably write a longer research proposal as we often did for real research projects.

Writing a research proposal is no doubt an intellectually challenging job for researchers. This is why I included in my prompt keywords *novel* and *important research question*—GPT-4 is usually not as sophisticated as a human assistant who will fill in the missing connotations and background knowledge of your prompt; instead, to obtain the desirable response, the prompt should be as clear and specific as possible, i.e., *Prompt Engineering*. Given its probabilistic nature, I drew multiple samples with the same prompt, and the one in Table 1.1 is a very representative one among the these samples. Other research proposals' titles are listed below:

- Exploring the Interplay between Human Ethics and Artificial Intelligence: A Framework for Responsible AI Development
- Towards Generalizable and Ethically Conscious Artificial Intelligence: An Interdisciplinary Approach
- Exploring the Ethical Dimensions and Human-Centric Development of Artificial General Intelligence

Me: Give me a dissertation research proposal that explores novel and important research questions of artificial intelligence.

GPT-4: (the following is a formatted version of GPT-4's response)
Title: Exploring the Ethical Boundaries and Social Impact of Artificial General Intelligence

- **Abstract:** The advent of artificial general intelligence (AGI) has the potential to bring about unprecedented advancements in various fields, while also raising novel and critical ethical concerns. This dissertation research proposal aims to explore the ethical boundaries and social impact of AGI, focusing on the development, deployment, and governance of these advanced systems. By employing a multidisciplinary approach that integrates computer science, ethics, and social sciences, the research will delve into the potential consequences of AGI on society, the environment, and the economy, with the objective of providing actionable insights for stakeholders in the AI community.
- **Objectives:** Investigate the ethical principles and frameworks that guide the development and deployment of AGI, with a focus on transparency, accountability, and fairness. Assess the potential social, economic, and environmental consequences of AGI adoption across various sectors, including healthcare, education, and transportation. Examine the existing governance structures and regulatory mechanisms related to AGI and propose enhancements to ensure responsible development and deployment. Develop policy recommendations to address potential risks and maximize the societal benefits of AGI.
- **Methodology:** The research methodology will comprise both qualitative and quantitative approaches, as follows:
  - **Literature review:** A comprehensive review of relevant literature, including academic publications, white papers, and industry reports, will be conducted to establish a solid theoretical foundation and identify key themes and gaps in the current understanding of AGI ethics and social impact.
  - **Case studies:** In-depth case studies will be conducted to investigate AGI applications in various sectors, identifying best practices and challenges in the development, deployment, and regulation of these systems.
  - **Expert interviews:** Semi-structured interviews with AI researchers, ethicists, policymakers, and industry practitioners will be conducted to gather insights on AGI ethics, governance, and potential societal impacts.
  - **Surveys and statistical analysis:** Quantitative data will be collected through surveys and secondary data sources to assess the public perception of AGI and its potential implications on the economy, society, and environment. This data will be analyzed using descriptive and inferential statistical techniques.
- **Expected Outcomes:** A comprehensive understanding of the ethical considerations surrounding AGI development and deployment, including transparency, accountability, and fairness. Identification of potential risks and benefits associated with AGI adoption across various sectors, such as job displacement, economic growth, and environmental sustainability. Assessment of existing governance structures and regulatory mechanisms related to AGI, highlighting areas for improvement. Policy recommendations for responsible AGI development, deployment, and governance, to ensure that the potential benefits are maximized while minimizing risks to society, the environment, and the economy.
- **Significance:** This research will contribute to the growing body of knowledge on the ethical and social implications of AGI, providing valuable insights for AI researchers, policymakers, industry practitioners, and other stakeholders. By exploring the ethical boundaries and social impact of AGI, the findings will help inform responsible development and deployment practices, fostering a future where AGI is used to enhance human well-being and protect the environment.

Table 1.1: A conversation with GPT-4 to ask it to produce a research proposal of AI research.

- Exploring the Interplay between Human Ethics and Artificial Intelligence: A Framework for Responsible AI Development
- Exploring Synergistic Approaches to Enhance Human-AI Collaboration: A Multidisciplinary Analysis
- Exploring the Ethical and Social Implications of Artificial General Intelligence: A Multidisciplinary Approach

A very interesting observation about these proposals is that they all deal with AI research from ethical or social angles, rather than the very technical and scientific nature of AI research. To make sure this is not caused by my browser history or cookies, I tried it on different computers, in incognito mode, and in different sessions with GPT-4. But the result remained the same. Admittedly, all of these proposals are *important research questions* in their own right. But these six proposals are far from being *novel* or *original*; on the contrary, the similarity between these proposal titles implies lack of diversity and bias towards certain aspects of prompts.

Two more observations can be made at a finer level than titles and topics:

- Observation 1: The arguments in GPT-4's response is too general and not supported by reasoning on facts . It keeps repeating the same abstract ideas without clearly defining them or relating them to details in reality, much less logical connections between these concepts.
- Observation 2: All the GPT4's responses follow the same pattern at both sentence and paragraph levels. For example, all of them start with a sentence that first agrees with the positive side of AI and then questions the negative side of AI. This pattern-following behavior makes GPT-4's writing insipid to read (if you read six or seven of them in a row).

In a nutshell, what makes GPT-4's proposals different from real human-written research proposals is the diversity and creativity in thinking, when reflected in writing, and the ability to combine specific facts and abstract ideas in a rigorous but creative way. Given the gigantic corpus that was used to train GPT-4, the resulting distribution in GPT-4 leads to the mean or the average of responses that humans have made in the corpus. However, the major criterion for assessment of a research proposal is whether this proposal stays away from the mean and explores the frontier of the in-distribution area in a creative and rigorous way. Human writers are clearly aware of the two different types of tasks—the average-response-needed task or the research-proposal-like task, whereas GPT-4 is not designed to distinguish them, let alone accomplish them effectively.

Admittedly, these observations are not GPT-4's defects because GPT-4 is not designed to work like a human content creator. GPT-4's responses to requests for **creative** and **rigorous** content do point out an ideal direction for constructing future AI. To be creative, the AI system must either **improve its understanding of existing experience (e.g., training corpus or learned parameters)** or **establish new understanding of new experience (e.g., user prompts to GPT-4)**. To examine GPT-4 for these abilities, I had another conversation with GPT-4, shown in Table 1.2. In this conversation, I asked GPT-4 to count the letter “e” in its last response. If you count the letter “e” in the response, you will find that there are 16 "e"; but GPT-4 replied 27. Then I asked another follow-up question to make sure that it has the concept of number and it can count to 16. Obviously, the training dataset has imparted to GPT-4 necessary prior knowledge, such as how to greet people and counting; but it failed to form new understanding on its existing knowledge when it was facing a new challenge.

Me:	Hi GPT-4, how is it going today?
GPT-4:	Hello! As an AI language model, I don’t have feelings or experiences, but I’m here to help you with any questions or information you need. What can I assist you with today?
Me:	How many "e" are there in your last response?
GPT-4:	In my last response, there are 27 instances of the letter "e".
Me:	Can you count from 1 to 16?
GPT-4:	Of course! Here are the numbers from 1 to 16: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16

Table 1.2: A conversation with GPT-4 to test its ability to combine existing knowledge in new ways.

As indicated above, another situation that entails creativity is where a human or an AI needs to establish understanding of new experience. I had another conversation with GPT-4 to illustrate this point, as shown in Table 1.3. This example is adapted from the pq-system, which was devised by Hofstadter (1979). It represents integer addition in a typographical manner, for example “- - p - - - q - - - - -” corresponding to “2 plus 3 equals 5”. Although GPT-4 gave a quite long response in Table 1.3, it failed to capture this new form of addition. A follow-up question was also asked to make sure that GPT-4 understands addition in its traditional form.

We now return to the initial topic of AI’s research proposal. In the above discussion about GPT-4’s research proposal, I pointed out the absence of the most important features of research proposal—the accuracy about details and the rigorousness of reasoning based on logical connections. Especially, the ability to reason about complex conceptual structures is always deemed a hallmark of real intelligence. GPT-4, though advertised as the most advanced AI system and costing billions[1] to develop, shows a weak ability on such tasks where rigorous reasoning is needed. For example, OpenAI included in their AI model testing framework ten simple logic puzzles that GPT-4 (version: Mar 14, 2023) failed, as shown in Table 1.4.

Such inconsistency of GPT-4’s performance across different tasks, e.g., information retrieval and solving logic puzzles, indicates there are still important missing pieces of the puzzle of AI. Among the ten puzzles in Table 1.4, you can find extremely easy ones, such as Puzzle 1 and 6, and also the ones that might take

---

[1] OpenAI built their entire deep learning stack and a supercomputer to train deep learning models, which has more than 285,000 industrial CPU cores, 10,000 industrial GPUs and 400 gigabits per second of network connectivity for each GPU server.

Me:	Given valid strings "- p - q - -", " - - p - q - - -", "- p - - q - - -", "- - p - - q - - - -", and "- - - p - - q - - - - -", is "- - - - - p - - - - - q -" valid? Why?
GPT-4:	Yes, "- - - - - p - - - - - q -" is a valid string based on the pattern observed in the given strings. The pattern consists of two characters, ’p’ and ’q’, separated by a varying number of hyphens. For each subsequent string, the number of hyphens before ’p’, between ’p’ and ’q’, and after ’q’ increase by one. Here is a breakdown of the given strings: - p - q - - Hyphens before ’p’: 1 Hyphens between ’p’ and ’q’: 1 Hyphens after ’q’: 2 - - p - q - - - Hyphens before ’p’: 2 Hyphens between ’p’ and ’q’: 1 Hyphens after ’q’: 3 - p - - q - - - Hyphens before ’p’: 1 Hyphens between ’p’ and ’q’: 2 Hyphens after ’q’: 3 - - p - - q - - - - Hyphens before ’p’: 2 Hyphens between ’p’ and ’q’: 2 Hyphens after ’q’: 4 - - - p - - q - - - - - Hyphens before ’p’: 3 Hyphens between ’p’ and ’q’: 2 Hyphens after ’q’: 5 Now, let’s look at the provided string: - - - - - p - - - - - q - Hyphens before ’p’: 5 Hyphens between ’p’ and ’q’: 5 Hyphens after ’q’: 1 The string "- - - - - p - - - - - q -" maintains the pattern of increasing hyphens between ’p’ and ’q’ and does not violate any rules observed from the given strings. Therefore, it is a valid string.
Me:	What is 5 plus 5?
GPT-4:	5 plus 5 equals 10.

Table 1.3: A conversation with GPT-4 to test its ability to establish understanding of new experience. Adapted from the pq-system in Hofstadter (1979)

quite a while to solve, such as Puzzle 3 and 4. According to our experience with logic puzzles and interaction with GPT-4, it seems that GPT-4 should be capable of solving logic puzzles because the the solving processes require multiple reasoning steps and GPT-4 works in a similar way, i.e., repeatedly predicting/sampling the next words given the already predicted/sampled words.

But why can’t GPT-4 solve them? Two characteristics might contribute to the difficulty of logic puzzles. First, there exist infinitely many logic puzzles and infinitely many conceptual structures behind logic puzzles. Therefore, although the corpus used to train GPT-4 goes beyond the knowledge of every single human being, it still cannot capture all the possible conceptual structures, among which some could be extremely complex and rare, not existing in the corpus. Let alone there are more logic puzzles that are to be discovered.

Secondly, some logic puzzles and some reasoning tasks are not solvable by purely executing operations mechanically. Taking Puzzle 3 in Table 1.4 as an example, to solve this, we need to start from the person who purported to have shaken hands with 99 truth-tellers, say Person 99. If Person 99 was telling the truth, then she must have shaken hands with Person 0; but all the people shaking hands with Person 0 were liars; thus, Person 99 was not telling the truth. We then move on to Person 98; since Person 99 is a liar, Person 98 must have shaken hands with Person 0; then Person 98 is liar. By repeating the same argument, we reach Person 0 with Person 1 through 99 being liars; thus, Person 0 is a truth-teller. Such a solution does not seem to pose a great challenge for mechanically-operation-executing models, like GPT-4. But you might have noticed the wording “By repeating the same argument” in the solution. The key point is that we did not really

Puzzle 1:	Solve the following logical puzzle. Carefully think step by step, and show your reasoning. If there is not enough information to solve the puzzle, conclude with 'There is not enough information.' There are five students, Anna, Bob and Cynthia, Dan and Eliza. They all tell the truth. Anna is taller than Cynthia. Bob says he's taller than Anna if and only if Eliza is the shortest. Cynthia is taller than Dan. Eliza is shorter than Dan. Who's the tallest in the group? Let's think step by step.
Answer:	Anna > Cynthia > Dan > Eliza. But, based on Bob's statement, there are still two possibilities: 1. Bob is taller than Eliza, making Eliza the shortest, making Bob taller than Anna, making Bob the tallest. 2. Bob is shorter than Eliza: this would still be valid, as Eliza wouldn't be the shortest and therefore Bob isn't taller than Anna. And Anna would be the tallest. So there's not enough information.
Puzzle 2:	Laura thinks that Jessica thinks that Angie is only 23 years old. Angie thinks Josie knows where Laura's mother is. Jessica thinks Laura was once an engineer. Josie thinks Laura is friendly. Based on the text, what thoughts do we know that Laura, Jessica, Angie, and Josie have?"
Answer:	Laura thinks: Jessica thinks Angie is only 23 years old. Jessica thinks: Laura was once an engineer. Angie thinks: Josie knows where Laura's mother is. Josie thinks: Laura is friendly.
Puzzle 3:	At a party, there are 100 people. Some always lie and some always tell the truth. They all know which one of them is a truth-teller and which one is a liar. After the party, you ask each person how many truth-tellers they shook hands with. Each person gives a different answer, ranging from 0 to 99. How many were truth-tellers and how many were liars?
Answer:	There is 1 truth-teller and 99 liars at the party.
Puzzle 4:	Two people want to cross a river. The only way to get across is with a boat that they find on one side; but that boat can only take one person at a time. The boat cannot return on its own, and there are no ropes to haul it back, yet both persons manage to cross using the boat. How did they do it?
Answer:	The people are on different sides of the river, so the person on the same side as the boat originally can cross first to bring the boat to the side with the other person, then that person can cross.
Puzzle 5	There are two men. One of them is wearing a red shirt, and the other is wearing a blue shirt. The two men are named Andrew and Bob, but we do not know which is Andrew and which is Bob. The guy in the blue shirt says, 'I am Andrew.' The guy in the red shirt says, 'I am Bob.' If we know that at least one of them lied, then what color shirt is Andrew wearing?
Answer:	Andrew is wearing the red shirt.
Puzzle 6:	Which word does NOT belong with the others? A. index B. glossary C. chapter D. book
Answer:	D. book
Puzzle 7:	The day before yesterday, Chris was 7 years old. Next year he'll turn 10. How is this possible?
Answer:	Assuming today is January 1st of any given year: Two days ago, on December 30th, Chris was 7 years old. On December 31st, Chris celebrated his 8th birthday. On December 31st of this year, Chris will celebrate his 9th birthday. By the end of this year, on December 31st, Chris will be 10 years old. Therefore, on January 1st of the following year, Chris will be 10 years old.
Puzzle 8:	Inhabitants of an island lie consistently on Tuesdays, Thursdays, and Saturdays, and they tell the truth on the other four days of the week. You have forgotten what day of the week it is, so you ask a passerby. 'Saturday,' he answers. 'And what day will it be tomorrow?' you inquire. 'Wednesday,' he replies. Can you tell what day it is today?
Answer:	Based on answer 1, today cannot be M, W, F, Su, or Sa (lying day). Based on answer 2, today cannot be M, W, F, Su, or Tu (lying day). So, today must be Thursday.
Puzzle 9:	You are on an island populated by two tribes. Members of one tribe consistently lie. Members of the other tribe always tell the truth. Tribe members can recognize one another, but you can't tell them apart. You meet two people, C and D on the island. C says, 'Exactly one of us is from the liars tribe.' Which tribe is D from?
Answer:	D is from the Liars tribe.
Puzzle 0:	There are five people in a room. Each person will either always tell the truth or always tell a lie. Each person is asked the following question: How many liars are among you? The answers are: öne,̣ ẗwo,̣ ẗhree,̣ f̈our,̣ f̈ive.̣ How many liars are in the room?
Answer:	There are four liars.

Table 1.4: Ten logic puzzles that GPT-4 (version: Mar 14, 2023) failed.

repeat the same argument from 99 to 1; instead, this argument made our observation to a higher level and observed a pattern **outside the system of basic reasoning operations**. This is where the difficulty comes in. Once the system of operations is fixed, it is difficult for it to make observation about itself from a third-party view. Thus, it is always possible to create new logic puzzles or reasoning task based on a mechanical model's current operations to attack it. For example, if Puzzle 3 does not stop at 0 and requires induction to negative integers or even to negative infinity, then mechanical execution will not be able to solve the puzzle. On the flip side, one may argue that a machine could have built-in meta-operations that supervise basic operations from a higher level to extract patterns for induction. But what can be used to supervise meta-operations if puzzles are built upon meta-operations? What if an induction of induction is needed to solve the puzzle?

Using an AI research proposal composed by AI as a starting point, this introduction gives a flavor of the specific research questions that I investigate in the following chapters of this dissertation. First, the current AI systems show limited sign of creativity, originality, or novelty outside the training dataset; it failed to develop new insights into old experience and establish understanding of new experience. In short, it generates content from its training set, but does not invent content. Second, the current AI systems are generally limited in dealing with the flexibility and complexity of abstract reasoning tasks. These abilities also correspond to the core cognitive abilities that are often tested in human intelligence tests. Inspired by how these abilities are studied in human cognitive science, I will propose the specific research questions in the large background of incompleteness of current AI systems and the science of human cognition.

CHAPTER 2

Research Questions

2.1 Problem Statement

In Chapter 1, we analyzed the difference between how the most advanced AI system generates research proposals and how an ordinary human research (e.g., me) composes a research proposal.[1] The human is fully aware of what kind of task he needs to do and thus think about how to do the task before really doing it. Chapter 1 is exactly an explicit manifestation of this thought process, which has been made implicit by most human proposal authors. This kind of self-awareness is similar to the solving process of logic puzzles that GPT-4 has failed, i.e., observing what is happening at a higher level and develop new understanding of the current situation based on one's existing knowledge of the world, e.g., extracting patterns from the operation history. It is an instinct for humans, as intelligent beings, to think and behave this way, whereas almost all AI systems, even the most advanced and sophisticated ones, have not realized these abilities yet.

This instinct is a main factor (or factors) that is measured by most human intelligence tests. It is a concrete factor that has its solid genetic, biological, and psychological grounds. Unlike benchmarking an AI system on typical datasets in a specific task domain, this latent factor is not exclusively associated with any specific task, but is connected to all intellectual activities to different degrees. Thus, intelligence tests usually consists of a wide range of tasks, such as visual puzzle, abstract reasoning, analogy making, arithmetic, vocabulary, etc. On the flip side, similar to benchmarking AI systems, strict protocols for designing and administrating intelligence tests must be followed to make sure that the testing result is valid and reliable. The most important characteristic of the protocols is that the design of testing items should reduce the effect of non-intelligence factors to a minimal level. For instance, if an intelligence test is designed to be used globally, the content of the test should be insensitive to cultural and language background; if an intelligence test is designed for preschoolers, it should not involve knowledge that is taught at elementary or middle schools. Intuitively, the more the intelligence tests are independent on non-intelligence factors, the better the intelligence tests can measure human intelligence.

An example of such intelligence tests, which is used with people of various occupations, in different countries, and at all ages, is the Raven's Progressive Matrices (RPM). In recent years, it has received more and more attention from AI researchers, mostly because statistical analysis shows that RPM is located at the

---

[1] It is very possible that some human writers' writing are extremely similar to the GPT-4's output. Some people might even unconsciously imitate GPT-4's writing as they are more and more using GPT-4 in their daily works, because they might think GPT-4, as a machine, will always give the most correct answer, like a calculator. This indeed raises a concern that AI systems, like GPT-4, will be an obstacle for humans to be exploratory and creative.

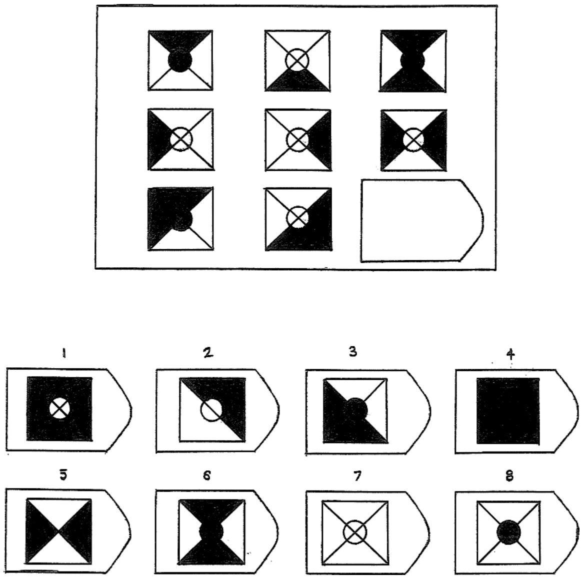


Figure 2.1: An example of RPM-like item. (Kunda et al. (2013))

center of all intelligence tests, that is, exhibiting strongest correlation with all other intelligence tests. Also AI research is generally shifting from task-specific ability to Artificial General Intelligence (AGI). Figure 2.1 shows an example of RPM items. It is a multiple-choice problem; to solve the item, one needs to select an answer choice that best completes the matrix. The answer to this example item is No. 4, for if you take union of the first two entries in each row (or column) at the pixel level, you will have the third entry.

RPM measures exactly the instinct of human intelligence mentioned above. First, for the test result of RPM to be valid, the participant should not have any prior experience with RPM. In particular, the participant should not be specially trained on RPM before the test. This is true for all human intelligence tests. Second, the administrator should not provide any information about meaning, purposes, or criteria of solving RPM items, Instead, the administrator only tells the participant that she needs to select an answer choice that best completes the matrix without explaining what “best” means. In a very special case where the RPM test is given nonverbally, for example, the administrator does not speak the language of the participant, the administrator uses body language, such as pointing, nodding, and shaking head, to convey the idea that the participant needs to select an answer choice that fits in the matrix. I will discuss more about psychometrics of RPM and the intelligence testing theory in Chapter 2. The key point here is that the participant is facing an completely unfamiliar situation without clear instruction, in which she needs to establish an understanding of the situation based on her existing knowledge of the world and react accordingly, i.e., using the instinct mentioned above.

Consider the example in Figure 2.1. Suppose that the administrator has told the participant verbally that she needs to select an answer that best completes the matrix. The participant would consider what "best completes" means; what are the criteria of a correct answer; she would inspect the matrix row by row or column by column; she would compare the matrix entries and superimpose one on another; this way she would probably notice the relation of taking union in the first rows; she would then verify if the relation holds in the second row; she would finally take union of the first two entries of the third row and find an unique answer, i.e., No. 4; given the uniqueness, she would be pretty confident that she got it correct. Before seeing this RPM item, the participant must have some knowledge about matrix, i.e., rows and columns are formed according to some rules; she knows that images can be superimposed on one another and compared at pixel level; she probably knows the concept of union; however, she has never seen these elements being pieced together; to solve this item, she needs to establish an understanding of the unfamiliar situation based on her existing knowledge. All RPM items are designed this way, i.e., using common sense, common concepts, common knowledge, and innate mental abilities to create unfamiliar situations for participants to understand.

If we compare human intelligence testing and AI testing, we can find a major difference between them—human intelligence tests measure specific cognitive abilities that underlie all intellectual tasks, whereas AI testing measures performance on specific clearly-defined tasks that involve almost all cognitive abilities. An unavoidable problem is that if the ultimate goal of AI research is AGI, a method will be needed to measure AI's cognitive abilities, rather than task-specific performance, sooner or later. But the difficulty for test AI's cognitive abilities is that it is hard to find a single clearly-defined test that if we build an AI system that perfectly pass the test, then we can say that we are done with the cognitive abilities that are tested by the test. That is, the success on this specific test guarantees success on all possible tests that require the same set of cognitive abilities; at least, failures are not caused by lack of these cognitive abilities.

As indicated above, RPM is located at the center of all intelligence tests and the intelligence factor that RPM measures is exactly the ability that the most advanced AI systems lack. Therefore, an ideal first step toward measuring AI's cognitive abilities can be taken by translating RPM tests to AI tests, including not just the test content but the administration protocols. If this is possible, then the new tests can guide the development of AGI systems in a more reasonable direction. On the flip side, if we can build an AI system that solves RPM or RPM-like tests in a similar way that human participants solve them in intelligence testing sessions, then this system is more likely to exhibit task-nonspecific cognitive abilities in many different tasks than RPM. **Therefore, the core proposal of this dissertation research is to build cognitively-inspired AI systems that can solve RPM or RPM-like tests in a similar way that human participants solve them in human intelligence testing.** This core proposal can be better addressed by decomposing it into several research questions, which will be discussed in the remaining sections of this chapter.

2.2 Research Question 1: From Psychometrics to Artificial Intelligence

Since the core proposal is derived from RPM, it is necessary to explore the psychometric origin of RPM and the computational works based on RPM and RPM-like tasks. In particular, the following specific research questions should be explored:

- What does RPM measure exactly? And how does RPM measure it?
- What are the entire task domain that RPM represents? i.e., what are the other tasks in the task domain?
- How do current AI systems solve RPM and RPM-like tasks?

2.3 Research Question 2: The Interplay between Perceptual and Conceptual Processing in Visual Abstract Reasoning

RPM is a classical visual abstract reasoning task. As many other visual abstract reasoning tasks, it requires reasoning about abstract concepts (e.g., union in Figure 2.1) which are represented by raw perceptual/visual elements (e.g., different geometric shapes and colors in Figure 2.1). In the case of trivial generalization, the combinations of abstract concepts and perceptual elements are the same across testing and training. This case corresponds to the common situation in machine learning where the distributions in training and testing datasets are the identical or similar. The difference is that, in common machine learning tasks, there is usually no clear distinction between abstract concepts and perceptual elements when collecting or creating the dataset, and thus no explicit control over abstract concepts and perceptual elements. In the case of nontrivial generalization, as the name indicates, the combinations of abstract concepts and perceptual elements in training are different from the ones in testing. For example, before seeing the example in Figure 2.1, the AI system knew the geometric shapes, colors, and the concepts of union, but never saw them combined together. Although visual abstract reasoning tasks, like RPM-like datasets, can be presented to AI systems in either trivial or nontrivial manners, a reasonable criterion for evaluating AI systems should include both of them.

Thus far, the nontrivial generalization in visual abstract reasoning poses a greater challenge for AI than the trivial one. The nontrivial generalization is fundamentally different from the classical views of learning tasks, such as approximating high-dimensional functions, and/or generalizing to similar probability distributions. Consider a neural network trained to classify images containing two geometric shapes (one on the left and the other on the right) according to the relation between these two shapes. We assign an image a positive label (1) if the size of the left shape is larger than that of the right one or (2) if the color of the two shapes is the same; otherwise, we assign it a negative label. If the neural network was well trained, it should have proper representations of size, color, “larger” and “same”, but only in pairs of (size, “larger”) and (color, “same”). If the network really learned the abstract concepts of “larger" and “same”, then it should also be able to handle the pairs of (color, "larger") and (size, "same"). This kind of generalization is particularly difficult because once the training was finished, the representations are fixed, but generalizing to the new pairs of (color, "larger") and (size, "same") actually gives new meanings to old perceptual elements—color and size—and new interpretations to the old abstract concepts—"larger" and "same".

Intuitively, in order for such nontrivial generalization to work, we need to either consider the possible new meanings and interpretations during training or design architectures that allow dynamically forming new meanings and interpretations when the new pairs are observed during testing. This intuition inspires me to consider the dynamics interplay between perceptual and conceptual processing. In particular, without the interplay, the perceptual and conceptual processing are just encoding abstract concepts and visual stimuli, respectively; with the interplay, the perceptual and conceptual processing are possibly encoding the underlying processes of applying abstract concepts on perceptual stimuli and deriving perceptual stimuli from abstract concepts. I hypothesize that the interplay between perceptual and conceptual processing could be a key to nontrivial generalization. It needs to be pointed out that there might exist other keys to nontrivial generalization and also that the interplay might not be a sufficient key (but just a necessary component). Nonetheless, I decided to investigate the effect of the interplay between perceptual and conceptual processing on visual abstract reasoning tasks. That is, my second research question is **to what extent the interplay mechanism between perceptual and conceptual processing can help solve visual abstract reasoning tasks, especially nontrivial generalization, in AI systems**.

2.4 Research Question 3: Computational Imagery

An introspection of how we humans solve RPM would arguably tell that we are solving RPM, a visual abstract reasoning task, **visually**. In cognitive psychology parlance, we can use mental imagery to solve RPM. Mental imagery is an imagistic representation that can be manipulated mentally. The advocates of mental imagery argue that the way mental imagery functions in human brain is the biological/neural basis upon which other higher cognitive abilities are built. Mental imagery is important for human cognition and for building AI systems because it allows abstract concepts to be incarnated and applied on visual stimuli to generate mental images. This is possible even when such application of abstract concepts is impossible or nonexistent yet in reality. Thus, mental imagery is an important cognitive ability for robustness and generalizability in unfamiliar situation and for creativity. Given these advantages of mental imagery, the third research question is **whether imagery, when implemented computationally in AI systems, is sufficient for solving visual abstract reasoning tasks**.

2.5 Research Question 4: From Generative to Productive

However, mental imagery is probably not the only basis of human cognition. In the community of cognitive science. the ongoing debate over imagery has been lasting for decades and never got resolved. Another major competitor is the propositional representation. Beyond cognitive science, AI researchers are also facing a similar debate over different representations. The difference between human cognition and AI is that no matter what representation (imagery or proposition, or both) human cognition uses, it supports robust intellectual ability in all situations, but choosing one representation against another in AI systems usually means significant limitations in some scenarios. Thus, the interesting of using imagery in AI system is to see how we can use it in a way that is as robust as in human cognition.

A possible direction is to extend the pure imagery-based system to an imagery-based production system. And interestingly, it seems that the work can be started from the generative models, such as autoencoder, GANs, transformers, and diffusion models, that have become especially popular in recent years. These models can be considered as prototypical imagery-based production systems in that they usually do not involve multi-step reasoning about complex structure of abstract concepts and rendering the outcome of each reasoning step in images. Consider the situation where a human subject solves RPM: she would inspect matrix entries row by row and column by column, apply the image operations, and generate many intermediate results; she would also make an analogy between rows and columns, which is at a higher level than the concepts embodied by rows and columns; when multiple concepts and perceptual elements were involved, she would iteratively process each of them, possibly retracting previous results and redoing them. Current generative models are not able to implement all these procedures and orchestrate them effectively. Thus, the fourth research question is **how to extend a generative model to a production system that is able to produce a flexible reasoning trajectory in imagery**.

2.6 Summary

According to the above discussion, we have the following research questions:

1. How are RPM and RPM-like tests used in human intelligence testing and in AI research?
2. To what extent the interplay mechanism between perceptual and conceptual processing can help solve visual abstract reasoning tasks, especially nontrivial generalization, in AI systems
3. Is imagery, when implemented computationally in AI systems, sufficient for solving visual abstract reasoning tasks?
4. How can generative models be extended to imagery-based production systems that are able to produce a flexible reasoning process in computational imagery?

Chapter 3 presents a comprehensive literature review of related works to answer Research Question 1. Chapter 4 through 6 will elaborate on Research Question 1, 2, and 3, respectively, including the works that have been done and the ones that could be done in future.

CHAPTER 3

From Psychometrics to Artificial Intelligence: An Introduction to Raven's Progressive Matrices

As being widely used to measure human intelligence, Raven's Progressive Matrices tests also pose a great challenge for AI systems. There is a long line of computational models for solving RPM, starting from 1960s, either for understanding the involved cognitive processes or solely for problem-solving purposes. Due to the rapid development of AI technologies, especially the advent of deep learning models in the last decade, the computational studies on RPM have also changed a lot. Therefore, this chapter look back at this long line of research. As the title—"An Introduction"—indicates, this chapter provides an all-in-one presentation of computational models for solving RPM, including the history and intelligence testing theory of RPM, RPM-like tasks, a conceptual chronicle of computational models for solving RPM, which reveals the philosophy behind the technology evolution of these models, and suggestions for transferring human intelligence testing to AI testing.

3.1 Introduction

Most AI researchers, if not all, must have ruminated on fateful questions, which are disturbing but cannot be answered yet, such as "how far are we on the way to achieve the human-level AI?" and "how long does it take for us to fully understand the fundamental mechanism of intelligence?" Some are more pessimistic, like "will human-level AI be realized in my lifetime?" Though these questions cannot be answered for now, every AI researcher is glad to see these questions being raised and attempts being made to answer them, because, whether optimistic or pessimistic, these questions represent the scientific conscience of AI research.

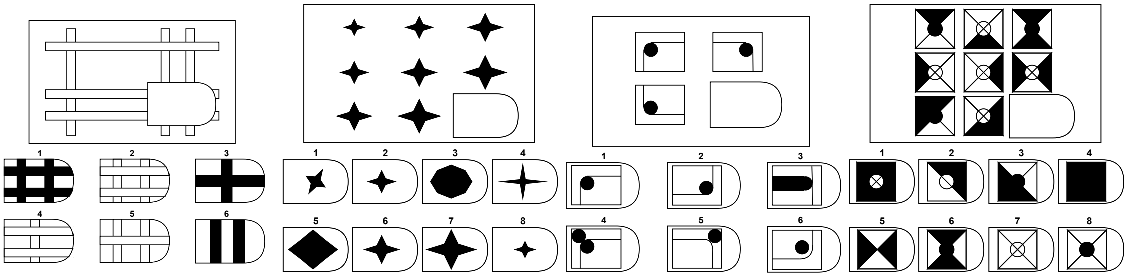
Works to answer these questions are mainly centered around comparing AI systems and humans on daily tasks that are considered indicators of intelligence. Among these works, the most direct way is to evaluate AI systems on human intelligence tests. The scope of intelligence tests is larger than the ability tests used in clinical setting. For example, SAT and MAT can be considered as intelligence tests. In addition, many developers and publishers do not name their tests intelligence tests for some people consider the word "intelligence" elitism and racism, and prefer to use more accurate words, like "tests of learning abilities", "assessment of memory and attention", and "development motor scales". Intelligence tests are usually classified into two categories—single-format tests and battery-types tests. The single-format test contains items of the same format while the batter-type test contains multiple subtests of different formats. As current AI systems require the problem format to be clearly defined, evaluations of AI systems on intelligence tests are mainly on the single-format tests or a subtest of battery-type tests.

Raven's progressive matrices (RPM) are a family of single-format tests that have been used to test AI systems in a substantial amount of works. Meanwhile, RPM has also become an impetus for developing more intelligent systems that could solve RPM as well as humans. The length of this research line dates back to 1960s; the width of this research line ranges across multiple disciplines, such as AI, cognitive science, neuroscience, psychometrics and so on. However, there has been lacking a work that inspects this research line in a joint view of its entire temporal and spatial span and establishes the theoretical depth of it. Given the recent development of this research line, we believe now it is a good point to do this a work.

We will start this work by reviewing the basics of RPM in the context of human intelligence testing in Section 3.2. The purpose of section is to answer the two theoretical questions that one would first ask about RPM—what RPM measures and how RPM measures it. The answers go well beyond the ones like "it measures human intelligence" and "it asks participants to solve problems". By answering these two questions, we intend to explain the rationale of using RPM as a human intelligence measure. We believe this is necessary for analyzing the rationale of using RPM as a AI measure, and, more generally, for establishing the theoretical foundation of AI testing.

We extend the discussion to the entire problem domain represented by RPM in Section 3.3. This domain includes several more tasks that are similar to RPM and also used for human intelligence testing and AI testing. To distinguish them from original RPM, we refer to them as RPM-like tasks. In these tasks, while items for human intelligence testing are mostly handcrafted by human experts, algorithmically-generated items are more and more useful in some special testing scenarios such as computer-based, adaptive, large-scale and/or repeated testing. Algorithmically generated items are also a realistic incentive for studies of deep learning models for solving RPM-like problems. Thus, in the second half of this section, we also reviewed the important works for algorithmic generation of matrix reasoning items, which exactly replicate the format of original RPM. In this section, we intend to (a) provide our readers with different choices of tasks and problem/data sets for different research purposes, (b) provide practical guidance for building algorithmic item generators, and (c) pave the way for the discussion of learning models in the following sections.

In Section 3.4, we propose a framework to collate all computational models for solving RPM and RPM-like tasks. We refer to this framework as a conceptual chronicle because it emphasizes the conceptual connections between computational models and the underlying logic for technological development. It is neither like the reviews that use specific taxonomies of reviewed works nor the ones that compile the reviewed works into a chronological order. Instead, it simulates the process of how a beginner's understanding of this field would naturally evolve as she knows more and more about this field. In a sense, it is more like chapter organizations of textbooks. We believe such a presentation is the best way for readers to gain a coherent understanding of this field.



(a) (b) (c) (d)

Figure 3.1: RPM examples of different formats and stimuli.

In Section 3.5, we zoom away from the computational models and address more general topics of AI testing. We first tackle the fundamental issue in this research field—i.e., the validity of using intelligence tests and similar tests to evaluate AI systems. The discussion is based on the initial idea that AI systems could be measured by these tests as human intelligence is measured by them. Unless this issue is properly resolved, the practice of applying these tests on AI systems would be restricted into pure problem solving for specific problems, rather than deepening our understanding of human intelligence and AI. Secondly, on the flip side, we also discuss the implications of human intelligence manifested on intelligence tests for building AI systems. The generalization ability and robustness of human intelligence on intelligence tests are far better than what current AI systems could achieve. We believe such a discussion is crucial for future works in this research field.

3.2 Raven's Progressive Matrices

For readers who are not familiar with RPM, Figure 3.1 shows some examples of RPM items. The original RPM tests contain items of four formats as shown in Figure 3.1. The items are presented as multi-choice problems. The context can be a single image with one piece missing (Figure 3.1a), or a 2×2 or 3×3 matrix with the last entry missing (Figure 3.1c, 3.1b and 3.1d). To solve an RPM item, one needs to select an answer from the answer set to complete the context matrix. In original RPM tests, the answer sets contain 6 choices for single-image and 2×2 matrices and 8 choices for 3×3 matrices.

Given different perceptual stimuli that populate the matrix, the item requires different cognitive abilities and skills. For example, the items in Figure 3.1a and 3.1b tap into cognitive abilities of perceptual processing. Particularly, Figure 3.1a requires processing perceptual continuity to interpolate the missing piece in (or match the answer choices to) the context image; Figure 3.1b requires processing perceptual progression to extrapolate the missing image. The other two items in Figure 3.1c and 3.1d differ from the first two because they requires not only the perceptual processing abilities, for example, perceptual decomposition and organization, but also abstract inductive reasoning, which involves constructing abstract symbols from raw perceptual stimuli and reasoning about these symbols.

Figure 3.1 represents the most typical designs of original RPM. It needs to be pointed out that RPM-like tasks are not restricted to these designs and that various designs have bee used in the RPM-like task to test different cognitive abilities and verify cognitive theories (more details in Section 3.3).

It has been claimed that RPM tests are the best single-format intelligence test that one can have. This claim is based on the statistical evidence that the test scores on RPM are highly correlated with all other common intelligence tests. RPM could be visually considered located at the center on the map of all intelligence tests (Snow et al., 1984), implying that the underlying trait behind RPM tests is also central to the traits that are measured differently. For this reason, while RPM receives much attention in clinical settings, it also receives a great deal of attention in research settings, especially in the communities of cognitive science and artificial intelligence.

3.2.1 What RPM Measures?

What RPM measures exactly? This simple question must have been haunting many researchers who are not psychologists or cognitive scientists for the first several years of their research on RPM. Well, the answer to this question may be quite straightforward to some researchers—it measures intelligence. But the others simply do not understand why these "drop in from the sky" items can tell about a person's intelligence. This question is probably better to be rephrased as "why and how does solving these problems composed of simple geometric patterns measure a person's intelligence?"

The answer is not a simple one, given the complex nature of human intelligence testing. First of all, RPM represents a type of intelligence tests that are theory-motivated. That is, the test development is inspired and guided by some abstract theories about intelligence, which involve factors that are not observable. In contrast, our stereotypical impression of tests is the ones that are related to our daily experience and pragmatic purposes. For example, SAT contains sections of writing, verbal comprehension, and mathematics because competence on them is necessary for students to perform well in college and graduate; the Armed Services Vocational Aptitude Battery contains sections of electronics, auto, shop, mechanical comprehension, and assembling objects, because these knowledge and skills are necessary for the technical positions in army. The development of these tests starts off with clear purposes and understanding of what specific behavior should be measured.

However, RPM, as an intelligence test, is to measure intelligence—a factor that is not clearly defined, directly observable, or measurable. Thus, theories have been constructed to explain the relation between intelligence and observable and measurable behavior. When RPM is not introduced to someone without clarifying the theories, she would have the question at the beginning of this subsection. In particular, John C. Raven, the author of RPM (Raven, 1936, 1941), had studied with Charles Spearman, who noticed that a person's performances on tests of different cognitive abilities are correlated and thus hypothesized that a factor—general intelligence g [1]—underlies all cognitive abilities. Spearman further pointed out that the g factor is composed of two abilities — *eductive ability* and *reproductive ability*. Eductive ability is the ability to make meaning out of confusion and generate high-level, usually nonverbal, schemata which make it easy to handle complexity. Note that the process of "eduction" is more often referred to as inductive reasoning. Reproductive ability is the ability to absorb, recall, and reproduce learned information and skills.

To test eductive and reproductive abilities, Raven developed RPM and Mill Hill Vocabulary Scale, respectively. In contrast to the pragmatic tests, the development of these tests started off with the author's personal understanding of these abilities. But, it is important to point out that the development of theory-motivated tests are not idiosyncratic because the developer needs to prove that the test indeed measures what it is expected to measure. The proof is usually achieved by collecting statistical evidence that the test score is correlated with certain measurable behavior and other tests, which are determined by the purpose of the test and interpretation of test score. For example, if the test is for recruitment, the test score should be correlated with future job performance; if the test is a general intelligence test, the test score should be correlated with cognitive ability tests and medical data such as fMRI data of the brain. In the terminology of psychometrics, the developer needs to validate the test to make sure it measures what it is expected to measure. However, the studies of RPM validity would make a new book; we simply claim that RPM is well-validated test of general intelligence.

Readers might have already noticed that there are two abilities under the umbrella of g and correspondingly two tests. What about the reproductive ability and its test? Why is RPM considered as the best single-format test for general intelligence instead of the other? Is eductive ability more important than reproductive ability? In his theory of general intelligence, Spearman did not treat these two abilities as separate factors. On the contrary, he believed that there is only a single factor—g—underlying all cognitive abilities, and eductive and reproductive abilities are two "analytically distinguishable components" of g (Raven, 2008). Eductive and reproductive abilities are better treated as two interwoven general cognitive processes, through either of which g can be measured. Since the test scores of RPM are best correlated to other intelligence tests, RPM is considered the most effective single-format intelligence test.

Now is a good point to compare to another two relevant concepts that pervade the literature of intelligence testing and our readers are probably more familiar with them. In the theory of general intelligence by Cattell (Cattell, 1941, 1943, 1963, 1987), he proposed that there are two general factors (emerging from factorial

---

[1] Spearman referred to g as general cognitive ability because he thought the word intelligence had been abused by many people.

analysis) subtending intellectual performance—fluid intelligence and crystallized intelligence. Fluid intelligence, g_f, is the ability to discriminate and perceive complex relationships when no recourse to answers is already stored in memory. Crystallized intelligence, g_c, consists of judgmental, discriminatory reasoning habits long established in a particular field, originally through the operation of fluid intelligence, but no longer requiring insightful perception for their successful operation. The definitions of fluid and crystallized intelligence resembles the ones of eductive and reproductive abilities. Moreover, fluid and crystallized intelligence are frequently used as synonyms of eductive and reproductive abilities in literature. But these two sets of terms are conceptually different. In particular, Spearman considered eductive and reproductive abilities as two components, while Cattell treated fluid and crystallized intelligence as factors. When we say components of a system, we mean that the components must work together for the system to work; if either of eductive and reproductive component does not work, the whole system does not work. But when we say factors (especially in factorial analysis), we mean different dimensions that each exert separable influence on experimental outcome and thus can be studied separately. We can calculate what percentage of the variation in the data is caused by which factor (using procedures in analysis of variance), but it is conceptual wrong to do so in component systems because the components' influences are not separable. Note that this does not mean that factors are completely independent because two factors can still correlate and jointly contribute to a proportion of variation. A good example is height and weight of an athlete, which are correlated, but still two different concepts and factors. As factors, their private and shared contribution to athletic ability can be determined statistically if we collect data of athletes. Therefore, when we are using these two sets of terms interchangeably, we need to be clear about which theoretical assumption about them are made and thus have different conclusions for the experiments if necessary.

Beside the conceptual issue behind terminology, another issue is that the boundary between theory-motivated and pragmatic tests is not so clear in practice. As more and more research is conducted on a pragmatic test, theories will be invented to explain human responses on the tests. Similarly, as a theory-motivated test is proven to be a valid measure for some mental trait, it is also possible to use it for pragmatic purposes. For example, RPM was once used for military recruitment in UK during World War II (Burke, 1958).

3.2.2 A Brief History of RPM

This review would be an incomplete one if we did not say something about the history of RPM, which is almost 100 years long. Admittedly, not every detail of this history is relevant to our research of RPM in the context of AI. However, the development of RPM in human intelligence testing would provide potential enlightenment for the future of AI testing, which is largely undefined yet. We will introduce the whole family of RPM in this subsection[2], and discuss the the motivation behind each RPM test and the connection between them.

Raven (1936) developed the first RPM test in 1930s when he was studying with Lionel Penrose, who was a geneticist and psychiatrist. This test was used to study the genetic and environmental determinants of intellectual defect. As other genetic studies, this study required a large population of subjects, including adult parents and children at all ages, being tested at different places, such as home, school, and workplace. It is, therefore, infeasible to administer full-length intelligence tests, such as Binet tests and Wechsler tests, which require hours for a session. In addition, because some subjects then were illiterate and many workplaces were too noisy for verbal questions, the testing items had to be nonverbal and as self-evident as possible. These practical requirements together led to the design of the first RPM test.

As we have mentioned, the development of RPM was theoretically inspired by the Spearman's theory of intelligence. Although the theory is instructive for understanding intelligence, the overarching *g* factor is a latent variable, which is not directly observable and measurable. This makes its measurement inherently complicated because one needs to identify the measurable activities and decide how they relate to the latent variable, for example, it can be calculated by weighting scores on multiple cognitive ability tests. To simply its measurement, Raven mentioned in his personal notes that he intended to develop "a series of overlapping homogeneous problems whose solutions required different abilities" (Carpenter et al., 1990). In particular, these items are homogeneous in the types of perceptual stimulus and abstract relations, but their difficulty varies in a wide range. If these homogeneous items are arranged evenly in an increasing order of difficulty, they together will form a ruler of intelligence. That is, a subject is less likely to be able to solve an item if she cannot solve the items before it. As the test is administered to more and more people and more data are collected, the item difficulty is determined more accurately (relative to people's ability to solve it; through psychometric procedures). Now, the outcome of this multi-ability, homogeneous, and increasing-difficulty design is that we can measure the latent variable *g* with a single single-format test. Intuitively, the RPM tests make the *g* factor directly measurable and the scores more interpretable as we use a tape measure to measure height and a thermometer to measure temperature.

RPM is a family of progressive matrices tests, including three main tests—Standard Progressive Matrices (SPM), Coloured Progressive Matrices (CPM), and Advanced Progressive Matrices (APM), and each test has multiple versions consisting of different items. The first RPM test is the SPM test published in 1938 (Raven, 1941), which is the ancestor of all the following RPM tests. Including the first version of SPM, all the SPM tests are composed of 60 items, which are organized into 5 set (A, B, C, D, and E) according to their difficulty.

---

[2] This subsection is mainly based on the manuals of RPM tests. For readability, we will not insert citations of the manuals in this subsection. Otherwise, it would be everywhere.

The item difficulty increases within each set and from Set A through Set E. Meanwhile, each set has a distinct theme manifested by the perceptual stimuli and conceptual relations of items in this set.

To spread the scores and have a better precision at the lower and upper ends of the ability range, the first versions of CPM and APM were developed and published in 1947. CPM reused the Set A and B of 1938 SPM and placed a transitional set of 12 items—Set Ab—between Set A and B. The items in this set were constructed to be intermediary in difficulty between Set A and Set B. Thus, CPM has had 36 items organized into three sets. As the name indicated, CPM is printed in color to appear interesting as it is often administered to children under 11. CPM can also be administered to mentally retarded persons, the elderly, and people with brain injury. Different from SPM and APM, CPM was published in two forms — the book form (i.e., a the paper-and-pencil test) and the board form. In the board from, each item is a board with a part removed and movable pieces as answer choices to complete the board. The board form has been proved to be equivalent to the book form, tapping the same cognitive process. Moreover, the board form has two advantages. First, the board form can be better administered without verbal instruction because the administrator can demonstrate the expected response by manipulating the board and answer pieces. This is important for people who are deaf people or unable to communicate for some reasons.

The APM was originally drafted in 1943 for use by the British War Office Selection Boards, who needed a more difficult RPM test that could provide better discrimination at higher levels than SPM. The APM test was published in 1947, consisting of two sets — Set I and Set II. Set I comprises 12 items covering all themes and sampled on the full test of SPM. In practice, Set I can be used to familiarize people with the test, sorting people into the "dull" 10%, "average" 80%, and "bright" 10%, and decide whether SPM or Set II should be used next. The 1947 Set II consisted of 48 items, which resembled the items in Set C, D, and E of SPM in presentation and argument. In 1962, 12 items making no contribution to the score distribution were dropped from Set II and the remaining 36 item re-arranged.

In the last decades, there has been a significant and steady increase in many intelligence test scores, including the SPM scores. Among all RPM tests, SPM is designed to cover the widest ability range. But this increase causes SPM to be less discriminative at upper levels of ability range (i.e., ceiling effect). In 1998, a new SPM test—SPM plus—was published to restore its discriminative power at the upper levels, and, meanwhile, keep its discriminative power at the lower levels unchanged. In particular, SPM plus includes all the items in Set A and B of SPM and replaces moderately difficult items with more difficult items in Set C, D, and E.

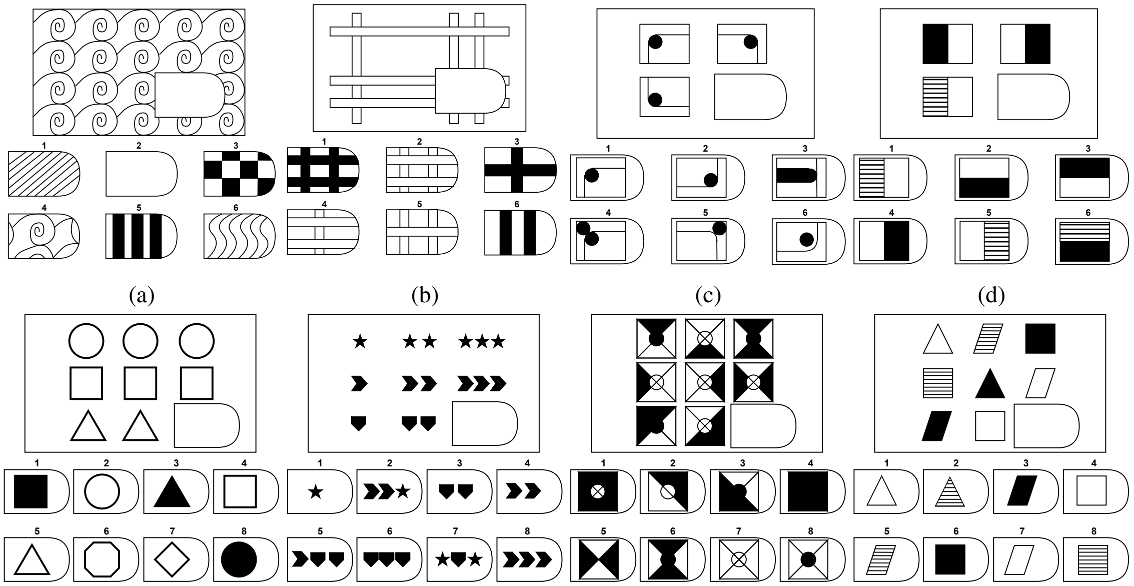
As a result of its simple self-evident format, insensitivity to culture and language, and centrality among all intelligence tests, RPM has been the most widely studied single-format intelligence test and has large amounts of testing data available for research. This, however, raises a concern that the test has become too well known and the participants could be coached for solving them or memorizing the answers. This is problematic when important decisions (such as educational opportunity and job recruitment) are made upon the test results. Therefore, parallel versions of CPM and SPM was developed in 1998. These versions are designed to be parallel to the classic SPM on an item-to-item and overall score basis so that the existing data of classic SPM and CPM could be used to analyze the data of the parallel versions.

The administration procedure of RPM tests is relatively flexible compared to other intelligence tests. RPM tests can be administered both individually and in groups. In individual test, one administrator guides one participant through the test. In group test, one administrator proctors the participants as in normal school exams. Individual tests introduce emotional factors which are not present in group testing or self-administration, and thus the scores are slightly lower than group tests, in which participant work on their own. But individual tests allow the administrator to make sure the participant understands what to do and observe the participant to collect more data, such as whether the participant uses a trial-error strategy. Thus, individual test is recommended when important decisions are made upon the test result. In both group and individual tests, instructions can be given verbally or using gestures such as pointing, nodding, and shaking head. In most cases, RPM tests are given in an untimed manner or with sufficient time to attempt every item since, when timed, the validity of scores is reduced according to statistical evidence. Moreover, it has been argued that RPM is neither a speed test nor a power test, or a combination of both. There is an exception that, after familiarization with Set I of APM, Set II of it was administered with a time limit to measure the speed of intellectual work.

To sum up, RPM is a big family of tests, including SPM, parallel SPM , SPM plus, CPM (with two forms), parallel CPM, and APM. All the RPM tests that are used today have gone through many revisions as more and more data are collected in different countries and from different groups of people. There also exist different procedures to administer the tests, which result in qualitative different results. When studying RPM in the context of artificial intelligence, it is important to point out which RPM test is used and how it is used in terms of the administration procedures.

3.2.3 What RPM measures exactly?

At the beginning of this section, we have tried to answer the question “what RPM measures” from a theoretical perspective. In short, RPM measures eductive ability, which is a component of general intelligence (i.e, the *g* factor or genera cognitive ability), and thus can be used as an index of general intelligence. However, the answer is still too abstract and does not land on the concrete items in RPM tests. To be honest, the answer at the beginning could apply to almost every test of eductive ability, fluid intelligence, or general intelligence. To tell the whole story of RPM, we further reify the answer by inspecting the concrete items and the administration procedures.


(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 3.2: Example SPM item series.

We have indicated in previous section that the test design is the outcome of an iterative process, in which the revised tests are repeatedly administered to people so that data can be collected to further revise the test.Since RPM is also a theory-motivated test, the test design is also determined by by the theory of intelligence and how it is implemented in the test. We take SPM as an example. To protect the secrecy of RPM tests, we created several new items (Figure 3.2) that simulate the item series in SPM. As mentioned, there are five sets in SPM (Set A, B, C, D, and E). The eight items in Figure 3.2 simulate the way how the item design varies from the first item of Set A to the last item of Set E. At the beginning of Set A, a participant will see an item similar to the one Figure 3.2a. The role of this item is to give the very basic idea of the test. This item is a good starting point in that no prior knowledge is required to solve the item and its solution is self-evident to almost every participant. In the standard administration procedure, this item is used for teaching trial. The administrator explicitly tells (possibly in a nonverbal way) the participant that “only one of answer choices can complete the pattern correctly” and which one it is correct for this item.

Note that in every administration procedure in the manual of RPM tests (individual or group, verbal or nonverbal), the administrator only tells the participant which answer choice is correct, but never explains why it is correct or the thinking process to solve it. This point is extremely important for the testing to be valid. The teaching trials are to help the participant with the format of the test, i.e., one needs to select an answer to complete the pattern, but not the content of the text, i.e., what pattern it is and how it is completed. The content part is just what the test measures—eductive ability. An even stronger but similar argument (Raven,

2008) is that it is not correct to describe RPM items as “problems to solve”. The instruction that an answer has to be selected does not means that it is a problem. Instead, only when the participant has made some meaning out of the item can the participant sees the item as a problem to solve. The meaning-making part is the core of RPM items, which measures the eductive ability.

After the items for teaching trials, the participant will see an item similar to the one in Figure 3.2b. This item takes an important transitional role that shifts the participant’s attention from the test format to the test content. In particular, this item explicitly exhibits the nature of the test content—relational reasoning. That is, to solve the following items, the participant needs to consider the relations between the objects rather than, for example, repeating the raw perceptual input in the teaching trials. In addition, the transitional role also lies in the appearance of the items: the teaching-trial items and the transitional items are not presented as matrices, but the transitional items are one step closer to the matrix structure in the following items (see Figure 3.2c through 3.2h), because the relations in transitional items happens in both the horizontal and vertical directions. These transitional items are necessary because they assure that the participant give valid responses to the following items based on the understanding accumulated in the previous items.

After the transitional items, the test enters 2×2 items like the one in Figure 3.2c and 3.2d, in which, geometric objects are separated into the disconnected matrix entries. These 2×2 matrices start with the ones that more rely on low-level perceptual processing (Figure 3.2c) and are relatively easy. After the participant is familiar with the format of 2×2 matrix, it and gradually move on to the ones that involves more abstract relations (Figure 3.2d) and are thus more difficult than the perceptual items.

The four items in Figure 3.2a through 3.2d represent the test design in the first two sets of SPM. The following three sets follow the same logic—each item is like a rung of a ladder that makes it possible for the participant to step on the next rung, and the maximum height the participant can reach depends on her strength for climbing the ladder. As a real ladder rung, an item cannot be two far from the previous one. For example, the participant will find an item similar to the one in Figure 3.2e, which is used to introduce the 3×3 structure. This item only differs from some items in Set A and B in the matrix size but underlying perceptual processing remains the same. After the participant gets familiar with the 3×3 structure, SPM moves on, as in the Set A and B, from perceptual items to the items that involve more abstract relational concepts, such as number (Figure 3.2f), binary logical operation (Figure 3.2g), and ternary permutation (Figure 3.2h). Moreover, the number of relations in item also gradually increases in the last three sets of SPM. For example, the items in Figure 3.2e, 3.2f, and 3.2g each contain only one relation; the item in Figure 3.2h contains two relations—permutation of object shape and permutation of filling texture.

The example series in Figure 3.2 epitomizes the design of SPM. Through this example, we can see the motivation behind the test design is to provide an ability ladder for the participant to climb. The rungs/items are distributed evenly so that the ladder is climbable. Furthermore, the ladder is climbable to participants for people at every ability level since it starts from the "ground"—i.e., the beginning trivial items requiring no prior knowledge—and guides the participant to move in the expected direction through conceptually connected items. Once the "field of thought" is established, how far the participant can go depends on her ability in this field.

In a sense, SPM is different from problem-solving tests that everyone has taken at school. Instead, SPM is a miniature that simulates a collection of all tests from elementary level to college level because one need to graduate from every level sequentially. Although the duration for these two types of testing is vastly different, both of them measure the learning potential of the participant. Note that the word "potential" here is more suitable than "ability" because the "potential" means a latent quality that develops under the influence of environmental factors. Since the environment factors can be better controlled in intelligence tests than in the education system, SPM is probably a better measure of learning potential. Moreover, potential is more than ability since the desire to learn knowledge and the courage to conquer new problems are also part of potential.

In general, RPM is much more than problem solving. Even the word "test" is misleading because of our stereotypical impression of test. RPM tests are a system for evaluating eductive ability through measuring the learning potential. However, the common practice of using RPM or RPM-like tests as purely problem-solving tests and making extravagant claim about corresponding abilities of AI systems in many AI studies have been a big misuse of these tests.

3.3 RPM-Like Tasks

In this section, we extend our discussion to the entire problem domain represented by RPM, which includes RPM-like tasks that inherited the basic elements of original RPM tests and implemented them in more enriched manners. Such RPM-like items can be found in almost every modern intelligence test. In contrast to the theoretical analysis in the last section, we take a more pragmatic approach in this section to describe these tasks. In particular, We surveyed four intelligence tests [3] that are widely used in clinical setting and/or frequently related to RPM in literature—Cattell's Culture Fair Intelligence Test (CFIT), Cognitive Assessment System-Second Edition (CAS2), Wechsler Adult Intelligence Test-Fourth Edition (WAIS-IV), and Leiter International Performance Scale-Revised (Leiter-R). Through this survey, we summarized five tasks in the problem domain—matrix reasoning, figure series, analogy making, contrastive classification, and open classification. In addition, We further survey the methods for algorithmically generating matrix reasoning items, which are a prerequisite for the discussion in the following sections of data-driven AI models for solving

---

[3] There are many more important intelligence tests, such as Kaufman, Stanford-Binet, and Wookcock-Johnson tests. But because of the limited access to these commercial tests and resemblance among their RPM-like items, we surveyed only four of them.

RPM-like tasks. As we have mentioned, the items in intelligence tests are mostly handcrafted and thus in a very limited number, which is far below the need of current data-driven models. This section could provide options of existing RPM-like items and suggestions of algorithmically creating new RPM-like datasets for different research purposed.

3.3.1 RPM-like Tasks in Intelligence Tests

Although the theories of intelligence behind the four intelligence tests are different, the RPM-like tasks in these tests are consistent to some degree in terms of what is measured. For example,

- the RPM-like tasks in CFIT measures the general cognitive ability, i.e., the *g* factor, and stresses that the *g* factor "reaches its purest expression, i.e., high *g* loading, whenever complex relationships have to be perceived" (Cattell, 1950);
- the RPM-like tasks in CAS2 measures the simultaneous processing ability in the PASS theory of intelligence (Das et al., 1994), i.e., the ability to "integrates stimuli into (conceptually) interrelated groups or a whole" (Naglieri et al., 2014);
- the RPM-like tasks in WAIS-IV "involves fluid intelligence, broad visual intelligence, classification and spatial ability, knowledge of part-whole relationships, simultaneous processing, and perceptual organization" (Wechsler et al., 2008);
- the RPM-like tasks in Leiter-R measure "fluid reasoning, deductive and inductive reasoning, and the ability to perceive fragments as a whole, generate rules out of partial information, perceive sequential patterns, and form new concepts" (Roid and Miller, 1997).

From these descriptions of RPM-like tasks in these tests, we can see that they all more or less involve measuring eductive ability or fluid intelligence. Given this internal connection between RPM-like tasks, it would be unsurprising to see common elements shared between them. For perceptual elements, to distinguish eductive ability (or fluid intelligence) with reproductive ability (or crystallized intelligence), they must not be unique to certain cultural groups. There are not too many choices satisfying this requirement, for example, elements from nature like sun and moon, human body (hand and foot), and common shapes and colors. Similarly, common conceptual elements, such as symmetry, topological relations, and number concepts, are also frequently used to create RPM-like items. Now, it is already very hard for test developers to design novel elements for RPM-like items because most of the appropriate elements have already been used in intelligence tests. If one comes up with novel perceptual and conceptual elements that can be used in RPM-like tasks, it will be a great contribution to intelligence test development. Exploration for proper perceptual and conceptual elements for RPM-like tasks is also helpful for building and evaluating AI systems working in this problem domain.

In addition to perceptual and conceptual elements, there are different formats to present these elements. According to these formats, we classify the RPM-like tasks in the four intelligence tests surveyed into five groups—matrix reasoning, figure series, analogy making, contrastive classification, and open classification. These formats are equally interesting to the perceptual and conceptual elements, as each format is a delicate way to present the same set of elements so that they can be instantly perceived as a problem to be solve but not a trivial one.

3.3.1.1 Matrix Reasoning

Since the the four tests are battery-type tests, they all have multiple subtests, including the RPM-like subtests. Therefore, to keep the whole test in a reasonable length, the RPM-like subtests are briefer than the original RPM tests. In particular, these RPM-like subtests do not necessarily implement the "ladder" design mentioned in Section 3.2.3, which is an import feature of the original RPM tests. Nevertheless, three of the four tests surveyed contain subtests that replicate the matrix format of original RPM: Test 3 of Scale 2 and 3 of CFIT, Matrices of CAS2, and Matrix Reasoning of WAIS-IV. To distinguish them with the following RPM-like tasks that we will discuss in later sections, we refer to them as matrix reasoning. Figure 3.3 summarizes matrix reasoning tasks through a diagram.

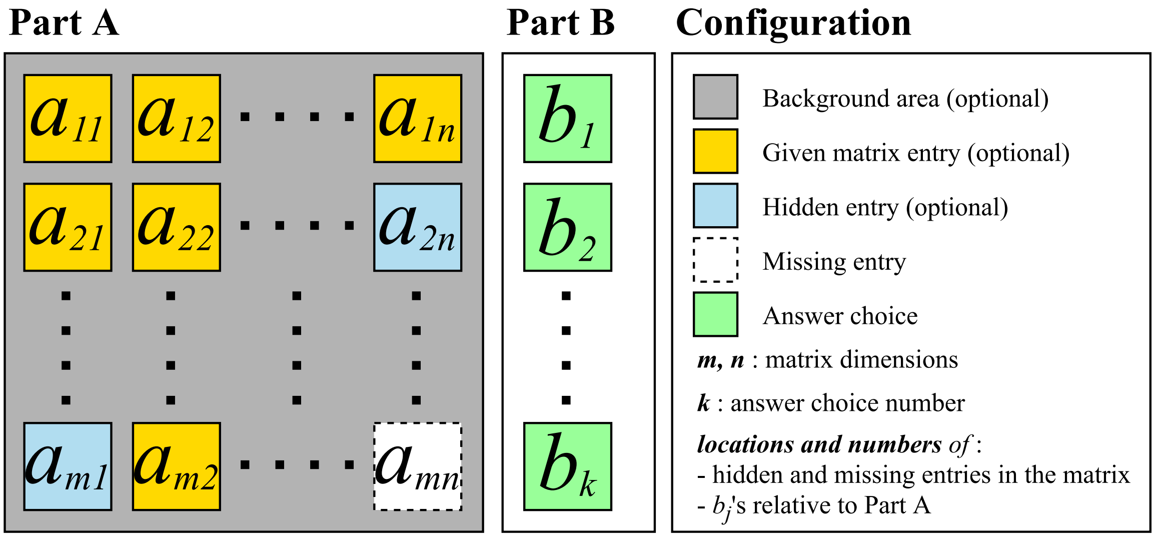

Part A
Part B
Configuration
a_{11}
a_{12}
a_{1n}
a_{21}
a_{22}
a_{2n}
a_{m1}
a_{m2}
a_{mn}
b_1
b_2
b_k
Background area (optional)
Given matrix entry (optional)
Hidden entry (optional)
Missing entry
Answer choice
m, n : matrix dimensions
k : answer choice number
locations and numbers of:
- hidden and missing entries in the matrix
- b_j's relative to Part A

Figure 3.3: A diagrammatic summary of matrix reasoning task

As shown in Figure 3.3, there exist two parts in a matrix reasoning item—the context of this multi-choice problem (Part A) and answer choices (Part B). Part A provides the contextual information through a background and a matrix as foreground. Examples of the background can be found in the items of Figure 3.2a and 3.2b. The matrix varies in size from 1×1 to 4×4 in most tests and has at least one missing entry. To increase the difficulty, there can be some entries, which are intentionally hidden but need not to be completed.

As indicated in the Configuration in Figure 3.3, the locations and numbers of these two types of entries can also be customized for each item. Part B consists of 5 to 8 answer choices in most tests. The reason that we separate answer choices from the context is not only that their functions are different but also that where answer choices are located relative to the context has an influence on the distribution of choosing each answer choices according to human experiment data. Therefore, this is a design choice that need to be considered in test development. This is also a noteworthy point when evaluating AI on RPM-like tasks. That is, it requires more investigation if AI systems behave differently when answer choices are located differently relative to the context and relative to each other.

Although the matrix reasoning tasks replicate the format of original RPM (with slight modifications such as hidden entries and different locations of missing entries), the content of them are more diverse than original RPM. For example, the difficulty of original RPM mainly lies in extracting conceptual relations, and the requirement for perceptual processing is relatively low; but, due to different underlying theories about intelligence, some RPM-like items are designed to load more on perceptual processing abilities, for example, mentally rotating complex 3D objects and the abstract conceptual relations are built upon such demanding perceptual processing.

3.3.1.2 Figure Series

Essentially, what makes RPM items meaningful testing questions is the relations between figures and how these relations are arranged in the 2D structure of matrices. There is no particular reason for using matrix structure. That is, as long as the spatial structure makes sense to the relations, one can use any suitable spatial structures (one could use a circular structure if the relations proceeds and comes back to itself, like modulo addition $+1 mod N$ and the circle of music keys). Thus, it would not be surprising to see a more fundamental structure—series—to be used in RPM-like tasks, such as Test 1 of Scale 2 and 3 of CFIT, Sequential Order and Repeated Pattern of Leiter-R, and part of Matrix Reasoning of WAIS-IV. We refer to RPM-like items of this structure as figure series. A diagram was given to summarize figure series items in Figure 3.4.

Figure series has the several characteristics that make it different from other RPM-like tasks. First, the structure of series determines that one or more relations are repeating themselves along the series. Note that the relation is not necessarily a binary relation and it could involve more than two consecutive entries in the series. Second, to provide sufficient contextual information, the figure series are usually longer than a row or a column of matrix reasoning. Third, there could be one or more missing entries in the series. In particular, the missing entry is not necessarily the last one.

Figure series could also be considered a special case of matrix reasoning task by restricting the dimensions of the matrix, but it is also conceptual different from matrix reasoning task. In matrix reasoning, there

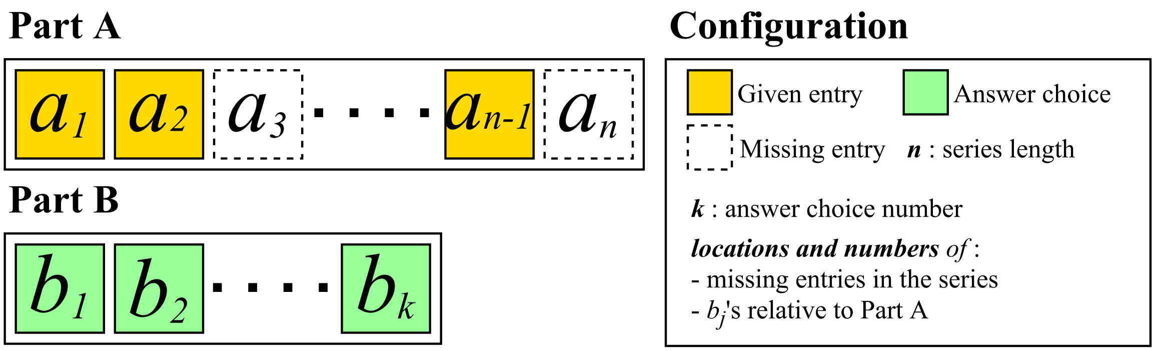


Figure 3.4: A diagrammatic summary of figure series task

can be multiple distinct relations along the rows and columns of the matrix. In most cases, the row relations are different from the columns one. One needs to figure out the relations in both row and and columns directions and assemble them to uniquely determine the answer. In figure series, multiple relations are repeating themselves in a single direction.

3.3.1.3 Analogy Making

Besides modifying the format of original RPM (as in figure series), the context of it could also be viewed from different angles. An important view is from an important human cognitive ability—analogy making. That is, by viewing the matrix entries as analogs, analogies can be drawn between rows, between columns, or between diagonal lines. The correct answer is thus the one that makes the best analogies out of the matrix. Therefore, the nonverbal analogy-making task could be considered as a close relative of RPM. A classic example of this task is the goemetric analogy problems (find images in (Lovett et al., 2009)) published in the 1942 edition of the Psychological Test for College Freshmen of the American Council on Education. These analogy-making items can also be found in intelligence tests we surveyed, such as Design Analogy of Leiter-R and part of Matrix Reasoning of WAIS-IV. A diagrammatic summary of this task is given in Figure 3.5.

In the analogy-making task, the context is explicitly separated into two parts, Part A and A' in Figure 3.5, which are composed of analogs from two different domains. Part A and A' correspond to the base and the target domains in general analogy making situation, where the base domain is usually a familiar one and the target domain is an unfamiliar one which is to be understood through the knowledge in the base domain. The analogy-making task simulates this situation by arranging the analogs in Part A and A' in the same way and removing one or more analogs in the Part A'. Note that, although the analogs in Figure 3.5 are listed in series, this does not mean that the same relations are repeating itself in the series as in Figure series. The analogs could be arranged in any spatial layout when the layout make senses to the relations between analogs. Since the analogs are usually in two series in most intelligence tests, the analogy making task resembles the

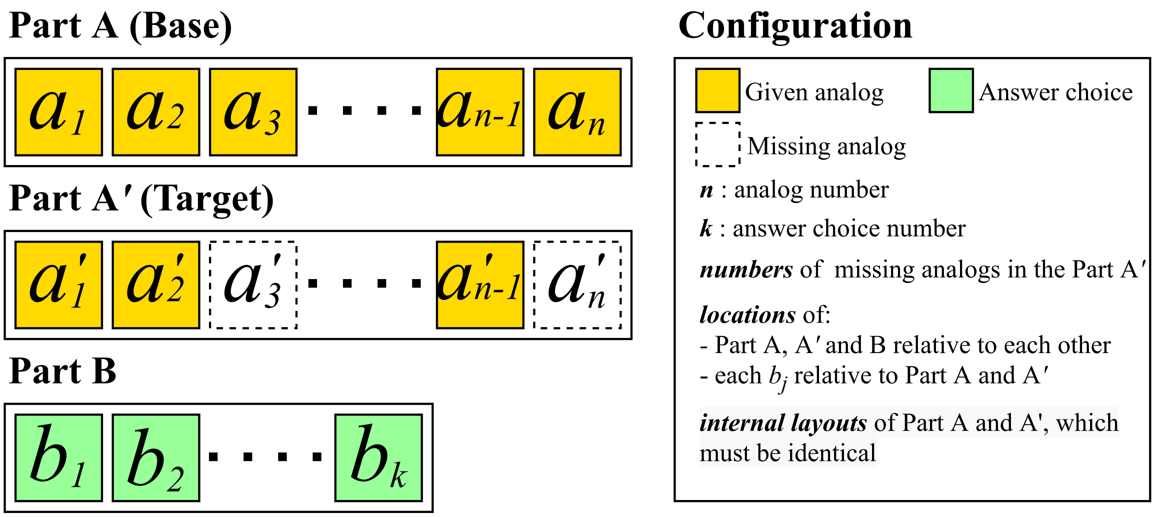


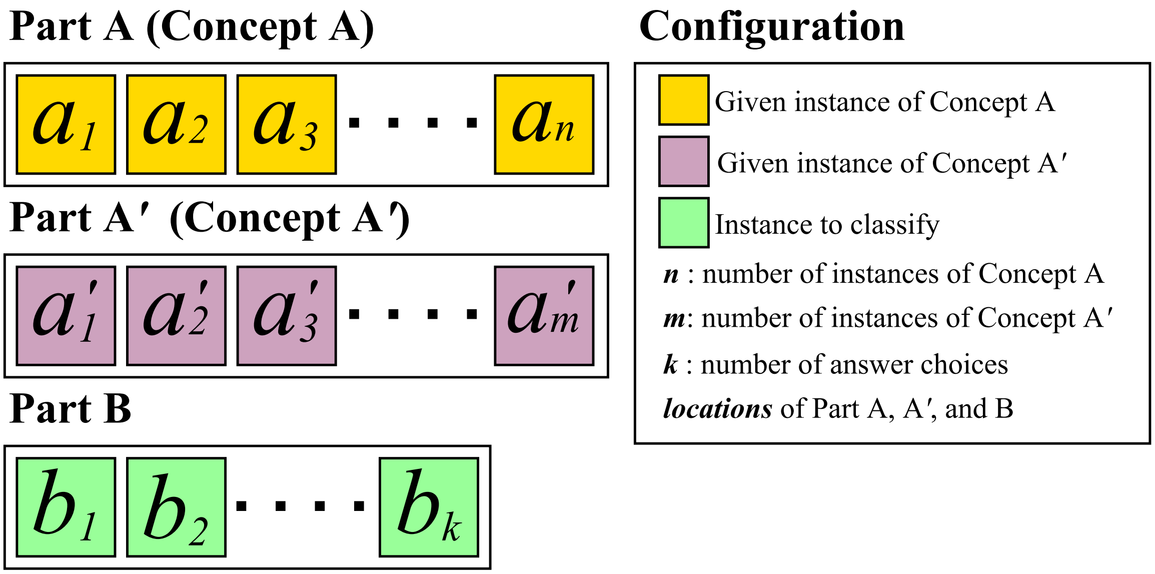
Figure 3.5: A diagrammatic summary of analogy making task

figure series task. But these two tasks are conceptually different and requires different cognitive abilities. The analogy-making task is also conceptually different from matrix reasoning task even when we artificially separate the rows or columns of a matrix into two parts. This is because, to make an “interesting” analogy, the base and target domains must be perceptually distant from each other and higher-order relations must be extracted from both domains. In matrix reasoning, this means that the rows (or columns) must be sufficiently perceptually different. These conditions are not always satisfied in matrix reasoning items, especially when there exist relations in both horizontal and vertical directions.

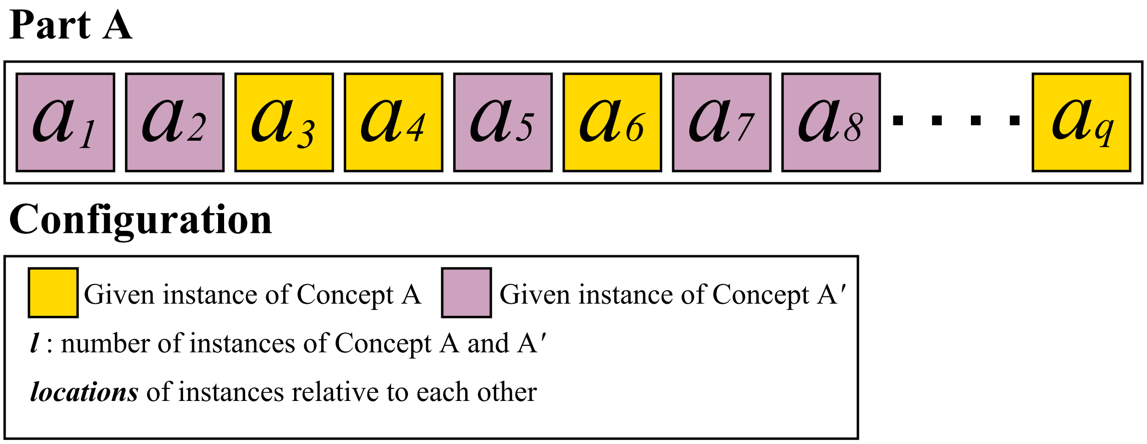
3.3.1.4 Contrastive Classification

Classification has long been used to probe human and artificial intelligence. It requires the participant to extract an abstract concept such that the given stimuli can be classified into these concepts. When these stimuli are like the ones in RPM, classification can be regarded as an RPM-like task as they both reason about the relation between multiple visual stimuli. In intelligence tests, classification tasks can be presented in a contrastive manner. That is, two groups of stimuli are presented and the two groups represent two contrastive but related concepts, for example, large-small, concave-convex, and high-low. But note that classification is not limited to antonym pairs for it also uses concept pairs like pentagon-hexagon and more random concepts like topological structures. The advantage of being contrastive is obvious: it allows the usage of complex and diverse concepts (rather than simple concepts describing perceptual attributes) to make the test intellectually interesting to participants; meanwhile, the complex and diverse concept would not make the item too open to solve as the concept is uniquely determined by a unique difference between the two groups.

The most representative contrastive classification is the Bongard Problems. It requires the participant to


Part A (Concept A)
a_1 a_2 a_3 .... a_n
Part A′ (Concept A′)
a'_1 a'_2 a'_3 .... a'_m
Part B
b_1 b_2 .... b_k
Configuration
Given instance of Concept A
Given instance of Concept A′
Instance to classify
n : number of instances of Concept A
m: number of instances of Concept A′
k : number of answer choices
locations of Part A, A′, and B

(a) Explicit contrastive classification

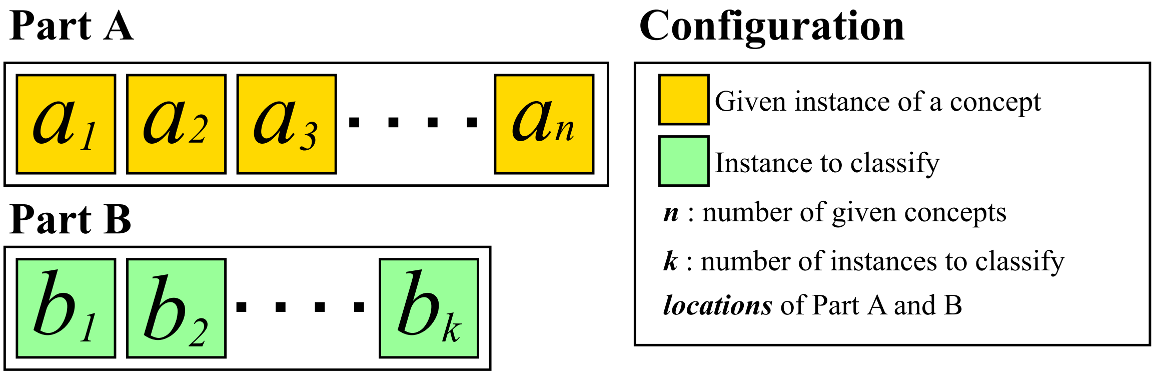

Part A
a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 .... a_q
Configuration
Given instance of Concept A
Given instance of Concept A′
l : number of instances of Concept A and A′
locations of instances relative to each other

(b) Implicit contrastive classification

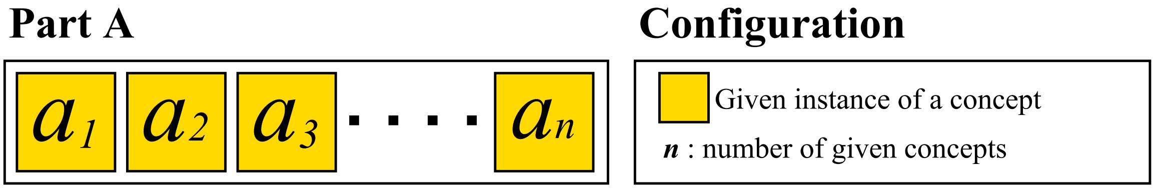
Figure 3.6: A diagrammatic summary of contrastive classification task

verbally describe the conceptual difference between the two groups. In most intelligence tests, contrastive classification is usually multi-choice problems, in which answer choices are selected to be a member of a conceptual group, i.e., identifying instances of the concepts drawn out of the two groups. The contrastive classification are usually presented in two manners—explicit and implicit ones. For explicit ones (Figure 3.6a), the two stimulus groups are explicitly separated, for example, the Bongard Problems and Test 2 of Scale 1 and CFIT. Explicit contrastive classification tasks are also used to evaluate AI system, for example, the SVRT and PSVRT datasets (Stabinger et al., 2021). For implicit contrastive classification (Figure 3.6b), the stimuli from two conceptual groups are mixed together and the participant needs to separate the them into two groups, like the famous Odd-One(s)-Out tests and Test 2 of Scale 2 and 3of CFIT. Note that, in contrastive classification tasks, spatial layout of stimuli is less important compared to matrix reasoning and figure series. The only requirement is that group membership is clearly indicated in explicit contrastive classification.

3.3.1.5 Open Classification


Part A
a_1
a_2
a_3
a_n
Part B
b_1
b_2
b_k
Configuration
Given instance of a concept
Instance to classify
n : number of given concepts
k : number of instances to classify
locations of Part A and B

(a) Explicit open classification



(b) Implicit open classification

Figure 3.7: A diagrammatic summary of open classification task

Classification task is naturally not contrastive in our daily cognitive activities. That is, the object to classify is not always accompanied by instances of another contrastive concept. Instead of being contrastive, the real-life setting of classification is more based on perceptual and conceptual similarity. Thus, we referred to it as open classification. In particular, the concepts involved in an open classification item can be completely unrelated. There could be only one single concept. For example, in the verbal similarity subtest of WAIS-IV, one would see an item like “in what way are dolphins and elephants alike” [4]. A possible answer is that they are both animals and a better answer is that they are both mammals. Different answers are scored differently. The more specific the answer, the higher the score. As shown by this example, verbal open classification items require a certain amount of prior knowledge to be intellectually interesting. When open classification is in nonverbal form, it could be considered as a RPM-like task. In the intelligence tests we surveyed, examples of nonverbal open classification include Test 4 of Scale 2 and 3 of CFIT and Classification Subtest of Leiter-R.

Similar to the contrastive classification, the open classification can also be presented in explicit or implicit ones, as summarized in Figure 3.7. The explicit open classification (Figure 3.7a) consists of two parts. Part A provides instances of multiple concepts (not necessarily contrastive or even related) with each instance representing a distinct concept. Part B consists of instances to classify into the concepts in Part A by matching to the instances in Part A. The implicit open classification (Figure 3.7b) is similar to the verbal open classification example except that the dolphins and elephants are replaced by nonverbal stimuli. The response format and scoring are also similar to the dolphin-elephant example.

3.3.1.6 Summary

The five categories of RPM-like tasks that we summarized from the intelligence tests are by no means comprehensive. The purpose of them is to expand our attention to the entire problem domain represented by RPM so that the AI research is closer to the nature of the problem domain rather than focusing only solve the original RPM or specific tests. The problem domain is much more diverse and larger than the approximately 100 original RPM items. The problem domain spreads out to all visual stimuli and relations among them that are proper to test people with certain prior knowledge and experience.

In item writing of intelligence test , a good “taste” is extremely important. Firstly, a good item first has to be straightforward for the participant to realize that this item is a problem to solve. This point seems saying nothing because any intelligence test item is a problem to solve. The word “problem” here should not be understood literally. In particular, the item is a problem to solve not because the the administrator tells the participant it is so or the participant knows that a test is composed of problems. Instead, the participant should realize this by observing the item and forming a conjecture that there should be underlying patterns based on the observation. This conjecture is more of feeling rather than a complete understanding of the solution or patterns, which means that it is based on a rough idea of what should be paid attention to solve the item. This characteristic to give the participant this feeling is important because it makes the item intellectually interesting and attractive to the participants and the participant is thus motivated to solve the item. Without this

---

[4] This item requires the participant to classify the objects into one of the many concepts that she knows.

characteristic, the participant would possibly give invalid responses, for example, giving random responses without thinking.

The second point in item writing is that the scope of item content should allow a large range of difficulty. Specifically, it should allow to create rather difficult items to test highly intelligent individuals. This point in itself is not an issue because there exist a huge amount of sophisticated abstract relations and patterns if one delves into any specific field. But, when combined with the first point—straightforward as a problem, this poses a great challenge because these points are contradicting to each other in many cases. A master of item writing is one who can reconcile these two points and achieve a combined effect that when the participant sees the item, she immediately understands in what way it is problem to solve and invests effective intellectual effort to solve it, and when a correct answer is reached, it would be an aha moment that the participant strongly believes that the problem is solved. In this sense, the five categories of RPM-like items mentioned above are masterpieces of item writing. But this does not mean that the problem domain is limited to these categories, and more efforts are needed to further explore the problem domain.

3.3.2 Algorithmic Item Generation of Matrix Reasoning

Algorithmic Item Generation (AIG) refers to approaches using computer algorithms to automatically create testing items. AIG was initially introduced to address the increased demand for testing items in the special test settings:

- Large-scale testing, for example, repeated tests in academic settings and longitudinal experiments, where many parallel forms are needed due to the retest effect.
- Adaptive testing, in which the next items are determined by the responses to previous items, which is a more efficient and reliable testing form, but also requires larger item banks.
- Computer-based and internet-based testing, which makes standardized tests more accessible to the public and brings the exposure control issue to a new level.

For AIG to work, test developers must have a deep understanding of what is measured and the corresponding problem domain, from which items are generated. In addition, test developers also need to examine the testing properties of generated items, such as validity and reliability, as they are examined in handcrafted tests. AIG has been studied and used in different areas, such as psychometrics, cognitive science, and education. It can be used to a wide range of testing items from domain-general tests, such as human IQ tests, to domain-specific tests, such as medical license tests (Gierl et al., 2012).

As RPM-like tasks are more and more used in human intelligence testing and AI testing, the demand for RPM-like items has been increasing rapidly. In particular, since data-driven AI systems were applied on

RPM-like tasks, the scale of this demand has been changed from hundreds to millions, which is impossible for human item writers to satisfy. Thus, AIG of RPM-like items has been receiving more and more attention. However, AIG of RPM-like items have been studied separately in different research fields. In this subsection, we aggregate these works from different fields together and systematically explore how AIG of RPM-like items works in both human intelligence testing and AI testing. To have a thorough discussion on technical details and theoretical implications, we focus on the matrix reasoning task, which is the most widely studied RPM-like task in both human intelligence and AI. In the rest of this subsection, we first review the AIG works of matrix reasoning for human testing. Then, we switch to the ones for AI testing.

3.3.2.1 Algorithmically Generating Matrix Reasoning Items for Human Intelligence Testing

Human intelligence tests consist of items which are carefully handcrafted by strictly following the procedures of psychometrics and theories of human intelligence. In particular, Handcrafted items must go through iterations of evaluation and calibrating for good psychometric properties before being included in the final item bank. The attrition rate could be up to 50% (Embretson, 2004). A variety of efforts in AIG have been made to free item writers from this onerousness. In the following, we discuss the typical AIG works of matrix reasoning for human intelligence testing. The title of each reviewed work is followed by a keyword of its most outstanding characteristic. The technical details of the works are summarized in Table 3.1.

3.3.2.1.1 Rule-Based Item Construction—Human-Based AIG

The term "algorithmic item generation" is more often "automatic item generation" in literature. The word "automatic" alludes to the usage of computer. But the algorithms and the theories of what to measure that support the algorithms are the very essence of AIG, rather than the computer. As it will be shown in this first reviewed work, computer is not necessary. Hornke and Habon (1986) conducted one of the earliest studies, if not the earliest, on AIG of matrix reasoning items. They created a procedure for item generation, hired university students to manually execute the procedure, and created 648 3×3 items. Each step in this procedure has finite clearly defined options so that the student can choose between them randomly. Although the diversity and complexity of these items are not comparable to ones handcrafted by human experts, no one had ever "automatically" created so many items before Hornke and Habon (1986).

Hornke and Habon considered the item writing task as the reverse of solving, which can be decomposed into three types of cognitive operations, which address three independent dimensions of the solving process. To generate items, Hornke and Habon thus designed a procedure that sequentially make choices on the three dimensions by selecting from finite sets of options:

- Variation rules of geometric elements: eight options are provided (see the first 8 matrices in Figure 3.8

Table 3.1: The technical details of generator programs in the reviewed works.

Generator	Format	Analogical Direction[a]	Geometric Element						Rules		Per. Org.[d]	Answer Set[e]
			Type[b]	Size	Color	Angle	Number	Position	Type[c]	Number		
Rule-Based	3×3	R, C, O	-	-	-	-	1-2	-	I, S, P, D	1-2	S, I, E	-
Cognitive	3×3	R, C, O	-	-	-	-	-	-	I, S, P, D	1-2	O, F, D	8
GeomGen	3×3	R, C	S, C,T, D2, H, R	-	-	-	-	3×3 grid	I, S, P, N	-	C, N	5+1
Sandia	3×3	R, C, M, S, O	O, R, T, D1, T1, T2	5 sizes	5 colors	5 angles	1-6	center	S, P	1-3	center, overlay	8
CSP	3×3	R, C	-	-	-	-	-	-	S, P, D	1-7	-	8
IMak	2×2	O	C, L, D0, T1	1 size	1 color	8 angles	3	-	P	1-4	E	8+2
PGM-shape	3×3	R, C	C, T, S, P, H	10 sizes	10 colors	-	0-9	3×3 grid	S, P, D	1-4	3×3 grid	8
PGM-line	3×3	R, C	L, D1, C	1 size	10 colors	1 angle	1-5	center	S, P, D	1-4	center, overlay	8
RAVEN	3×3	R	T, S, P, H, C	6 sizes	10 colors	8 angles	1-9	fixed in configurations	I, S, P, A, D	4	7 configurations	8

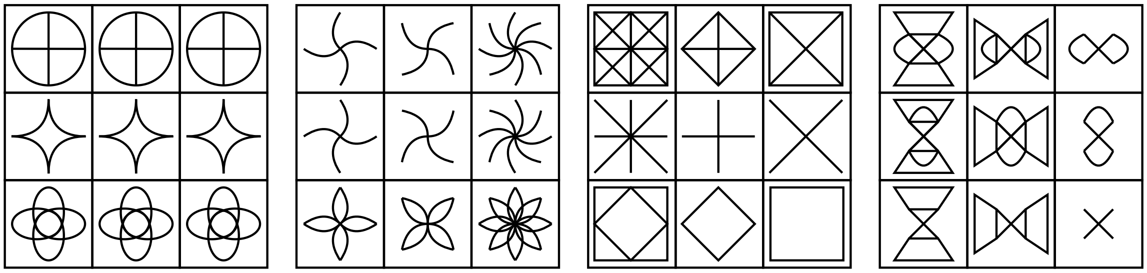
[a] R=row, C=column, M=main-diagonal, S=secondary-diagonal, O=R+C (see (Matzen et al., 2010)).

[b] S=square, C=circle, R=rectangles, T=triangles, H=hexagons, O=oval, L=line, D0=dot D1=diamond, T1=trapezoid, T2="T", D2=deltoid, P=pentagon..

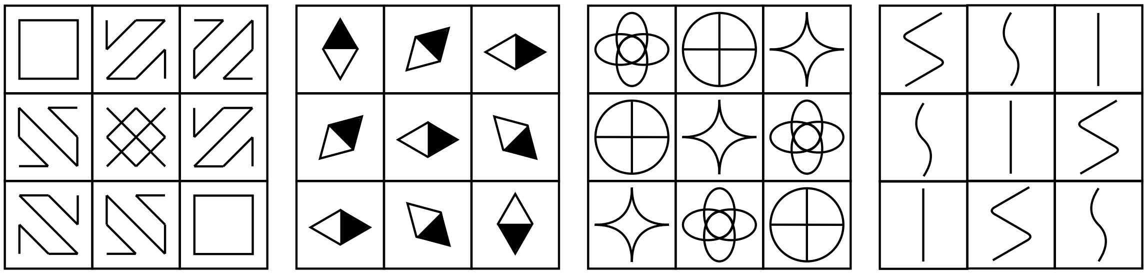
[c] I=identify, S=set/logical operations, P=progression, D=Distribution of 2 or 3 values, N=neighborhood, A = arithmetic (see (Carpenter et al., 1990; Arendasy, 2002)).

[d] S=separation, I=integration, E=embedding, C=classical, N=normal, O=overlay, F=fusion, D=distortion (see (Hornke and Habon, 1986; Embretson, 1998; Primi, 2001)).

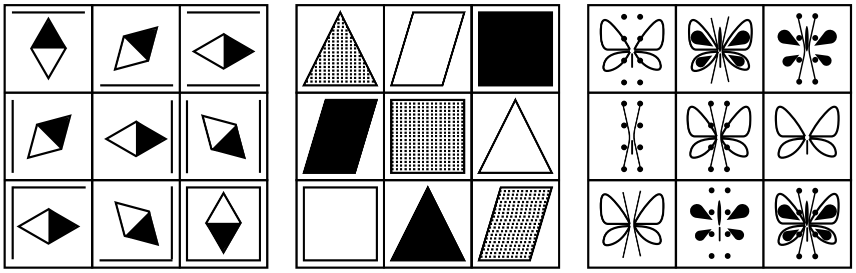
[e] +1="none of the above", +2="none of the above" + "I don't know".



Identity Addition Subtraction Intersection



XOR Orientation Progression Closed Shape Variation Open Shape Variation



Separation Integration Embedding

Figure 3.8: Example items created by following Hornke and Habon's AIG procedure.

for examples)—identity, addition, subtraction, intersection, exclusive union (or symmetric difference), progression, variation of open/closed gestalts (i.e. permutation of three hollow/solid shapes).

- Analogical directions: a variation rule proceeds in row or column direction.
- Perceptual organizations: this dimension addresses how multiple variation rules are combined into a stimulus in a matrix entry. Three options are provided (see the last 3 matrices in Figure 3.8 for examples): separation, integration, and embedding. Separation means that separate geometric elements are used for different variation rule; integration means that different attributes of a single geometric element are used for different variation rules; and embedding means that different parts of a single geometric element are used for different variation rules.

In their experiment, the hired students were given a set of geometric shapes (e.g. differently sized squares and triangles) and instructed to create items by jointly sampling the 3 dimensions and geometric shapes from the given set. The students were told to create each item by combining at most two variation rules. Therefore, the resulting item bank contained only 1-rule and 2-rule items. Human experiments on this item bank showed that the cognitive operations corresponding to these 3 dimensions explained approximately 40% of the item difficulty. As for the unexplained 60%, other early studies (Mulholland et al., 1980) indicated that the numbers of elements and rules were also major sources of difficulty. Although this "human-based" AIG work looks a bit primitive compared to the computational power today, the way it decomposes the generating process has a long-lasting influence on the following works.

3.3.2.1.2 Cognitive Design System Approach—Combination of Cognitive Modeling and Psychometrics

Embretson (1995, 1998, 2004) introduced the Cognitive Design System Approach. Different from other AIG works that focus on generating items, this approach focuses on human testing by integrating cognitive modeling and psychometric models and theories (such as IRT theory and models) into a procedure that is similar to how human experts create and validate intelligence tests. A matrix reasoning item bank was generated as a demonstration.

This approach starts with cognitive modeling of the solving process of an existing cognitive ability test at the information-processing level. In the demonstration, Embretson (2004) reused the cognitive model proposed by Carpenter et al. (1990), which have also been used in many other AIG works of matrix reasoning. However, Embretson (2004) also pointed out that the cognitive model did not include perceptual encoding or decision processes in the solving process. Thus, Embretson (2004) incorporated three extra binary perceptual stimulus features—object overlay, object fusion, and object distortion—in the generation procedure, which represent three different types of mental decomposition of the complete gestalt into its basic parts. Object overlay and fusion are similar to separation and embedding in Figure 3.8, while object distortion refers to perceptually altering the shape of corresponding elements (e.g. bending, twisting, stretching, etc.). A software—ITEMGEN—was developed based on this approach.

Once the cognitive models are determined, the stimulus features are accordingly determined. It then integrates these features into psychometric models to estimate item properties (e.g. item difficulty and item discrimination), formulated as parameterized functions of the stimulus features. The function parameters are initially set by fitting the psychometric models to human data on the existing cognitive ability test. Thereafter, the item properties of newly generated items (by manipulating the stimulus features) can be predicted by these functions. The prediction and empirical analysis of the newly generated items are compared to further adjust the parameters. Once the functions are sufficiently predictive, the psychometric model can be integrated into an adaptive testing system to replace a fixed item bank and generate items of expected properties in real-time. To sum up, the Cognitive Design System Approach is more than constructing an item generator; it also takes into account the psychometric properties of the generated items.

3.3.2.1.3 MatrixDeveloper—4-by-4 Matrices

MatrixDeveloper (Hofer, 2004) is an unpublished software for generating matrix reasoning items. It has been used in a series of psychometric studies of algorithmically-generated matrix reasoning items (Freund et al., 2008; Freund and Holling, 2011a,b,c). According to the limited description in these studies, the MatrixDeveloper is similar to the Cognitive Design System Approach in terms of variation rules (e.g. the five rules of the cognitive model of (Carpenter et al., 1990)) and perceptual organizations (i.e. overlap, fusion, and distortion). The difference is that it generates 4×4 matrix items, which are uncommon for matrix reasoning task. Theoretically, it can thus accommodate more variation rules than 3×3 or 2×2 matrices so that the differential effects of variation rules can be better studied.

3.3.2.1.4 GeomGen—Perceptual Organization

The early cognitive modelings of solving handcrafted matrix reasoning items tend to characterize the items by the numbers of elements and rules and types of rules, for example, (Mulholland et al., 1980; Bethell-Fox et al., 1984; Carpenter et al., 1990). This characterization is consistent with the firsthand experience of working on the items and direct measures of human behavior (such as accuracy, response time, verbal protocols, and eye-tracking). In addition, the rationale of this characterization could be explained through the working memory theory of Baddeley and Hitch. However, for creating new items, we need to consider at least one more factor—perceptual organization (Primi, 2001). It tells how geometric elements and rules are perceptually integrated to render the item image. For example, the third dimension in the procedure of Hornke and Habon (1986) is a specific way to deal with perceptual organization. More generally, perceptual organization involves the Gestalt grouping/mapping of elements using Gestalt principles such as proximity, similarity, and continuity. This factor is less clearly defined and no systematic description of this factor has ever been proposed. But, to create new items, one has to adopt some formalized ways to manipulate perceptual organization.

(Arendasy, 2002; Arendasy and Sommer, 2005) proposed a generator program—GeomGen—that adopted a binary perceptual organization, which was reused and extended in many following works.The perceptual organization in GeomGen provides two options—classical view and normal view. In classical view, the appearance of geometric elements changes while numbers and positions of them remain constant across matrix entries. In normal view, numbers and positions of elements change while the appearance of them remain constant across the matrix entries. An obvious difference between the two views is how the correspondence between elements from two matrix entries is established. And this difference is important because it leads to items that requires different cognitive processes at the very first step of correspondence finding before the rules between matrix entries are considered.

The taxonomy of perceptual organization in GeomGen is only a specific way to define perceptual organization but by no means the unique way. For example, (Primi, 2001) proposed another important taxonomy—harmonic and nonharmonic, which, together with GeomGen taxonomy, forms a more comprehensive description of perceptual organization that is adopted in many following AIG works.

Primi (2001) describes "harmonic organizations as visually harmonic items display perceptual and conceptual combinations that represent congruent relationships between elements, whereas nonharmonic organizations tend to portray competitive or conflicting combinations between visual and conceptual aspects that must be dealt with in reaching a solution." Primi (2001) mentioned that, in the practice of AIG, the nonharmonic items could be derived from the harmonic ones by manipulating the geometric elements to cause misleading Gestalt groupings, as shown in Figure 3.9. The correct Gestalt grouping/mapping (i.e. element correspondences) are obvious in harmonic items, whereas nonharmonic items requires extra cognitive effort to resolve the conflict between competing gestalt groupings and mappings.

In summary, the contributions of all the aforementioned factors—the number of elements, the number of rules, the type of rules, and perceptual organization—to item complexity could be explained by their effect on the central executive component of working memory. But the ways they exert their influences are different. The number of elements and rules relate to the short-term memory management and goal (or strategy) management, whereas the type of rules and perceptual organization relate to selective encoding and short-term memory management Primi (2001). According to the literature of AIG of matrix reasoning,

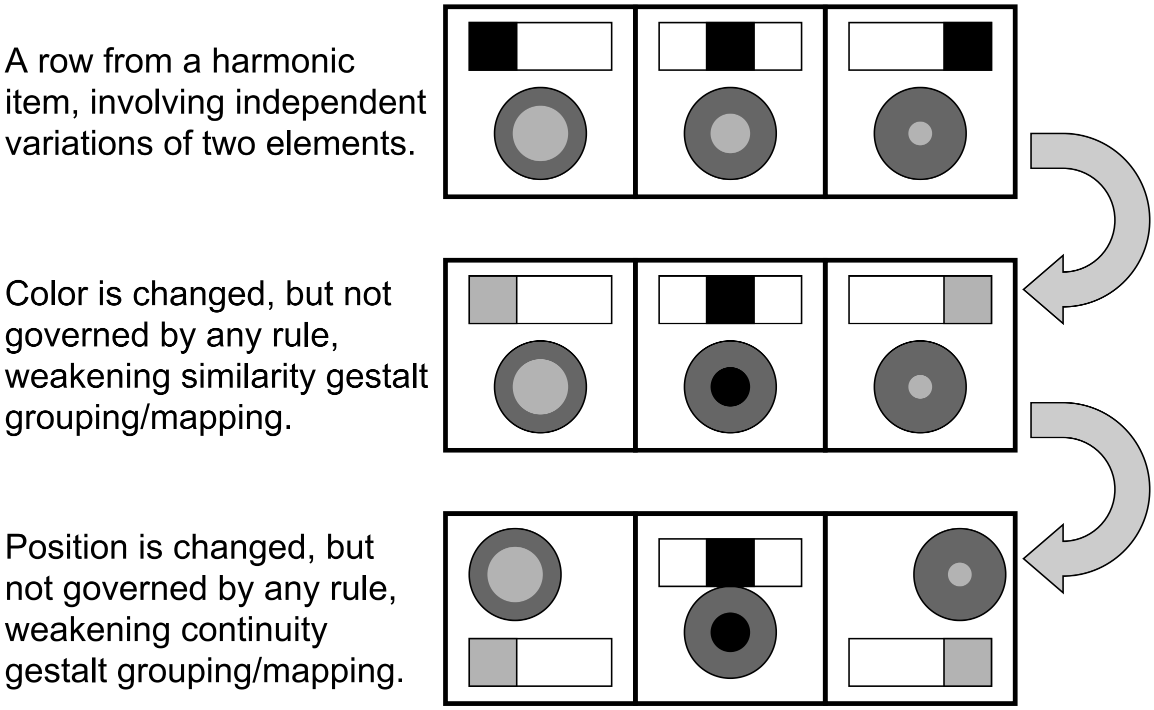

A row from a harmonic item, involving independent variations of two elements.
Color is changed, but not governed by any rule, weakening similarity gestalt grouping/mapping.
Position is changed, but not governed by any rule, weakening continuity gestalt grouping/mapping.

Figure 3.9: An example of deriving nonharmonic items from harmonic items.

the type of rules and perceptual organization are less investigated and might be important for understanding the solving process of matrix reasoning and the item difficulty. Several human studies came to the same conclusion Primi (2001); Arendasy and Sommer (2005); Meo et al. (2007), while other researchers might have different opinions on this (Embretson, 1998; Carpenter et al., 1990).

3.3.2.1.5 Sandia Matrix Generation Software—High-Fidelity SPM Generator

The previous works study AIG more from the perspective of cognitive science and psychometrics. Less details about algorithms and software development were given in the works. But, in practice, we are also interested in how these ideas are implemented and, especially, accessibility of the generator software. Matzen et al. (2010) provided in their work a representative example of this that could "recreate" the 3×3 SPM with high fidelity—Sandia Matrix Generation Software.

Matzen et al. (2010) identified two basic types of 3×3 items in SPM— the element transformation and the logic problems. An element transformation refers to a progressive variation of a certain attribute of the element. There could be multiple variations in different directions, for example, a color variation in the row direction and a size variation in the column direction. However, in every single direction, there is only one attribute varying. This is because, on one hand, it is so in the original SPM, on the other, multiple attributes varying in the same direction does not increase complexity of the problem (to human participants) compared to only one attribute. The attributes considered for transformation problems are shape, shading, orientation, size, and number, each of which takes values from an ordered categorical domain. The logic problems involve operations such as addition/subtraction, conjunction (AND), disjunction (OR), or exclusive disjunction (XOR) of elements. Each generated item is either a transformation one or a logic one, but not both.

In addition, Sandia Matrix Generator generates answer choices in a way of the original SPM problems. An incorrect answer choice could be (a) an entry in the matrix, (b) a random transformation of an entry in the matrix, (c) a random transformation of the correct answer, (d) a random transformation of an incorrect answer, (e) a combination of features sampled from the matrix, or (e) a combination of novel features that did not appear in the matrix.

The item difficulty was studied through an item bank of 840 generated items. The problem set contained problems of 1, 2 or 3 rules (in row, column or diagonal direction). Note that the original SPM problem does not contain 3-rule problems. The generated problem set and the original SPM were given to the same group of college students. Experimental data showed that the generated items and the original SPM had very similar item difficulty. In particular, the data further showed that the item difficulty was strongly affected by the number of rules, analogical directions, and problem types (i.e., transformation problems versus logic problems).

3.3.2.1.6 CSP Generator—First-Order Logic Representation

A more important thing about AIG is to give a general formal description of the generating process, rather than developing various specific generator software. Wang and Su (2015) made such an effort to formalize the generating process of matrix reasoning items through the first-order logic, and turned AIG into a constraint satisfaction problem (CSP) by formulating the "validity"[5] of RPM items into a set of first-order logic propositions.

In particular, a variation rule is represented as an instantiation of Equation (3.1) and (3.2),

$$\exists \alpha \forall i \in \{1,2,3\} \exists o_{i1}, o_{i2}, o_{i3} P(\alpha, o_{i1}, o_{i2}, o_{i3}) \tag{3.1}$$

$$\begin{aligned} P(\alpha, o_{i1}, o_{i2}, o_{i3}) = & Unary(\tau(o_{i1},\alpha), \tau(o_{i2},\alpha), \tau(o_{i3},\alpha)) \wedge \\ & Binary(\tau(o_{i1},\alpha), \tau(o_{i2},\alpha), \tau(o_{i3},\alpha)) \wedge \\ & Ternary(\tau(o_{i1},\alpha), \tau(o_{i2},\alpha), \tau(o_{i3},\alpha)) \end{aligned} \tag{3.2}$$

where α is a goemetric attribute, o_{ij} is a geometric elements in the figure of Row i and Column j, $\tau(\alpha, o_{ij})$

---

[5] Not exactly the same definition of validity in psychometrics

is the value of α of o_{ij}, and P is a predicate that describes the variation pattern of attribute α in each row. In Equation (3.2), the predicate P further equals a conjunction of three predicates—*Unary*, *Binary*, and *Ternary*—representing three categories of relations commonly used in matrix reasoning, as illustrated in Figure 3.10.

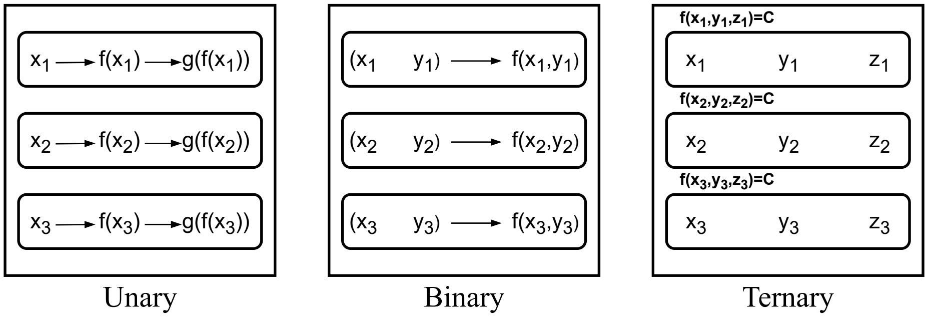


Figure 3.10: Three categories of relations commonly used in matrix reasoning (Wang and Su, 2015).

An interesting observation of Figure 3.10 is that, mathematically, the unary relation is a special case of the binary relation, which is a special of the ternary relation. That is, the ternary relation is theoretically sufficient to generate all the items. However, interpreting the same variation as unary, binary and ternary relations requires different working memory abilities and thus leads to different difficulties. Therefore, these three categories are cognitively different, and need to be separately included in a generator program to achieve a better control over psychometric properties.

Equation (3.1) and (3.2) represent only the variation pattern of a single attribute α. There could be multiple variation patterns of different attributes in a matrix, i.e., multiple different instantiations of Equation (3.1) and (3.2). Meanwhile, it is also possible that some attributes are not assigned any instantiations of Equation (3.1) and (3.2). In this case, they could be given either constant values or random values across matrix entries. Random values may cause distracting effects in the generated items, which is similar to the nonharmonic perceptual organizations in (Primi, 2001).

To generate an item through Equation (3.1) and (3.2), the generator program samples values from finite domains to determine (a) the number of rules (i.e., the number of the instantiations of Equation (3.1) and (3.2)), (b) the attribute α for each rule, (c) the values of $\tau(\alpha, o_{ij})$, (d) the specific types of *Unary*, *Binary*, and *Ternary* relations. The matrix image is rendered from the instantiations of Equation (3.1) and (3.2), and each incorrect answer choice is generated by breaking an instantiation of Equation (3.1) and (3.2) (i.e., using values not satisfying them).

The generated items and the APM test were also given to a small group of university students. The experimental data showed that the overall difficulty and rule-wise difficulty (number of rules) were similar to the items in APM. However, as the author pointed out, their generator could not synthesize all the items in APM for some underlying transformations were hard to implement. When the items were created with distracting attributes, the generated items became much more difficult for human subjects.

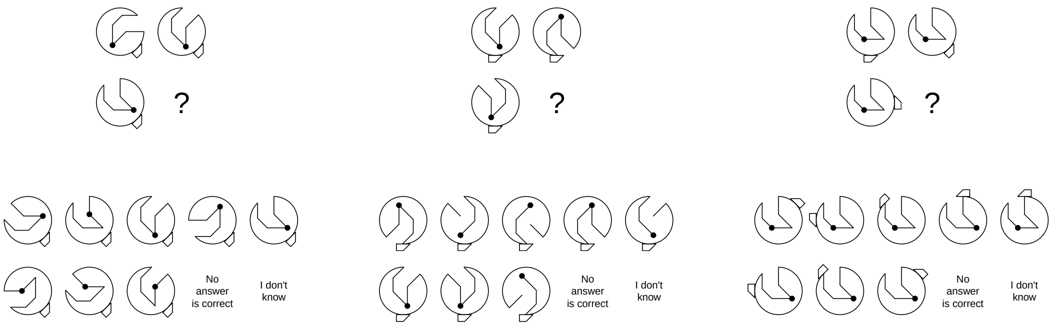
3.3.2.1.7 IMak Package—Open Source

Although there have already been many works on AIG of matrix reasoning, the generator software and the source code are usually not easily available to the public. This makes it hard to reproduce and build upon these works. Blum and Holling (2018) realized this point and released their generator as an R package—IMak package—that is globally available via the Comprehensive R Archive Network. The source code and detailed documentation of their work come with the R package. New items could be obtained by simply three lines of R code in the R interpreter—one for downloading the package,one for importing the package,and one for generating the items.

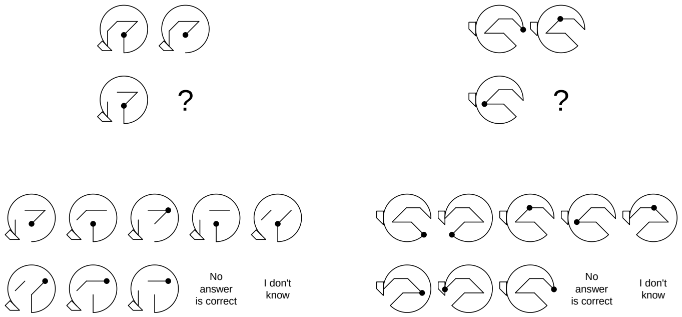
The author's purpose of developing the IMak package is to study the effect of types of variation rules on item difficulty. The generator was thus designed to manipulate the types of rules while keeping other factors constant, and, thus, the generated items look quite different from the generated items of the generators mentioned above. For example, Figure 3.11 shows some example items that we created through this package, each of which exemplifies a basic rule type. With the current release (version 2.0.1), the geometric elements are limited to the main shape (the broken circle plus the polyline in it), the trapezium that is tangent to the main shape, and the dot at one of the corners of the polyline. Furthermore, the size and shape of these element are fixed for all generated items, but the position, orientation and existence would vary according to 5 basic rules.

As shown in Figure 3.11, there are 5 basic rules in IMak. All the rules are in the outward analogical direction (i.e. row and column). For example, in Figure 3.11a, the main shape is rotated counterclockwise by 45 degrees in the first row; the main shape is rotated counterclockwise by 90 degrees in the first column. Then the correct answer would be a counterclockwise rotation of the main shape by 135 (45 + 90) degrees compared to the top left one. Similarly, all the other 4 examples follow the same analogical direction. Each item could contain up to 4 rules (because mains shape rotation and reflection are conflicting). This design seems to excessively simplify the RPM-like problems, but it does serve the very purpose of study the differential effect of rules by fixing other factors.

Besides open-source accessibility and the special design of geometric elements, IMak has four other distinctive features that are inspiring for following works. Firstly, IMak generates 2×2 format of AIG of matrix reasoning. Being affected by the famous work of (Carpenter et al., 1990) on RPM, the vast majority of AIG works would only generate 3×3 matrices. The 2×2 items have largely been neglected in the AIG



(a) Main shape rotation (b) Main shape reflection (c) Trapezium rotation



(d) Line segment subtraction (e) Dot movement

Figure 3.11: Example items generated through the IMak package. Each item exemplifies a single basic rule. The correct answer is set to the first answer choice for demonstration.

works of matrix reasoning. Secondly, the answer set contains two more meta-choices “no correct answer” and “I don’t know”, which encourage subjects to solve the items more constructively rather than eliminating responses. Thirdly, the variation of one element could depend on the variation of another element. For example, the dot’s moves depend on the variation of the main shape, since the dot only moves along the polyline in the main shape. This kind of variation rule is rare in matrix reasoning items, but common in real-world problem-solving, and it represents an extra complexity factor of matrix reasoning.

Last but not least, IMak used a rule-dependent strategy to generate incorrect answer choices. For 1-rule items, 4 distinct values of the attribute of the rule are sampled, including the correct value; since all other attributes remain constant in the matrix, another random attribute is chosen and sampled for 2 values. The resulting 8 (4×2) combinations make the 8 options in the answer set. For 2-rule items, 3 values are sampled for each of the 2 attributes of the 2 rules, resulting in 9 combinations, and one of them is discarded. For 3-rule items, 2×2×2 combinations are sampled in the same way. For 4-rule items, 2×2×2×2 combinations were sampled in the same way, and half of them are discarded.

In a human experiment, 23 generated items were administered to 307 participants from Germany, Indonesia, and Argentina. Reliability, validity and unidimensionality were initially verified by the experiment results. Particularly, item difficulty could be partly predicted from the number and type of rules based on psychometric models. As a summary, the open source software is a more recommended way to publish AIG works, especially for research purpose, as it can be shared across research groups around the world. More importantly, the studies should not be restricted to a fixed set of items but the way the generator is designed.

3.3.2.2 Algorithmically Generating Matrix Reasoning Items for AI Testing

We now review two AIG works of matrix reasoning that were specially for AI testing. The datasets generated in these two works are extremely influential on the data-driven AI models for solving RPM-like tasks because almost all of them were tested on one or both of these two datasets. In addition, we also review the works that address the context-blind issue of the algorithmically generated datasets reviewed, which is a special and important issue for data-driven AI models.

3.3.2.2.1 Procedurally Generated Matrices

Based on the five rules in (Carpenter et al., 1990), Barrett et al. (2018) continued the first-order logic approach of Wang and Su (2015) and created a large-scale (1.2M items) dataset of matrix reasoning items—Procedurally Generated Matrices (PGM). Since the generator program and source code are not publicly available, our discussion is based on the description in (Barrett et al., 2018) and our observation of the dataset.

In PGM, an instantiation of Equation (3.1) and (3.2) in the first-order logic approach was denoted by a triplet $[r,o,a]$ of relation r, object o [6] and attribute a. These three factors are not independent. Particularly, Figure 3.12 summarizes their dependencies in the generator of PGM. Figure 3.12 consists of 29 paths from the left to the right, corresponding to 29 $[r,o,a]$ triplets[7].

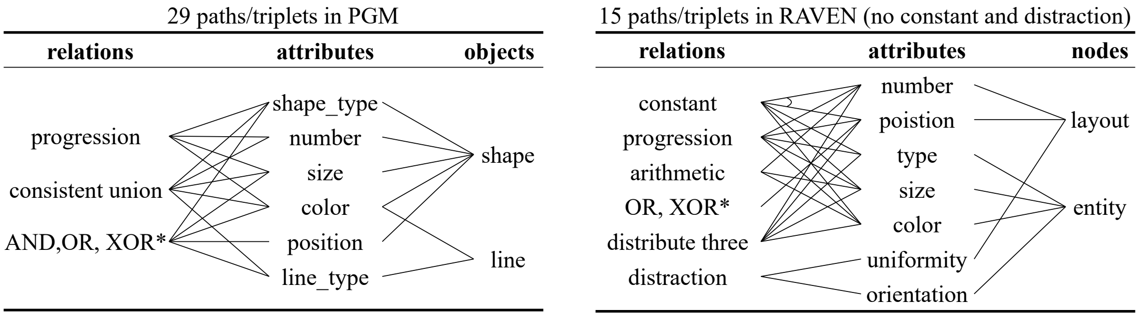
As shown in Figure 3.12, the objects in PGM are classified into two disjoint subsets—shape and line. In the shape subset, closed shapes are arranged in 3×3 grid (fixed positions in this case) inside each matrix entry (do not mistake this with 3×3 matrices). In the line subset, line drawings spans the whole area of a matrix entry and are always centered in the matrix entry. A PGM item can include both shapes and line drawing, with the shapes superimposed on the line drawings, but the reasoning about these two are completely independent. Thus, in Table 3.1, we split PGM into two rows to describe it more clearly.

The generation procedure of a PGM item could be described by 5 steps: (a) sample 1 to 4 triplets from the 29 triplets described in Figure 3.12 (number triplets and position triplets can not be selected simultaneously); (b) determine the analogical direction for each triplet: row or column; (c) sample attribute values for each triplet from their domains (Different sampling methods are specifically implemented for different rules and

---

[6] Geometric elements in the AIG works for human intelligence tests are commonly referred to as goemetric "objects", or objects for short, in AI works. We thus use the term "object" in the discussion of AIG works for data-driven AI models.

[7] This number—29—equals the number of triplets mentioned in the work of Barrett et al. (2018), which, however, did not provide a list of the 29 triplets. Therefore, we could only conjecture that the 29 triplets here are the ones used in PGM.


29 paths/triplets in PGM

15 paths/triplets in RAVEN (no constant and distraction)

* AND, OR, XOR is three different relations, contributing three paths.

* OR, XOR denotes two different relations, contributing two paths.

Figure 3.12: Left: The dependencies among relations, objects, and attributes used to generate the Procedurally Generated Matrices (PGM) dataset (Barrett et al., 2018). Each path from left to right corresponds to a $[r,o,a]$ triplet representing a variation pattern in the matrices. As one can check, there are 29 paths, i.e. $[r,o,a]$ triplets, in the graph. Note that Barrett et al. (2018) did not differentiate between "shape_type" and "line_type" and referred to both of them as "type". But these two are treated as two distinct attributes in PGM's implementation. Right: The dependencies among relations, nodes, and attributes used to generate the RAVEN dataset. Note that we listed "distraction" as a rule in this graph to indicate that uniformity and orientation are distracting attributes. The paths from constant through number and position to layout are treated as a single rule in RAVEN's implementation. Therefore, there are 15 paths/rules in the graph.

attributes); (d) determine the attribute values for unspecified attributes (either constant or random); and (e) render all attribute values into a pixel image of the matrix.

The relations used in the PGM dataset, which are also referred to as rules in other literature, stem from the 5 rules of APM summarized in (Carpenter et al., 1990), as follows:

- Constant in a row.
- Quantitative pairwise progression.
- Figure addition or subtraction, i.e. the set union and set diff (not arithmetic addition and subtraction), which could also be considered as the logical operator OR and XOR.
- Distribution-of-three-values, i.e. the consistent union.
- Distribution-of-two-values, i.e. the logical operator XOR.

Comparing the PGM relations to the above rules, we found that they are almost equivalent. The "constant in a row" corresponds to the without-distraction mode in PGM. The "distribution-of-three-values" corresponds to the consistent union in PGM. The "figure addition or subtraction" and "Distribution-of-two-values" are logical operator OR and XOR in PGM. However, the PGM has one more relation—AND—in addition to the 5 rules in (Carpenter et al., 1990) to be more complete.

3.3.2.2.2 Relational and Analogical Visual rEasoNing

The spatial configuration, as an important dimension of perception organization, is highly restricted in PGM—3×3 grid for the shape subset, all-centered for the line subset, and superimposing a shape item on a line item. To enrich the spatial configuration of AIG of matrix reasoning, Zhang et al. (2019a) developed a new generator and generated the Relational and Analogical Visual rEasoNing (RAVEN) dataset. In particular, RAVEN includes 7 hardcoded spatial configurations, as shown in Figure 3.13. The source code of RAVEN's generator is available online[8]. The discussion of RAVEN is thus based on the inspection of the RAVEN's generator's source code.

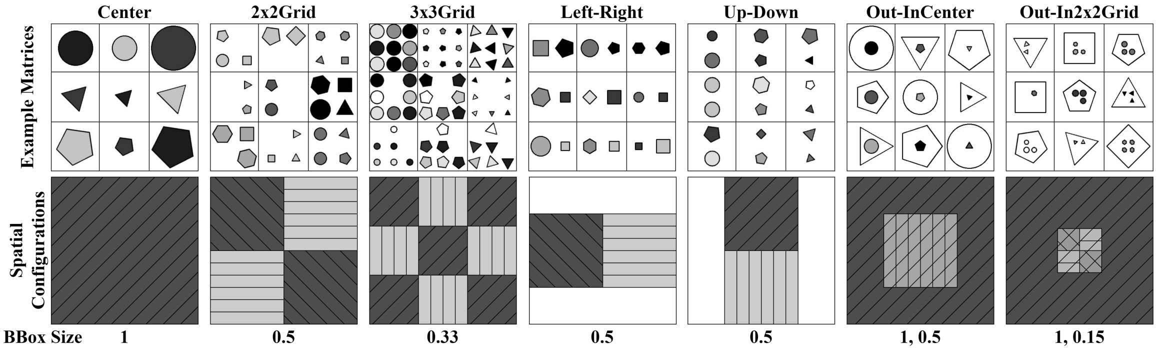


Figure 3.13: 7 hardcoded spatial configurations—center, 2x2Grid, 3x3Grid, Left-Right, Up-Down, Out-InCenter, and Out-In2x2Grid—are used to arrange objects in each matrix entry in the RAVEN dataset. Each configuration is represented by the bounding boxes that objects could occupy. The position and size of each bounding box are hardcoded in the generator program. An example matrix for each configuration is given in the first row (image obtained by running the generator code). Note that not every bounding box has to be occupied, but every object has to be in one of the bounding boxes.

The 7 configurations are derived from a more general symbolic representation framework for images—Attributed Stochastic Image Grammar (A-SIG). In A-SIG, an image is described by a tree structure, where the conceptual granularity becomes finer and finer from root toward leaves. To generate RAVEN, the tree structure is predefined as a general A-SIG tree as shown in Figure 3.14, which consists of 5 conceptual levels—scene, structure, component, layout, and entity—and uses a stochastic tree-traversal process to generate images. In general, the main idea of an A-SIG tree is that, while traversing the tree, if the current node has dashed edge to its child nodes, then expand a single random child node; if the current node has solid edge to its child nodes, then expand all its child nodes. Attributes and their attribute value domains are attached to nodes so that images can later be generated by sampling from these domains after the tree structure is determined. Such a stochastic traversing process from the root to leaves would generate a skeleton of a class of images—i.e. a spatial configuration. However, the 7 configurations in RAVEN were hardcoded in the language of A-SIG, rather than generated through this stochastic traversing process, which could otherwise have made RAVEN more diverse in spatial configuration.

To compare with the PGM dataset, we represent PGM items also in A-SIG, as shown in Figure 3.15. The line configuration of PGM is basically the same as the center configuration of RAVEN except that the entity

---

[8] https://github.com/WellyZhang/RAVEN

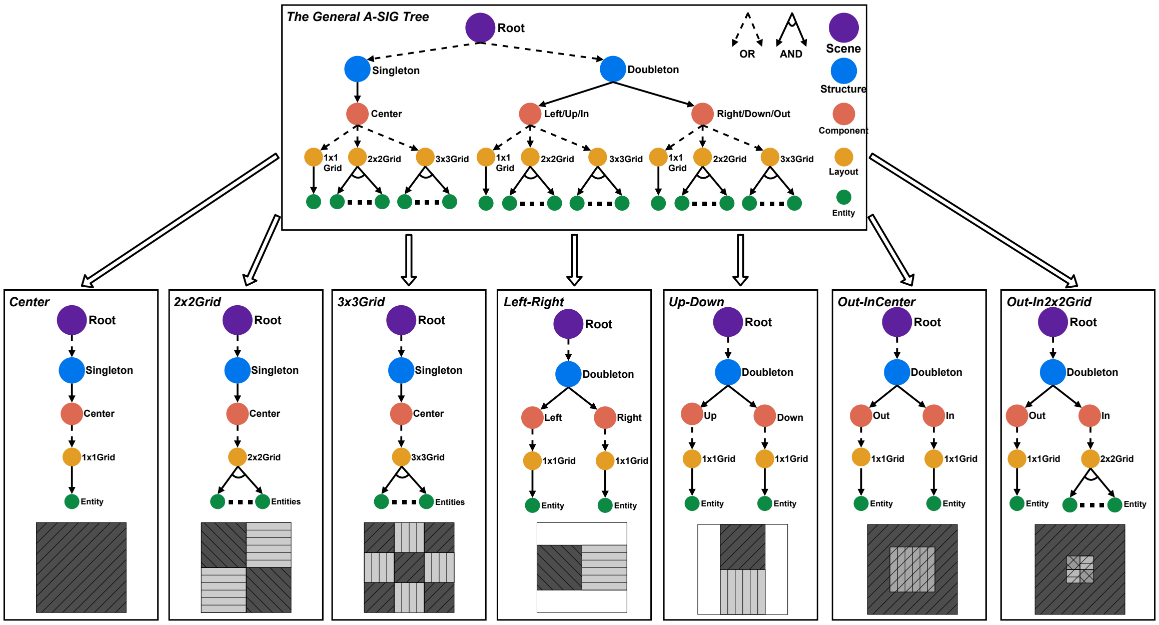


Figure 3.14: The general A-SIG tree and 7 specific A-SIG trees used in the RAVEN dataset (image adapted from (Zhang et al., 2019a) by adding more technical details from the source code of the generator). The root node denotes the scene that the image describes. The structure nodes are the containers of different spatial structures. A structure is composed of components that could be overlaid with each other. Each component has its own layout and, more importantly, variation rules, which are independent of other components. The layout node, as its name indicated, contains the attributes specifying the number and positions of geometric objects. Entities represent geometric objects with attributes, not including number and position.

types (shape) are different. The shape configuration of PGM is almost the same as the 3x3Grid configuration of RAVEN except that bounding box sizes are slightly different. The shape-over-line configuration of PGM is also conceptually similar to the double-component configurations of RAVEN. The general difference between PGM and RAVEN lies in the layout and entity nodes. As shown in Figure 3.15, the PGM dataset is not able to separate the concepts of “entity” and “entity layout” by using triplets $[r,o,a]$. That is, the object o takes the roles of both layout and entity nodes, but could not play the roles effectively and simultaneously.

RAVEN inherited all the five rules from (Carpenter et al., 1990). Moreover, the “addition-and-subtraction” rule is extended in RAVEN containing not only figure addition and subtraction (i.e., the set operations “OR and XOR”) but also arithmetic addition and subtraction, which were not discussed in (Carpenter et al., 1990). Since these two operations are conceptually different, we refer to the arithmetic addition and subtraction as “arithmetic”, and the figure addition and subtraction as “OR and XOR”. In addition, the “distribution-of-three-values and distribution-of-two-values” from (Carpenter et al., 1990) are merged into a single rule in RAVEN by considering the latter as a special case of the former with a null value for one of the three values. Therefore, RAVEN has a slightly different rule set compared to PGM. Similarly, we could represent the variation rules of RAVEN also as triplets — $[r,n,a]$ where n represents nodes (layout or entity) in A-SIG trees,

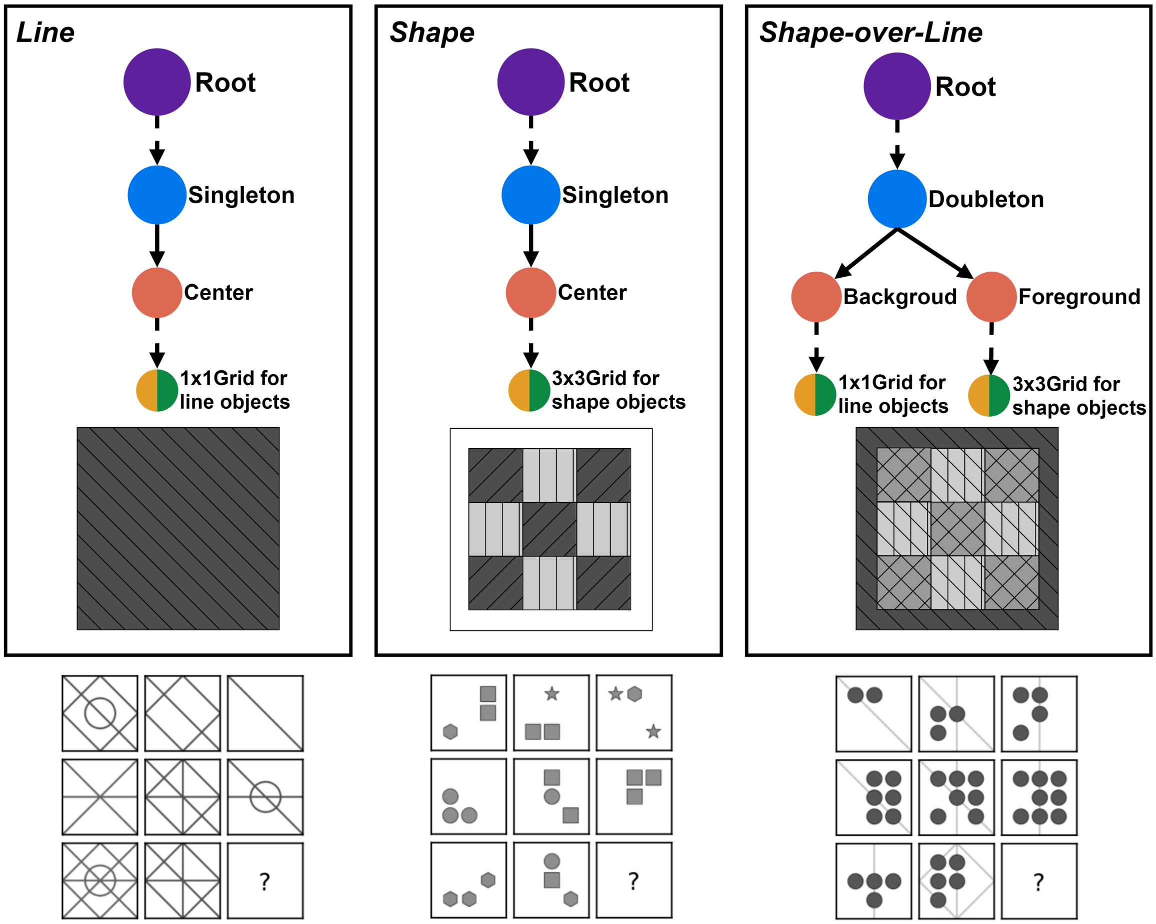


Figure 3.15: The spatial configurations of PGM represented in A-SIG to compare with RAVEN. There are 3 spatial configurations in PGM—line, shape, and shape-over-line—in PGM. The example matrix is given for each configuration at the bottom (images taken from PGM dataset).

and *r* and *a* are relations and attributes, being the same as PGM. Then Figure 3.12 shows the dependencies among *r*, *n* and *a*.

PGM and RAVEN generators are similar in some aspects. In particular, they share two similarities. First, their choices of attributes, attribute domains, and rule types are similar. For example, they both forbid number-rule and position-rule from co-occurring in an item because these two attributes would probably conflict with each other. Second, although RAVEN has more spatial configurations, these configurations are not structurally different from PGM (as can be seen from the comparison of their A-SIG trees). Meanwhile, PGM and RAVEN are different in two aspects. First, they are different in the number of rules in an item. In PGM, 1 to 4 triplets were sampled from the 29 triplets. In contrast, in a RAVEN item, every attribute is governed by a rule except the two distracting attributes (uniformity and orientation). Thus, there are 4 rules (for number/position, type, size, and color, respectively) in each RAVEN item. Second, the rules in RAVEN are all row-wise while the rules in PGM are either row-wise or column-wise.

3.3.2.2.3 Context-Blind Issue

The answer sets in RAVEN were generated in a similar way to the way in the first-order logic approach. That is, each incorrect answer choice is created by modifying a single attribute of the correct answer. RAVEN is slightly different from (Wang and Su, 2015) because RAVEN has only 5 attributes (not including the distracting attributes) whereas (Wang and Su, 2015) has 15 attributes. Hence, in (Wang and Su, 2015), every incorrect answer has a unique attribute on which it differs from the correct one; but RAVEN has to reuse some of the 5 attributes to generate 7 incorrect answers, i.e. an attribute is given different values to generate multiple incorrect answers.

This method of creating incorrect answer choices reaches the maximum level of distracting and confusing effect, because one must identify all the variation rules to solve the problem. On the contrary, ignoring any rule would lead to multiple choices. However, this design has a major drawback—it fails the context-blind test for multi-choice problems. In a matrix reasoning item, the incomplete matrix is the context of the multi-choice problem that provides information for solving the problem. Failing the context-blind test means that it is possible for human participants or computational models to solve the item while turning blind to the context.

Two works (Hu et al., 2021; Benny et al., 2021) separately pointed out the context-blind issue of RAVEN. They provided evidence that data-driven AI models can achieve high accuracies (from 70%+ to 90%+) when only given access to the answer sets of RAVEN. The context-blind performance of some data-driven AI models is even better than the normal performance with full access to the items. This implies that data-driven AI models are capable of capturing the statistical regularities in the answer sets. The reason for this context-blind issue obviously lies in the generating process of answer set. In particular, since each incorrect answer choice is a variant by modifying a single attribute of the correct answer choice, the correct answer must be the one that possesses every common feature among all the choices (or, equivalently, the one most similar to every other choice).

Both Hu et al. (2021) and Benny et al. (2021) proposed their own solutions to this issue—the Impartial-RAVEN and RAVEN-FAIR datasets. These two datasets have the same context matrices as the original RAVEN and regenerated the answer sets in different ways. The similarity and difference between these three versions can be clearly illustrated by putting them in simple graphs. If we represent each answer choice as a vertex and each modification of an attribute as an edge, then the answer sets of the three versions can be depicted by the graphs in Figure 3.16. The answer set of the original RAVEN is created by modifying an attribute of the correct answer. Thus, its graph is a star centered at the correct answer (the solid vertex). And what the aforementioned computational models in the context-blind test captured was the unique center of the star structure.

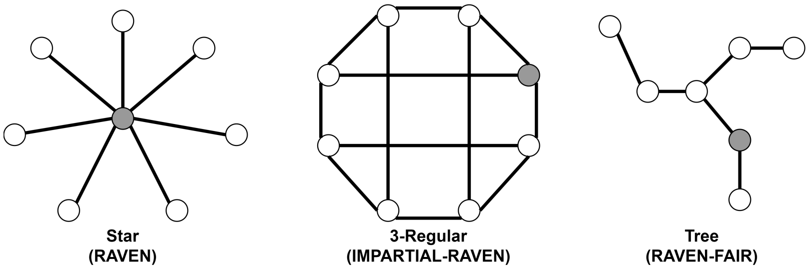


Figure 3.16: The answer sets of three versions of RAVEN datasets depicted in graphs. Each vertex is an answer choice and two adjacent vertices differ by one attribute.

Hu et al. (2021) proposed the Impartial-RAVEN, in which the answer set can be represented by a 3-regular graph in Figure 3.16. To create such a graph, three independent attributes are randomly chosen from the five attributes of RAVEN, and three values of the three attributes are sampled from the three attribute value domains, respectively, so that the newly sampled values are different from the ones of the correct answer. Then, by assigning new values to these attributes combinatorially, we would have 2^3=8 answer choices, including the correct one. The relations among these 8 answer choices form the 3-regular graph in Figure 3.16.

Benny et al. (2021) proposed a less regulated procedure to generate answer sets. Starting from an initial answer set consisting of only the correct answer, an answer choice is randomly selected from the current answer set, and then an attribute of the selected answer choice is randomly altered to alter to create a new answer choice; repeat this process until we have 8 answer choices. This procedure results in tree structures similar to the one in Figure 3.16.

These two enhanced versions of RAVEN were tested by context-blindly training the baseline model in (Zhang et al., 2019a) and the CoPINet model in (Zhang et al., 2019b). The accuracy decreased to below 20%. Ideally, a human subject or computational model who context-blindly works on the RAVEN items should perform as well as a random guess, i.e. $1/8$=12.5%, which implies that the answer set per se does not provide any useful information for solving the item. However, in the practice of item writing, to maintain a certain level of distracting and confusing effect of incorrect answer choices, the majority of incorrect answer choices must share some similarities among themselves, with the correct one, and the context matrix, which would raise the performance of random guess a bit. On the flip side, without this design, it would be quite easy for subjects to find the correct answer, because incorrect answers would be very much perceptually distinct from the context and other answer choices. Therefore, a reasonable context-blind performance would be slightly higher than random guess. The balance is determined by the item writer's judgment.

A subtle difference between the two enhancements of RAVEN could be found by comparing their graphs in 3.16. If we consider a single trial (in a probabilistic sense) where we context-blindly give a participant (or an AI model) an item from Impartial-RAVEN and an item from RAVEN-FAIR, the probability that this participant solves the Impartial-RAVEN item would be almost the same as the probability of solving the RAVEN-FAIR item. However, if we repeat this with different items again and again, the performance on RAVEN-FAIR would probably exceed the performance on Impartial-RAVEN, assuming that the participant is intelligent enough to figure out the graph structures behind the answer sets, and thus makes an educated guess by selecting the “center” (or the max-degree vertex) of trees in a probabilistic sense. In this case, we would say that the RAVEN-FAIR is context-blind valid at the item level, but not at the dataset level.

3.3.2.3 Summary

In this subsection, we reviewed AIG works of matrix reasoning items. We classified the works into two groups by their purposes—whether it is for human intelligence testing or for AI testing. The works in the first group aim at not only generating items but also good psychometric properties. As the classical studies on intelligence tests, these works are usually based on cognitive models and psychometric models. The choices of stimulus features are thus determined by the cognitive and psychometric models. Particularly, the factors—the number of elements, the number of rules, the type of elements, types of rules, analogical directions, and perceptual organizations—are usually considered in this line of research. Among these factors, the types of elements and rules and perceptual organization are the less investigated ones due to the difficulty in defining and formalizing them.

The works in the second group can be seen as the continuation of the first group, but the psychometric aspects are less emphasized. For example, in an human experiment of PGM, in which 18 items were administered to human participants, participants without prior experience failed almost all the items, whereas participants with prior experience scored above 80%. Such a result is definitely not what a psychometrician would expected from a test for eductive ability, fluid intelligence, or general intelligence. In contrast, the result appear to be a result from a test of reproductive ability or crystallized intelligence. Generally speaking, this result implies that the datasets for AI testing do not necessarily qualify for human intelligence testing.

More importantly, this gives rise to another interesting question—how do we assess the performance of data-driven AI models on the large datasets such as PGM and RAVEN? On one hand, some data-driven AI models indeed perform well on AIG items that pose great challenges to human subjects; on the other hand, training on the large-scale datasets specially prepares the AI models for a highly restricted subset of the problem domain, but human subjects, who are not trained at all, or just trained on several examples from this subset, could perform well in the entire problem domain.

Similar questions were asked when AI systems first entered the area of human testing (Detterman, 2011).

Efforts have been made to address these questions. (Bringsjord and Schimanski, 2003; Bringsjord, 2011) address this issue by incorporating AI testing into a general concept—psychometric AI. Hernández-Orallo et al. (2016) proposed that (a), instead of collecting items, we should collect item generators, and (b) the generated items should be administered to machine and human (and even other animals) alike (universal psychometrics). All these propositions are constructive and, meanwhile, suggest much higher requirements for AIG studies.

Current AIG datasets are far below the level of flexibility and diversity that human item writers can achieve. For example, the spatial configurations in PGM and RAVEN are fixed; inter-element variation, in which the variation of one element depends on the variation of another element, is also very rare; so are perceptually and conceptually ambiguous analogies. A more promising methodology for AIG of RPM-like tasks for AI testing is to study the problem domain and human cognition, rather than construct ad hoc generator programs. Huge uncharted territories lie in the complexity factors such as the types of elements and rules and perceptual organization, and how the nature of problem changes as different administration/evaluation protocols are used for human subjects and AI models.

3.4 Computational Models for Solving RPM and RPM-Like Tasks

In previous sections, we have established the basic understanding of the problem domain represented by RPM, which lays the foundation for us to discuss the core topic of this article—computational models for solving the problems. Similar to the way in the previous discussion, we start from the origin of the research, keep the prerequisite knowledge at a minimal level, and unfold our discussion in a manner that reveals the philosophy behind technical development in simplest language.

The ultimate purpose of this section is to help our readers develop a solid understanding rather than enumerating as many previous works as possible in chronological order or in an arbitrary taxonomy. Therefore, we use a narrative, which simulates the process of how an novice's understanding of the solution to the problem domain would naturally evolve if not influenced by the external conditions (such as computational power) and other relevant research works. This narrative is not real history but specially designed to reduce the complexity for understanding. In particular, the computational models that arise late in this narrative might arise early in reality, and vice versa. Examples like this are common in scientific research: the original concepts behind some cutting-edge technologies might have been there for decades before these technologies are implemented, but some alternatives to the original concepts, due to being easy to implement, might have already been implemented before the cutting-edge technologies; when we look back at these concepts, we rearrange the order to make the concepts more coherent and understandable. Thus, this narrative is a conceptual chronicle for understanding rather than a real chronicle for recording.

In this conceptual chronicle, we divide the development of computational models for solving RPM into five stages—imagery-based approach, logical reasoning, neuro-symbolic reasoning, learning approach, and data manipulation. With hindsight, we found that an upward-spiral pattern is looming out of these five stages. That is, researchers are making process while visiting the same places again and again with better and better understanding. The places could be specific research questions or a type of approach to answer the research questions. The conceptual chronicle starts from a straightforward approach (imagery-based approach) which is specific to the problem domain but very effective; it then moves on to more and more general approaches (logical reasoning, neuro-symbolic reasoning, and learning approach); when these approaches is still incapable of solving the problem domain perfectly, it returns to the study of the problem domain per se and solves the problem in a similar way to the first approach, but uses completely different set of techniques. The same upward-spiral trajectory could be described differently (e.g., different methodologies are alternatively dominating the research of intelligence), but the pattern that it revisits the same places again and again until the entire problem domain is perfectly solved remains unchanged.

In the rest of this section, we will use the acronyms of computational models for simplicity and please refer to Table 3.2 for their full names.

Table 3.2: Computational Models for Solving RPM and RPM-like Tasks

Acronym	Full Name	Article
-	Gestalt Algorithm	(Hunt, 1974)
ASTI	Affine and Set Transformation Induction	(Kunda, 2013)
ASTI+	Affine and Set Transformation Induction Plus	(Yang et al., 2020)
-	Fractal Model	(McGreggor et al., 2014)
-	FAIRMAN	(Carpenter et al., 1990)
-	BETTERMAN	(Carpenter et al., 1990)
CogSketch+SME	CogSketch and Structual Mapping Engine	(Lovett et al., 2009)
-	Anthropomorphic Solver	(Strannegård et al., 2013)
-	ANALOGY	(Evans, 1964)
-	Analytic Algorithm	(Hunt, 1974)
ALANS2	ALgebra-Aware Neuro-Semi-Symbolic	(Zhang et al., 2020)
PrAE	Probabilistic Abduction and Execution	(Zhang et al., 2021)
VAE-GPP	Variational Autoencoder and Gaussian Process Priors	(Shi et al., 2021)
TRIVR	Two-Stage Rule-Induction Visual Reasoning	(He et al., 2021a)
NVSA	Neural-Vector-Symbolic Architecture	(Hersche et al., 2022)
Pairwise-ADV*	Pairwise Attribute Difference Vector	(Mekik et al., 2017)
Triple-ADV*	Triple Attribute Difference Vector	(Mekik et al., 2018)
DeepIQ	Deep IQ	(Mańdziuk and Żychowski, 2019)
CNN+MLP	-	(Hoshen and Werman, 2017)
CNN+decoder*	-	(Hoshen and Werman, 2017)
ResNet+MLP	-	(Barrett et al., 2018)
Wild-ResNet+MLP	-	(Barrett et al., 2018)
WReN	Wild Relation Network	(Barrett et al., 2018)
LEN	Logic Embedding Network	(Zheng et al., 2019)
MXGNet	Multiplex Graph Network	(Wang et al., 2020)
multi-layer RN	multi-layer Relation Network	(Jahrens and Martinetz, 2018, 2019, 2020)
SRAN	Stratified Rule-Aware Network	(Hu et al., 2021)
MRNet	Multi-Scale Relation Network	(Benny et al., 2021)
Rel-Base	Basic Relational Reasoning	(Spratley et al., 2020)
Rel-AIR	Attend-Infer-Repeat Relational Reasoning	(Spratley et al., 2020)
CNN+LSTM+MLP	-	(Barrett et al., 2018)
Double-LSTM	-	(Sekh et al., 2020)
ESBN	Emergent Symbol Binding Network	(Sinha et al., 2020)
NTM	Neural Turing Machine	(Sinha et al., 2020)
ARNe	Attention Relation Network	(Hahne et al., 2019)
HTR*	Hierarchical Transformer Reasoning	(An and Cho, 2020)
NI	Neural Interpreter	(Rahaman et al., 2021)
SCL	Scattering Compositional Learner	(Wu et al., 2021)
4 VAE+WReN	4 variants of VAE plus WReN	(Steenbrugge et al., 2018; van Steenkiste et al., 2019)
generative-MRNet*	-	(Pekar et al., 2020)
LoGe	Logic-Guided Generation	(Yu et al., 2021)
MCPT	Multi-label Classification with Pseudo Target	(Zhuo and Kankanhalli, 2020)
PRD	Pairwise Relations Discriminator	(Kiat et al., 2020)

to be continued on the next page

Table 3.2: (continued from previous page)

Acronym	Full Name	Article
LABC	Learning Analogies by Contrasting	(Hill et al., 2019)
CoPINet	Contrastive Perceptual Inference Network	(Zhang et al., 2019b)
DCNet	Dual-Contrast Network	(Zhuo and Kankanhalli, 2021)
ACL	Analogical Contrastive Learning	(Kim et al., 2020)
Meta-ACL	Meta Analogical Contrastive Learning	(Kim et al., 2020)
MLCL	Multi-Label Contrastive Learning	(Małkiński and Mańdziuk, 2020)
FRAR	Feature Robust Abstract Reasoning	(Zheng et al., 2019)
-	Continual Learning	(Hayes and Kanan, 2021)
DRT	Dynamic Residual Tree	(Zhang et al., 2019a)
-	GAN	(Hua and Kunda, 2019)
-	Structural Affinity Method	(Shegheva, 2018)
PGM	Procedurally Generated Matrices	(Barrett et al., 2018)
RAVEN	Relational and Analogical Visual Reasoning	(Zhang et al., 2019a)

* No Acronym was given in the original article. We created a name to clearly refer to it in our discussion.

3.4.1 Stage 1: Imagery-Based Approach

Visual mental imagery refers to mental images that play a functional role in human cognition(Kosslyn et al., 2006). The most important characteristic of mental imagery is that human can experience mental imagery in the absence of the concurrent sensory input. Try to answer this question "how many windows are there in your house?" when you are not home (use another building you are). Most people answer this question by imagining their houses. This imaginary house is a mental imagery. Some people count the windows by mentally walking in and around their houses, while others do so by mentally rotating their houses. Whether walking in and around the houses or rotating the horse, they inspect and manipulate on this mental representation, as they inspect and manipulate the real object. Further more, mental imagery can be unrealistic, for example, some people rotate their houses upwards or downwards without the houses falling apart. For this reason, the ability of using mental imagery is important for creativity. This point makes another important characteristic of mental imagery.

Evidence from psychology and neuroscience (Kunda et al., 2013) suggests that mental imagery is frequently used by human participants to solve RPM items. Intuitively, a human participant would inspect objects in the matrix, compare them by mentally superimposing one on another, mentally transform the objects, and mentally estimate perceptual similarity. Without turning to more sophisticated techniques and terminology, this description is the most immediate one that one can think of to describe the solving process (although they might not use the term "mental imagery"). For this reason, imagery-based computational models (Hunt, 1974; Kunda et al., 2009, 2013, 2010; Yang et al., 2020; **Yang, Yuan** et al., 2022) were constructed to solve RPM and RPM-like items. In general, these models represent matrix entries by pixel images, 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).

Although these systems have proven to be effective for solving the RPM items, they appear relatively late in development of computational models for solving RPM and are still an underexplored approach in the AI community. This is partly because directly working on raw perceptual input data, especially applying various operations on pixel images and computing similarity, requires the computational power that was not available at the beginning of this line of research. Another reason might be that the theory of mental imagery has been studied mostly in cognitive psychology and received less attention in the AI community. Nonetheless, it is still a promising approach for general problem solving.

Before we dive into Stage 2, it would be better to chew on the idea of imagery-based approach a bit. The imagery-based approach provides an "in-place" solution, i.e., solving a visual reasoning problem "visually" without introducing auxiliary devices such as preprocessing of raw perceptual input. There is nothing wrong for being parsimonious because being parsimonious is a general principle of problem solving (Occam's Razor). On the flip side, a tacit consensus in artificial intelligence is that certain degree of abstraction is desirable. That is, the approach must include steps that transform the raw perceptual input into a more abstract form which reduces the complexity of problem solving. Abstraction is even deemed a hallmark of valuable AI techniques—the more abstract, the more intelligent the approach is. According to this criterion, the imagery-based approach is not intelligent at all. This could be another reason that imagery-based approach received less attention in the AI community. Because experiments also show that mental imagery plays an important role in human cognition, this brings us into a dilemma of the criteria of being intelligent in problem solving. Note that the "abstract" end of this dilemma is not proficiency in using the abstracted information but the process or ability to abstract.

The most valuable contribution of imagery-based models is not on problem solving but bringing this dilemma into light. Being mindful of this dilemma and criteria of being intelligence would put future AI systems in a more promising direction. Although this dilemma is of vital importance to AI research, there is not simple answer to it. A possible solution is that we can try to understand imagery and abstraction as two factors, correlated or independent, rather than two options contradicting to each other. A simple analogy could be made at this point to clarify this idea: in graduate math classes, instructors are undoubtedly teaching knowledge that are quite abstract; experienced instructors are able to convey the abstract knowledge in very visual languages so that it is more accessible to students. A stronger claim is that abstract concepts are always associated with some imagery representations in human thinking. This claim might not be correct in all cases, but indeed points out an important feature of human intelligence. This solution also resembles the psychometric treatment to human intelligence. AI systems can be evaluated in the both dimensions of the two factors, as human intelligence is measured. And being intelligent means that the system needs to score high in both dimensions.

Another similar solution is to view abstraction and mental imagery as two distinct and necessary cognitive processes that complement and cooperate with each other. Which of them manifests depends on the task and the subject; that a subject does not show one of them does not mean that this subject does not possess it. Conditional arguments like this is quite common in the study of human intelligence. For example, mental information processing speed (measured by special tasks) is greatly correlated with general intelligence test performance of people who score lower in the test, while processing speed is not correlated with that of people who score higher. But this does not mean the highly intelligent people cannot think fast. Either way, the dilemma is resolved by stressing that these two options are not exclusive to each other.

3.4.2 Stage 2: Logical Reasoning

Based on the discussion at the end of the last subsection, the reason why we choose logical reasoning as the second stage in this conceptual chronicle is obvious. The computational models using logical reasoning works on abstract representations of RPM-like items. For example, a entry image A in a matrix could be described by a series of propositions such as "triangle(A)=True, triangle-large(A)=False, triangle-on-the-left(A)=True, square(A)=True, square-small(A)=True, and so on". In this example, these representations are restricted to Boolean expression, but we can use more expressive formal logic like "color(A)=green, number-objects(A)=3, texture(A)=dotted, and so on". The abstract representations in these models are either manually constructed or obtained through a preprocessing module. For example, the earliest computation model for solving RPM-like items—ANALOGY (Evans, 1964)—consists of two modules and first part is for constructing such representations[9], whereas the influential models—FAIRMAN and BETTERMAN (Carpenter et al., 1990)—use handcrafted logic representations.

Each computational model in this stage has a customized formal system for representing RPM-like items. This system is either specially designed for solving a specific problem set of interest or reusing some standard systems such as regional connection calculus and scalable vector graphics. Based on a formal representation system, the three main components of logical reasoning are implemented. In the context of RPM, the entry images in the matrix are the premise and the answer choices are possible consequences, whereas the rules are to be determined. The models in this stages split into two branches according to how rules are determined.

---

[9] Interesting anecdote about Evans' work: because the memory of computer at the time was so limited, the program had to be separated into two modules which were executed serially. But the models in this stage and following stages are designed to be so.

3.4.2.1 Rule Matching

The first branch is rule matching, in which the model hardcodes finite predefined rules and matches rows and column to each of the predefined rules. For example, a predefined rule describing the number of objects could be "number-objects(A)+number-objects(B)=number-object(C)", in which A,B and C are entry images in a row or column of a 3×3 matrix. If a rule applies to the first row(s) or columns(s), it is reproduced on the last row or column to generate the formal representation of the missing entry. Many computational models have been constructed this way to solve RPM-like items (Hunt, 1974; Carpenter et al., 1990; Ragni and Neubert, 2012, 2014). This might look amazing from the current point of view because it simply would not generalize due to the predefined rules. However, this is not true from the perspective of problem solving. The readera who are skeptical about this can make analogies to other cases, like consider how many rules one need to derive the integer field and to derive the real number field, and consider also the expressive power of these number fields. The reason why these number fields can be represented concisely and completely is that when we discuss them in math, the symbols (i.e., elements in these fields) does not need to bind with concrete entities. In computational models of logical reasoning, the formal representation systems are in charge of binding symbols with entities. Thus, it is very partial to argue that rule matching models are not generalizable without referring to the formal representation system. If the geometric visual stimuli are extremely simple (as in most general intelligence tests) or the formal representation system is extremely powerful, the rule matching models will generalize as well to the whole problem domain as several rules can produce the entire integer field and real number field. This conditional argument echoes with the discussion about abstraction and imagery at the end of last subsection as they all are dealing with the problem of the level of abstraction and the operations that can be implemented at that level.

Another observation on the rule matching models is that most of the works in this branch are for cognitive modeling rather than problem solving. For example, the models in (Ragni and Neubert, 2012, 2014) are implemented on the ACT-R cognitive architecture. The purpose of cognitive modeling based on cognitive architecture lies at the information-processing level, i.e., modeling how information is exchanged between multiple cognitive function modules. But how the information is processed exactly inside each module is not really the focus of cognitive modeling. This corresponds to how a human or computation model comes up with a rule that happens to be able to solve the item at hand. Thus, the use of predefined rules would be understandable from the perspective of cognitive modelling. As a summary, the rule matching approach, though might not be able to solve all possible items in the problem domain, fulfills its duty perfectly in problem solving and in cognitive modeling.

3.4.2.2 Rule Induction

In contrast to rule matching, the second branch—rule induction—is mainly studied for problem solving (Bohan and O'Donoghue, 2000; Davies and Goel, 2001; Ragni et al., 2007; Schwering et al., 2007; Strannegård et al., 2013) with an exception that the analogy-making models are closely related to both cognitive modeling and problem solving (Tomai et al., 2005; Lovett et al., 2007, 2009). Rule induction means that the models need to discover the rules, i.e., how an entry is transformed to the next one or how the entries in a row or column are related, in a more open manner. In particular, the rules are represented as the identical and different parts between the logical structures of entry images, and/or how the different parts are changed to identical parts by transformations. The rules are also logical representations in nature. After the rules are discovered, the models have two options—they can either reproduce the rules on the last row or column to generate the answer and compare it to the answer choices, or insert each answer choice into the last row or column to induct a rule for the last row or column and compare it to the previously inducted rules. These two options corresponds to the two strategies—constructive matching and response elimination—commonly adopted by human participants (Bethell-Fox et al., 1984). The latter one is more often adopted by analogy-making models as it is similar to how an analogy is drawn by human.

Rule induction is a larger topic that goes beyond the traditional format of logic reasoning. For example, the goemetric objects and rules can be represented in a vector-symbolic architecture (Rasmussen and Eliasmith, 2011), in which geometric objects are represented as vectors and rules are inducted and applied through operations on the vectors, such as circular convolution. If we take a closer look at the details of calculation, we would find that the calculation is a different way to implement the rule induction in the models mentioned above (of course, vector-symbolic architecture has its own advantages and purposes). Another example is that reinforcement learning methods can be used to train an agent to induct the rules in matrix reasoning items (Raudies and Hasselmo, 2017), i.e., when the agent forms a correct rule (action in reinforcement learning) when it attends to certain row or column in the matrix (state in reinforcement learning), the algorithm rewards the agent.

The boundary between rule matching and rule induction is not always so clear in practice. To what extent a model is performing rule matching or rule induction depends on how the potential rules are provided to the model: if only several specific rules are provided, it is rule matching; if a huge rule space is provided by specifying some "bases" or "generators" of it, it is rule induction; and there is a lot of places in between. One can even argue that a rule induction model is in nature a rule matching model because it matches to the whole or a subspace of the rule space in some implicit way, and, thus, there is no such distinction between rule matching and rule induction. Nonetheless, there are indeed examples of rule induction that no one will consider as rule matching. For example, consider a free group in abstract algebra, in which finding an element satisfying a specific condition could be so difficult even if the generators of this free group look so simple. Other similar examples could be find in the problems of program synthesis and inductive programming.

3.4.3 Stage 2.5: Neuro-Symbolic Reasoning

The reason why we use a decimal in the title is that this stage is an intermediate stage that shares features with both its predecessor and successor. Since the influence of its predecessor and successor is stronger than this stage, this stage is relatively short and rapidly transits to its successor.

The models of neuro-symbolic reasoning consists of two modules—a neural perception frontend and a symbolic reasoning backend. The neural perception frontend (implemented as neural networks in most cases) extracts/approximates the distributions over the values of each entry in the predefined formal representation system. The symbolic reasoning backend performs probability calculation according to a predefined set of rules. In a sense, neuro-symbolic reasoning can be considered as a special case of rule matching in logical reasoning. The probability formulae in the backend are determined by the predefined rules and the output of the reasoning, such as the probability that a rule exist in rows or columns, the missing entry contains a certain value in its representation, or a certain answer choice is correct. Similarly, different implementations of frontend and backend have been used to construct probabilistic reasoning models, such as ALANS2, PrAE, VAE-GPP, TRIVR, LoGe, and NVSA (Zhang et al., 2020, 2021; Shi et al., 2021; He et al., 2021b; Yu et al., 2021; Hersche et al., 2022).

Compared to logical reasoning, neuro-symbolic reasoning clearly requires that a dedicated neural processing module is used to construct the formal abstract representation of each entry image. In addition, it also takes into account the uncertainty in perception by using probability to represent and reason. Technically speaking, neuro-symbolic reasoning is only a small step forward compared to logical reasoning. The reason why it is listed as a separate stage is that it is a natural watershed between **knowledge-based approaches** and **data-driven approaches**, because the neural perception frontend requires training data while imagery-based and logical reasoning are knowledge-based. In the next two subsections, we will elaborate on data-driven approaches.

3.4.4 Stage 3: Learning Approach

An obvious characteristic of the first three stages is that they all rely on the predefined representation systems of geometric objects and relations/rules between geometric objects. To reduce the reliance on such explicit prior knowledge, the learning approach has been introduced into the field of RPM-like tasks. This section reviews the learning models, especially deep learning, for solving RPM-like tasks. We divide the learning approach into four types according to the structures of learning models. For each type, we provide a high-level functional description that applies to all the models of the type while trying not to complicate the discussion with too much technical details. The purpose of this taxonomy is to reveal the structural evolution of the learning models (from Type 1 to Type 4), analyze the reason why it evolves this way, and, more importantly, provide a guidance for research works in this field.

3.4.4.1 Type 1

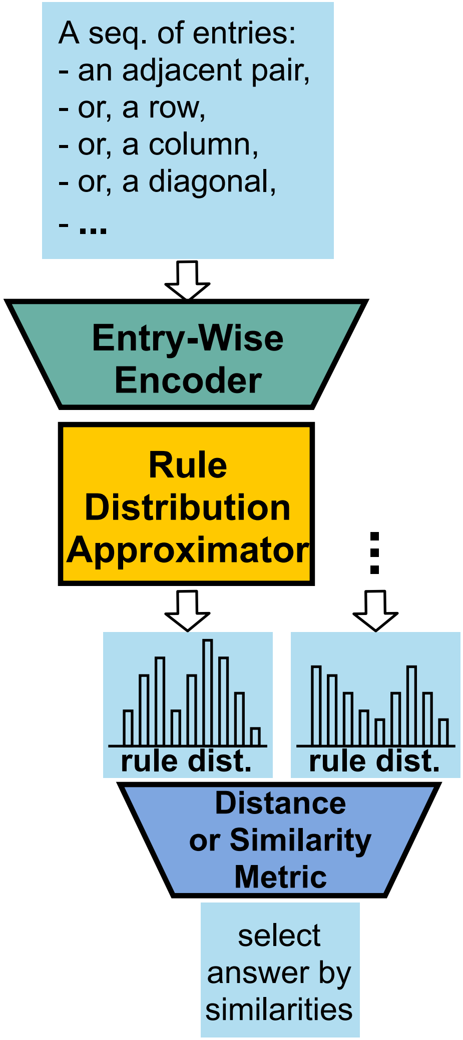


Figure 3.17: Learning approach Type 1

A natural solution to reduce the reliance on the predefined representation system of geometric objects and rules is similar to the upgrade from the logical reasoning to the neuro-symbolic reasoning: instead of approximating the distribution of attribute values of entries, we can directly approximate the conditional distribution of possible rules given multiple matrix entries (through standard or customized neural network approximators). Therefore, the rules work only as labels to distinguish between different rules and no formal representations of geometric objects and rules are involved in computation.

Two typical examples of Type 1 are Pairwise-ADV and Triple-ADV (Mekik et al., 2017, 2018) which approximate distributions of random variables of binary and ternary rules, respectively. A binary rule variable indicates whether a binary rule applies to two adjacent entries, for example, whether the objects in the two entries are of the same color, while a ternary rule variable denotes whether a ternary rule applies to three adjacent entries, such as the number of objects in Entry C equals the sum of geometric objects in Entry A and B. Another example of Type 1 is DeepIQ (Mańdziuk and Żychowski, 2019), in which the variable of rules between two adjacent entries is an ordered categorical variable (rather than binary), for example, the objects in the two entries differ by 3 units in their sizes. The random variables used these two examples are similar to using the different formal representation systems in the logical reasoning approach, but they are functionally equivalent.

The parallelism heuristic—spatial parallelism implies abstract conceptual parallelism–is commonly used to determine the combinations of matrix entries to present to the distribution approximator. 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, or general similarity metrics, like Euclidean distance, are used to measure the similarity; the answer choice is chosen so that it gives a last row/column whose rule distribution is most similar to the ones of the context rows/columns.

A diagram of Type 1 is given in Figure 3.17. Note that a entry-wise encoder is used to process each input entry individually, which is similar to the perception frontend in probabilistic reasoning. But, unlike the perception frontend, the entry-wise encoder does not necessarily output distributions over the predefined representations of geometric objects. The entry-wise encoder is to represent any latent space that can be used to approximate the rule distributions. After the entry-wise encoder encodes every entry in an input sequence, the embeddings of these entries are further aggregated and processed by the rule distribution approximator; the rule distributions of different sequences are finally compared to select the answer choice. The entry-wise encoder and rule distribution approximator can be implemented based on various neural network modules, such as CNN, ResNet and MLP. In practice, these two modules are jointly trained given the ground-truth rule labels of entry sequences.

3.4.4.2 Type 2

Unlike the approaches in Stage 1, 2, and 3, Type 1 has avoided composing computing streams that explicitly rely on the predefined formal representation systems. But it still relies on the ground-truth rule labels and the parallelism heuristic. This issue is solved in Type 2, which is free of the reliance, as shown in Figure 3.18.

This type converts an RPM into a classification problem, where the class labels are the correctness of each answer choice. In particular, when only one answer choice is included in the input, it is a binary classification problem; when all answer choices are included, it is a multi-class problem.

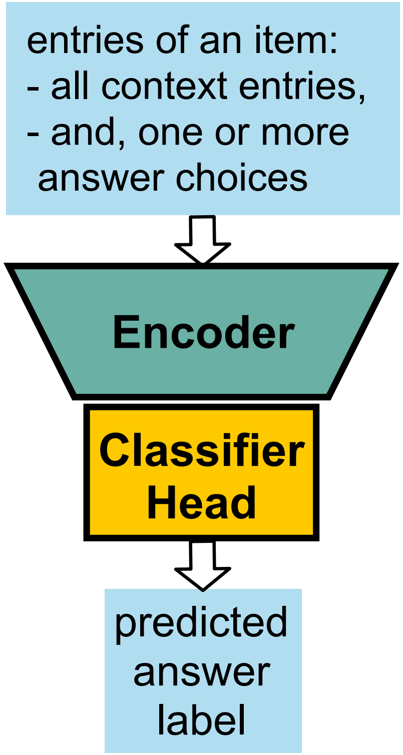


Figure 3.18: Learning approach Type 2

Readers might have noticed a difference between Figure 3.17 and Figure 3.18—the entry-wise encoder has been replaced by an encoder (not necessarily entry-wise). As the name indicates, the encoder takes as input multiple entries and, thus, the relational information between entries are thus encoded into its output. This difference gives rise to the difference between perceptual and conceptual processing. In RPM, perceptual processing is the processing of each single matrix entry, whereas conceptual processing generally involves reasoning about the relations between multiple matrix entries, i.e., the rules that govern the variation of multiple matrix entries. If one wishes to explicitly separate these two types of processing, one would have a module that attends to each entry individually and another module to aggregate the outputs of the first module, as in Type 1. This design choice is important for building computational models for visual abstract reasoning tasks. By changing the name to "encoder", we implies that Type 2 does not necessarily require an explicit separation of perceptual and conceptual processing.

Hoshen and Werman (2017) implemented the first Type-2 model using a CNN encoder and an MLP classifier, and tested it on simple figural series and RPM-like tasks. This CNN+MLP model has since been used

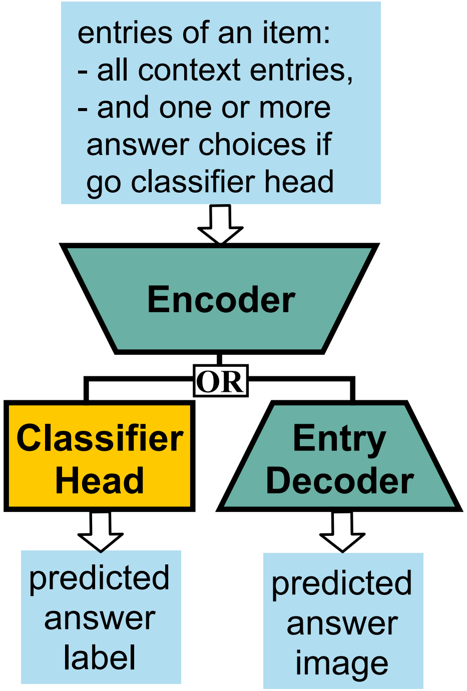


Figure 3.19: Learning approach Type 2+

as a baseline to evaluate later works. Being influenced by popular works in image classification, the early attempts to solve RPM-like tasks by learning models mostly follows the structure of Type 2, for example, the Wild-ResNet+MLP model (Barrett et al., 2018) and the ResNet+MLP model (Zhang et al., 2019a), respectively representing the binary and multi-class versions of Type 2. In the work of Hoshen and Werman (2017), they also proposed the generative counterpart of the CNN+MLP model, by replacing the MLP classifier with a deconvolutional module to generate the predicted answer image (no answer choice is provided as input in this case). We include this modification in Type 2 by upgrading Type 2 to Type 2+, as shown in Figure 3.19.

3.4.4.3 Type 3

By following the formulation of image classification, Type 2 eliminates the reliance on the ground-truth rule labels and the parallelism heuristic. However, visual abstract reasoning is a conceptually different tasks from image classification. In particular, abstract high-order relations need to be built upon raw perceptual visual input. The reason why a human participant thinks a visual abstract reasoning item difficulty is not because she cannot recognize the simple geometric objects in the item, but because the abstract concepts and relations could be complex, diverse, and hard to be extract from the simple geometric objects. In the latter case, concrete concepts are built upon complex visual stimuli, for example, recognizing daily objects in various backgrounds. Therefore, without further customization, the standard learning models for image classification is not able to give a satisfying solution to visual abstract reasoning.

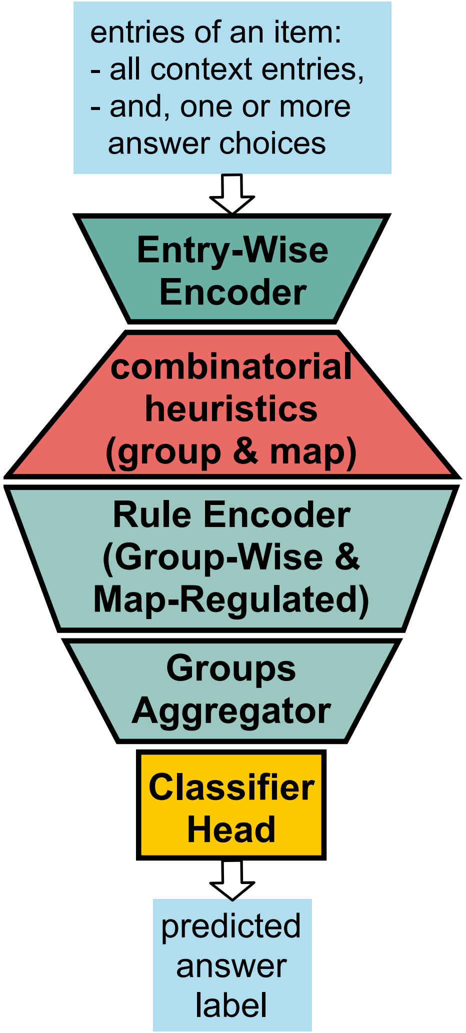


Figure 3.20: Learning approach Type 3

By comparing the Type 2 with its predecessors, which performs well on RPM (but rely on predefined formal representation systems of geometric objects and rules), we find that an unnecessary design in Type 2 is that it does not separate perceptual and conceptual processing, which has been proved to be beneficial for visual abstract reasoning in many later works. This observation leads us to Type 3, as shown in Figure 3.20. Note that one can argue that Type 3 as a special case of Type 2, by regarding everything before the classifier head as a single module. But models based on this specification generally perform better than the typical models of Type 2.

After the entry-wise encoder encodes every entry, these entry embeddings go through a combinatorial process, in which subsets of these entry embeddings are selected and fed into next module subset by subset. In Figure 3.20, we use two trapezoids of opposite orientations for the entry-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, 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 model of which entry embeddings, together as a group, would make an instance of a rule. Each group is individually processed by a singleton rule encoder to produce a rule embedding for the group. At last, all rule embeddings are aggregated for classification.

A typical example of Type 3 is the WReN model (Barrett et al., 2018). WReN takes as input all context entries and one answer choice (thus solving binary classification). The entry-wise encoder is a small CNN (plus tagging the entry embeddings with one-hot position vectors indicating the entries' positions in the matrix). For combinatorial heuristics, WReN considers all binary rules (i.e., relations between every two entries). Note that WReN does not use the parallelism heuristic, which is commonly used in other models; but the position-tagged entry embedding compensates this, because the rule encoder can easily find the non-parallel sequences through position tags and output a specific rule-embedding to indicate this for the following processing. The groups aggregator in WReN is simply a summation.

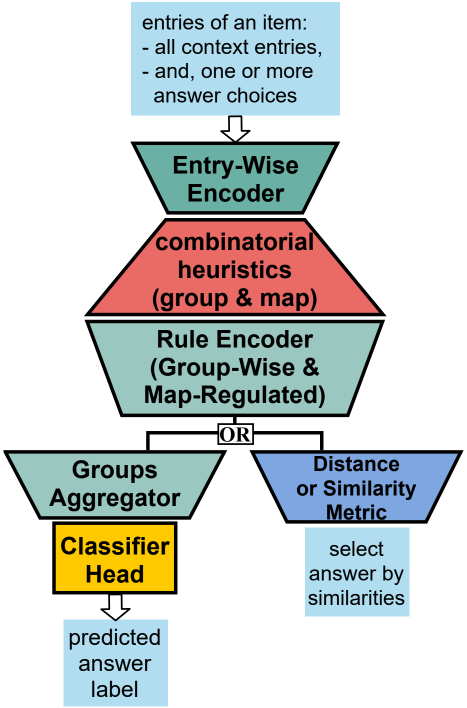


Figure 3.21: Learning approach Type 3+

Following WReN, a series of models of Type 3 have been created, using different entry-wise encoders, combinatorial heuristics, rule encoders and groups aggregators. For example, LEN (Zheng et al., 2019) considers only ternary rules for combinatorial heuristics, i.e., groups every three entries together for rule encoding, and applies gating variables to each groups in aggregator instead of tagging positions of entries (unsurprisingly, the experiment results showed that all gating variables but the ones of rows and columns were zeroed); MXGNet (Wang et al., 2020) also considers ternary rules, uses CNN or R-CNN as entry-wise encoder, relies on parallelism heuristic for combinatorial heuristics (instead of gating variables), and employs a graph-learning-based rule encoder that regards the 3 entries as a graph and computes the graph embedding as the rule embedding.

Different from the previous Type-3 models, multi-layer RN (Jahrens and Martinetz, 2020, 2019, 2018) extends the relation encoding in WReN into a multi-layer form. This is, the relation embeddings of entry groups are not aggregated into a single embedding for classification, but into multiple embeddings, which are further fed into another combinatorial module and rule encoder. Therefore, one could visualize multi-layer RN as a Type-3 model, repeating the middle three modules as many times as needed. Intuitively, higher-order relations can better be extracted through this multi-layer design.

The SRAN model (Hu et al., 2021) adopts a more complicated encoding scheme by using multiple encoders and multiple rule encoders, where entries of two context rows/columns of 3×3 matrices (6 in total) are encoded entry-wise, 3-entries-wise, 6-entries-wise by three different encoders, and the resulting entry-embeddings, 3-entry-embeddings and 6-entry-embeddings are sequentially integrated by three rule encoders into a single rule embedding, representing the rule of these two context rows/columns. The encoding scheme of SRAN, though complicated, does not deviate too much from Type 3. But, in stead of using rule embeddings to solve the item as an classification problem, SRAN directly uses similarity metrics of rule embeddings to select the answer, as in 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 Type 3+, as shown in Figure 3.21.

MRNet (Benny et al., 2021) is another Type-3 model using multiple entry encoders and multiple rule encoders, which process the input at multiple resolutions, determined by different layers' output in a CNN entry-wise encoder. The computational streams of different resolutions proceed separately and are aggregated at the end for classification.

Both these two models—SRAN and MRNet—are examples of using multiple entry encoders and multiple rule encoders. Another model—NSM (Shekhar and Taylor, 2021)—would be a better example to show the flexibility of Type 3. In particular, NSM solves the analogy-making task through two different rule encoders—a LSTM rule encoder and a modular network encoder—for the base domain and the target domain, respectively. Moreover, the structure of the modular network depends on the output of the LSTM rule encoder. These examples imply that, to build a Type-3 model, one can use not only multiple encoders but also different types of encoders, and that the multiple encoders can be assembled in more complex ways, rather than being parallel.

Readers might have noticed the words “group” and “map” in the diagrams of Figure 3.20 and 3.21. By these words, we intend to call attention to a mechanism that is pervasive in information processing 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 mapped [10] and thus to be processed in the same way[11]. These two types of decisions are interdependent on each other; more precisely, they are better to be viewed as two aspects of the same cognitive process. These decisions have to be made repeatedly at every level in information processing. Unfortunately, there might not be a centralized or universal theory for this grouping-mapping mechanism. As one can see in these Type-3 models, they all resort to some specific heuristics, which might not be always incorrect for visual abstract reasoning tasks.

3.4.4.4 Type 4

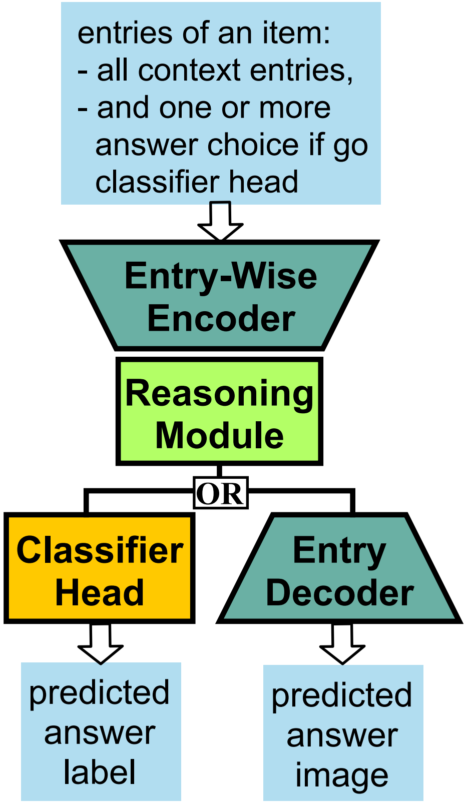


Figure 3.22: Learning approach Type 4

Now, it is a good time to look back at the path that we have walked down for reviewing data-driven approaches and summarize how we come here:

---

[10] or aligned, or corresponded; we use “map” to resonate with structure-mapping theory of analogy making; i.e., if two entities in the base and target domains are mapped to each other, then they are analogous to each other.

[11] i.e., processed by the same module to force the analogical relation between them.

- From neuro-symbolic reasoning to Type 1: we eliminate the need of predefined representation systems of geometric objects and rules, but introduce the need of ground-truth rule labels. Parallelism heuristics is also inherited.
- From Type 1 to Type 2: we eliminate the need of the ground-truth rule labels and parallelism heuristics, but the models do not perform well, because we use models for image classification, which is a fundamentally different task from visual abstract reasoning. The problem is solved in an image-classification way.
- From Type 2 to Type 3: we separate perceptual and conceptual processing to make the model more suitable for visual abstract reasoning. Although the models perform reasonably well, specific grouping-mapping mechanism (or combinatorial heuristics) are needed for solving different RPM-like tasks.

In this path, every time we want to eliminate the need of some prior knowledge, we introduce one or more neural networks modules to learn it from annotated data. This general solution in Stage 3 makes the procedural aspect, i.e., the process of computing, of solving RPM less of problem as the procedure can be interpolated from the input and expected output, through learning. The critical research point of the learning approach that determines the outcome of learning is thus shifted to the structural aspect. In visual abstract reasoning, the structural aspect includes hierarchical structure of processing, for example, the separation of perceptual and conceptual processing. With another layer of analogical processing (i.e., higher-order processing involving multiple relations), it would make a more complete hierarchy. In another dimension, the structural aspect also includes grouping-mapping mechanism we mentioned above. If one cannot abstract the task into these factors in the structural aspect and identify the "atomic" ones that can be easily solved through learning, the resulting learning model would not be effective and generalizable in the entire problem domain.

As indicated above, a remaining factor unsolved in Type 3 is the combinatorial heuristics. Type 4 attempts to solve it by regarding grouping-mapping mechanism and rule encoding as a single "atomic" factor that can be learned through a single module—reasoning module, as shown in Figure 3.22. The reason for combining them is empirical and pragmatic, because they are interwoven and it is hard to say the former determines the latter, or the other way around. Since the reasoning module of Type 4 contains no grouping-mapping heuristics, its output does not necessarily indicate various rule among entries, and thus cannot be processed as in Type 3/3+. Thus, supervising signals are directly applied on this output. If we go back to see the structure of Type 2, you will find that Type 4 resembles Type 2 in appearance. Nonetheless, Type 4 is much more effective than Type 2 on RPM and it takes many trials and errors to settle on this solution. It has now become a relatively stable solution to visual abstract reasoning, and different core techniques have been used to implement the reasoning module. We summarize the works into four categories using distinct reasoning kernels.

3.4.4.4.1 Reasoning Kernel 1: CNN

CNN has been a basic tool to extract features from raw perceptual input, and the extracted features are not only relevant for solving specific downstream tasks, but also representing correlations in the input. Solving visual abstract reasoning tasks is also to process correlations among raw inputs. Theoretically, CNN would have been an effective solution to RPM-like tasks. However, several early influential works (Barrett et al., 2018; Zhang et al., 2019a,b) argued that CNN and CNN-based models are not capable of solving RPM-like tasks. [12]. Since then, the research has been mainly focusing on other solutions. Ironically, after several years of exploration, Spratley et al. (2020) proposed two Type-4 models—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 processing and the perceptual processing are separated. Taking Rel-Base as an example, its entry-wise encoder is a CNN module and its reasoning module is also a CNN module; all the entry embeddings are first stacked together and then convolved with convolution kernels in the reasoning module. But the baseline CNN-based models do not have this artificial separation. Therefore, we conjecture that the outstanding performance of many non-CNN models is not because they found better solutions than CNN, but because they separate perceptual and conceptual processing. On the flip side, another implication is that when using a single CNN module for both perceptual and conceptual processing, it is an extremely difficult task to learn this separation from data, i.e., learn the hierarchical structure of the task and how the information at each level is correlated. However, from the perspective of general problem solving, it would be impossible for us to know when the perceptual-conceptual separation lies for every possible task; in this case, we would have to use a single huge monolithic model; and how such a model can be trained effectively would be an important future research question.

3.4.4.4.2 Reasoning Kernel 2: LSTM

A typical Type-4 model is the CNN+LSTM+MLP model (Barrett et al., 2018). This model takes as input all context entries and one or more answer choices. Each entry 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. This model is also used as a common baseline in many later works. LSTM has also been combined with other modules: Double-LSTM (Sekh et al., 2020) uses two LSTM modules, which each specialize in different rule types and are coordinated by an extra module trained to predict the rule type[13]; ESBN and NTM (Sinha et al., 2020; Webb et al., 2020b), combining LSTM with external memory modules, can also be used as the

---

[12] This is also why the CNN+MLP and ResNet+MLP models have been constantly used as baselines.

[13] The reliance on ground-truth rule labels slightly deviates from our definition of Type 4.

reasoning kernels in Type 4.

3.4.4.4.3 Reasoning Kernel 3: 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 reasoning kernel is the ARNe model (Hahne et al., 2019). It extends the Type-3 model, WReN, by inserting between the entry-wise encoder and the combinatorial heuristics a self-attention module. Note that although ARNe inherits the combinatorial heuristic of WReN, it is no longer a Type-3 model because the self-attended embeddings no longer represent individual entries. Instead, each self-attended embedding contains information about all the matrix entries, and should better be considered as 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, simply aggregating the input and predicting the answer. With hindsight, a reasonable order should have been first testing the self-attention module with a simpler classifier head rather than WReN.

A similar example is the HTR model (An and Cho, 2020), where an R-CNN entry-wise encoder is used to extract all geometric objects in each entry and two self-attention-based sub-modules are used to move the reasoning from object-level to entry-level and from entry-level to matrix-level. The first sub-module takes as input the object embeddings in a single entry and sums up the self-attended object embeddings as the entry embedding. Unlike ARNe and WReN solving RPM as binary classification, HTR solves it as multi-classification. Therefore, the output of the second sub-module contains 8 embeddings corresponding to the 8 answer choice. These 8 embeddings are fed into a contrastive classifier head (Zhang et al., 2020) to predict the answer label.

A more general example is the Neural Interpreter model (Rahaman et al., 2021), which implements its most basic building block "function" as a self-attention module associated with two learnable vectors, which affects the module's computation and its access to input data, respectively. The self-attention modules are analogous to functions in programming language (as the term "interpreter" indicates), with one vector defining the function body and the other defining the function signature (type-matching particularly). A neural interpreter is composed of multiple iterations of a finite set of functions. As the original self-attention in Transformer, it converts a set of embeddings into a set of corresponding embeddings decorated with relational information. Neural interpreter was tested on RPM as binary classification. A CNN entry-wise encoder is used to produce entry embeddings. As in BERT (Devlin et al., 2018), a classification token is included in the input embeddings, whose corresponding output embedding was fed into a linear classifier head.

3.4.4.4.4 Reasoning Kernel 4: Multi-Head Rule Detector

The last reasoning kernel is closely related to the rule encoder of Type 3. Recall that the combinatorial heuristics module in Type 3 groups the entry embeddings into multiple groups, and each group is separately processed by the rule encoder to obtain a rule embedding for this group. Although this rule encoder has 1-in and 1-out, it is responsible for recognizing and encoding all the possible rules that might occur in the input. Recall that, by moving from Type 3 to Type 4, we intended to eliminate the reliance on combinatorial heuristics. An natural alternative solution could be an "all-in-all-out" rule encoder (rather than 1-in-1-out), which takes as input all the entry embeddings of a matrix (no grouping) and outputs all the possible rules. The relationship between "1-in-1-out" and "all-in-all-out" is analogous to the relationship between image classification versus object detection, where multiple objects exist in the image. Particularly, the new rule encoder can have multiple output heads, where later supervising pressure can be applied to force each head to represent a specific rule or specific rules. Therefore, we refer to this reasoning kernel as multi-head Rule detector. This kernel is underrepresented because we found only one model using this kernel—the SCL model (Wu et al., 2021), but it is very efficient for visual abstract reasoning.

3.4.5 Stage 4: Data Manipulation

The reported performance of some learning models in Stage 3 has already surpassed human performance under certain circumstances. However, the unreported or non-highlighted performance is far from satisfactory. A serious issue is that abstract concepts are not learned by these models because they do not generalize well when the abstract concepts are presented in different perceptual stimuli. This type of generalization is fundamental to visual abstract reasoning and also a hallmark of human intelligence. Therefore, the exploration has never stopped. Since the four types of learning models in the last stage explored many structural possibilities for building learning models, we have observed more and more efforts on studying the problem domain per se and how it is solved by human. This is perfectly understandable because when one realizes that all the existing tools do not work, she will naturally scrutinize the problem per se and try to understand why it is different from previously solved problems. These efforts result in the works of Stage 4, which utilize the features of visual abstract reasoning task that do not necessarily exist in other tasks. These efforts also resonate with the upward-spiral pattern we mentioned at the beginning of this conceptual chronicle as these task-specific features are also heavily used in the approaches in Stage 1 and 2, though in different ways. In particular, datasets of RPM-like items are delicately manipulated to present the task to learning models in a similar way of how human perceive and conceptualize RPM items. This way, the works in Stage 4 could force the models to learn abstract concepts and specific visual stimuli, distinguish between them, generalize the abstract concepts to the entire domain, and, finally, build the ability on the entire problem domain.

3.4.5.1 Auxiliary Training

For the models 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-like item, which is a multi-hot vector indicating the attributes of geometric objects and rules in this item. These meta-targets are usually accessible in algorithmically-generated datasets, such PGM and RAVEN. The learning models can thus be trained on the answer labels and meta-targets simultaneously. The training on meta-targets is often referred to as auxiliary training in literature.

Intuitively, this extra supervising signal can boost the accuracy of the answer-label classifier head. Auxiliary training was first tried with 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' accuracies (Barrett et al., 2018). Similar observations on PGM were also found in other studies (Pekar et al., 2020; Hahne et al., 2019). In particular, the ARNe model would not even converge without auxiliary training.

However, the effect of auxiliary training is still inconclusive. Benny et al. (2021) showed that auxiliary training on PGM could only increase the accuracy of 1-rule items but decrease the accuracy of multi-rule items. This could cause the decrease of the overall accuracy when the dataset is composed of complex RPM-like items. Besides being affect by rules, the effect also differs between datasets. It has been reported that the auxiliary training would generally decrease the performance on the RAVEN dataset (Zhang et al., 2019a,b; Zheng et al., 2019; Wang et al., 2020), with one exception (Kim et al., 2020), which used a special contrastive loss and will be discussed later. Besides, Małkiński and Mańdziuk (2020) also showed contradictory results that when the meta-target is encoded in a sparse manner (the above works 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.

3.4.5.2 Disentangled and Generative Representations

The neuro-symbolic reasoning in Stage 3 has been frequently using standard neural networks, such as autoencoder and CNN, as the perception frontend to construct representations of entry image with explicit symbolic meaning. In contrast, as we mentioned in Type 1 of Stage 4, the symbolic meaning of encoders' output is not guaranteed. In addition to representations with symbolic meaning, disentangled and generative representations are used in Stage 4. For example, the Type-1 model, DeepIQ (Mańdziuk and Żychowski, 2019), uses a variational auto-encoder (VAE) as its encoder, which is pretrained on entry images of the Sandia dataset and kept frozen when the rule approximator is trained later.

Several advantages of disentangled and generative representations in RPM have been reported, such as data efficiency (van Steenkiste et al., 2019), robustness to distracting attributes (Zheng et al., 2019) and better OOD generalization (Steenbrugge et al., 2018). Disentangled and generative representations of entry images are usually obtained through VAE or its variants. For examples, in Type 3, β-VAE, FactorVAE, β-TCVAE and DIP-VAE were pretrained on entry images and the frozen encoders were combined with WReN (Steenbrugge et al., 2018; van Steenkiste et al., 2019); 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) (Pekar et al., 2020). For Type-4 models, the VAE is usually jointly trained with the reasoning module, for example, the aforementioned ESBN model (Sinha et al., 2020), and the LoGe model (Hersche et al., 2022), which uses vq-VAE as its encoder and decoder. Another special example of Type 4 is the Rel-AIR model (Spratley et al., 2020), which integrates into its encoder an Attend-Infer-Repeat model (Eslami et al., 2016)—a model that can bee thought of as iterative VAE.

3.4.5.3 Contrastive learning and Manipulating Data

In addition to supervised learning, contrastive learning has also been used for solving RPM. We need to point out that the techniques of contrastive learning have been highly adapted to employ the structural and analogical characteristics of RPM and thus might not strictly follow the paradigms of contrastive learning. Particularly, the characteristics 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.

3.4.5.3.1 Intra-Item Contrasting: Row/Column Contrasting

The minimum structure that can be contrasted is rows/columns of a matrix. This type of contrasting was first attempted in the MCPT model (Zhuo and Kankanhalli, 2020), where 8 answer choices are inserted into the 3×3 matrix to obtain 10 rows/columns (2 context rows/columns and 8 answer choice rows/columns). The context rows/columns are assigned pseudo-label 1 and answer choice rows/columns are assigned pseudo-label 0; and this newly constructed pseudo-dataset of row/columns is learned by a Type-2 model, assuming that only one “mis-assigned” pseudo-label for the correct choice row/columns does not affect the final result of learning. To solve RPM, the answer 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 (Kiat et al., 2020) 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 different ways, such as rows/columns from different RPM-like items, randomly shuffled rows/columns of the same RPM, or filling the third row/column with a random non-choice entries. In PRD, a Type-2 model is used to learn a metric to measure the similarity between the two rows/columns in a pair. 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 (Pekar et al., 2020) contrasts the answer choice rows/columns completed by the generated answer to the answer choice rows/columns completed by the given answer choices.

The rationale of moving from single-row/column to double-row/column contrasting was also exemplified by the LABC training/testing regime (Hill et al., 2019), which makes the contrasting 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-like item is adapted by muting some digits of its meta-target vector and regenerating the incorrect answer choices according to the muted meta-target vector. Since meta-targets represent the rules and geometric objects that are used to generate RPM items, the newly-generated answer choices are partially correct. This way, the model will have to compare such answer choice rows/columns and the context rows/columns to find the correct answer, instead of only seeking meaningful variations in the answer choice row/column as in the single-row/column contrasting. LABC makes this idea more systematic by introducing the concepts of semantically and perceptually plausible answer choice corresponding to muting different subsets of mete-target digits and using distracting objects and rules.

3.4.5.3.2 Intra-Item Contrasting: Matrix Contrasting

Instead of contrasting rows/columns, we can also contrasting the matrices completed by each answer choice. This is essentially contrasting the answer choices in the context of context entries. The Type-2 model, CoPINet (Zhang et al., 2019b), is the first model performing such contrasting. The contrasting in CoPINet is two-fold—contrastive representation and contrastive loss. First, for an RPM-like item, the embeddings of the matrices completed by each answer choice 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 matrices completed by correct and incorrect answer choices and increases the gap between their predicted values. This contrastive loss function could be easily embedded into models of parallel computation streams, for example, the aforementioned HTR model (An and Cho, 2020).

We need to point out that row/column contrasting and matrix contrasting are not exclusive. For example, the DCNet model (Zhuo and Kankanhalli, 2021) first uses row/col contrasting to compute the matrix embeddings and then uses the matrix contrasting to predict the answer.

3.4.5.3.3 Inter-Item Contrasting: Single-Label Contrasting

The above contrasting has been restricted within a single RPM-like item. The contrasting can also be between multiple items. The ACL and Meta-ACL (Kim et al., 2020) 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 entry as an empty image), X_i an incomplete matrix obtained by replacing the i-th entry with a white-noise image, and X' an incomplete matrix obtained by randomly reordering the entries 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 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 (Małkiński and Mańdziuk, 2020) 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. 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 incorrectly completed matrices is performed as in CoPINet. These two types of contrasting losses are jointly optimized.

3.4.5.4 Other Dimensions of Manipulating Data

Besides contrasting, there are also other dimensions of manipulating data. For example, the FRAR model (Zheng et al., 2019) utilizes a reinforcement learning teacher model to select items from an RPM-like item back to train a student model. 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-like datasets have also been examined in the setting of continual learning. For example, the RAVEN dataset can be divided into 7 batches according to its spatial configurations and the models are trained with different methods to mitigate forgetting when sequentially learning the 7 batches in different orders (Hayes and Kanan, 2021).

3.4.6 Summary

The food for thought to share with the readers is that the study of the problem domain and the exploration for general solutions are both important for the overall advance in this field, as indicated by the upward-spiral pattern in the conceptual chronicle of computational models reviewed above. On one hand, the technical development always explores new methods, on the other hand, it inevitably revisits the old ideas again and again until the problem is perfectly solved. Therefore, the most recent models are not necessarily superior to the traditional ones in nature, and the early approaches, like the imagery-based approach, might trigger the next cycle of technical development in future research.

3.5 Discussion

After a historical overview of RPM and the problem domain represented by RPM in Section 3.2 and 3.3 and a conceptual chronicle of computational models for solving this problem domain in Section 3.4, we will zoom away in this section to discuss more general topics related to intelligence testing and AI systems. A good introduction to these topics is through a fundamental cognitive process—analogy making. In particular, we list the following analogies about intelligence tests and AI system:

- Analogy A—Intelligence Test : Human :: Intelligence Test : AI system
- Analogy B—Intelligence Test : Human :: AI Test : Human
- Analogy C—Intelligence Test : Human :: AI Test: AI System
- Analogy D—Intelligence Test : AI System :: AI Test : Human
- Analogy E—Intelligence Test : AI System :: AI Test : AI system
- Analogy F—AI Test : Human :: AI Test : AI system

The AI tests in the analogies above specifically means the tests that are inspired by human intelligence tests and specially designed to evaluating AI systems, for example, PGM and RAVEN datasets. These AI tests represent the motivation of testing AI systems in a similar way of human intelligence testing. To be rigorous, we enumerate all the possibilities of permutating tests and test-takers in the above analogies. These analogies represent research questions in different fields. For example, cognitive scientists might be interested in A; test developers might be interested in B and E; AI researchers, might be interested in A, C, E, and F; and some people might be interested in D simply for exploration purpose. Many of works reviewed above allude to one or more of these analogies. But most of them did not take one more step to examine whether these analogies hold or under what conditions they holds. In this case, the result of these works should be interpreted with caution. When it comes to AI testing, We are particularly interested in Analogy C. It describes a situation where human intelligence testing and AI testing are similar and common test theories could possibly apply to both cases. This analogy further gives rise to two general dual topics that are important for building and testing AI systems, respectively:

- How tests measure subjects: the validity of measuring AI in a similar way human intelligence is measured;
- How subjects solve tests: the implication of human intelligence for building AI systems.

3.5.1 The Validity of AI Testing

Analogy C—Intelligence Test : Human ::AI Test: AI System—calls attention to the connection between human intelligence testing and AI testing. It describes a situation where AI tests based on human intelligence tests are used to evaluate AI systems, as human intelligence tests are used to measure human intelligence. However, whether this analogy holds remain largely unknown to us. If they are, conclusions about human intelligence can be translated to AI systems. For example, one can claim that an AI system has the ability of visual abstract reasoning if the system passes the tests of the algorithmically-generated datasets mentioned above. Analogy C is best represented by the learning models in Stage 3 because the learning models are mainly evaluated through specially designed AI tests, such as PGM, RAVEN, and Sandia. Most of the works discuss their AI systems and contributions in the background of human cognitive abilities, and attempt to draw the conclusions that are comparable to human intelligence when the AI systems perform well. Unfortunately, when we are enjoying the acclamation, an elephant in the room is still in the room—the analogy simply does not hold and there is no validity in building and evaluating these models in the way they are currently built and evaluated. Note that the word "validity" is two-fold: on one hand, it is the validity in psychometrics; on the other, it is practically meaningless. We will now elaborate on this using learning models as an example.

To prove the idea that the AI testing in the reviewed works is psychometrically invalid, we check if the determinants of validity of human intelligence testing hold for AI testing.

- The first determinant is that human intelligence tests, as other psychological tests, is to measure individual difference on some tasks. Statistical evidence show that the performances on many tasks are correlated, and experts use the word "intelligence" to denote the latent factor or factors that cause the correlation. In other words, it is humans' behavior that comes first; then the word "intelligence" is abductively defined to explain humans' behavior. When an AI system shows similar behavior on the tasks which are comparable to human performance on these tasks, it is not necessarily the same factor(s), i.e., human's intelligence factor(s), that is behind the behavior of the AI system. To satisfy the first determinant in AI testing, we needs to show that the underlying mechanisms are the same or equivalent in all cases. Otherwise, we need to be more cautious when we are describing the AI system's ability and explicitly distinguish it from human cognitive abilities.

- The second determinant is the requirements for designing human intelligence tests: human intelligence tests are usually short to prevent the participant from being exhausted; the stimuli in intelligence tests are diverse and there is usually no repeating stimulus in a single test; meanwhile, the stimuli in intelligence tests are also concise so that it does not introduce confounding factors; the items need to be evenly spread on the spectrum of difficulty so that people at different ability levels can be measured; and so on. All these requirements contribute to the validity of intelligence tests and are not easily satisfied in AI tests. An exception is the Cognitive Design System Approach by Embretson (2004), but this approach has not used to develop any test for AI systems.

The determinants listed here are by no means complete given the complex nature of human intelligence testing, but are sufficient to break the analogy between human intelligence testing and AI testing.

Given the fundamental distinction between human intelligence testing and AI testing, we might simply abandon the idea of establishing the validity by comparing AI testing to human intelligence testing. Instead, as most works in AI, we analyze AI systems for solving intelligence tests and intelligence-test-like datasets purely from the perspective of problem solving, and claim that these AI systems are more capable of solving the tests or datasets than human participants. However, this brings us back to an old issue: the AI systems are specially prepared or trained on the items that are similar to the one used for testing, whereas testing items are kept secret from human participants, let alone training. For visual abstract reasoning, no AI system has shown performance that is comparable to human, especially when generalizing an abstract concept to new visual stimuli that were not associated with this concept before.

Nonetheless, we can still argue that these AI systems are useful because they can at least act as automatic tools to free humans from simple repeating tasks in our daily life. However, this is also not true because intelligence tests, especially general intelligence tests, are designed to distant from the our daily activities so that the result is not affected by one's previous experience. Thus, the ability to solve intelligence test items would not be able to assist human in most cases. Moreover, a cognitive ability or general intelligence does not correspond to a specific clearly defined task that is constantly repeating in certain scenario. Instead, it is abstracted from various daily activities. That is, it is common but also very sparse across various daily activities, and, more importantly, deeply interwoven with other abilities. There is simply no such simple clearly-defined repeating tasks where these AI systems can be applied. For other complex ill-defined tasks, these AI systems also need to be integrated with various other AI systems of different abilities. This kind research, though valuable, is still infeasible at the current stage of AI.

We can try to continue this debate by proposing more contributions and purposes of building AI systems for solving intelligence tests or intelligence-test-like tests. As long as the contribution is relative to human intelligence, we can always come up with a reason to refute it (except that the contribution is pure scientific exploration). Unfortunately, comparing to human intelligence is unavoidable on our way to implementing human-level AI. It seems that we have come to a dead end.

The solution lies in the theory of analogy making and the origin of intelligence tests. Let us first check the analogy-making aspect of Analogy C to see if we interpret the analogy correctly. One of the most important theories of analogy-making is the structure mapping theory by Gentner (1983). It emphasizes the similarity between the relations in the base and the target domains, rather than the literal similarity between objects in the base and target domains. In particular, the corresponding objects can be starkly different in a literal sense without compromising the strength of the analogy, when the corresponding relationships are similar. This seems trivial to humans who know how to make analogies. But people indeed make mistakes by relying on literal similarity rather than relational similarity when interpreting analogy. In fact, we did in interpreting Analogy C above. We started from corresponding human intelligence tests with AI tests by literal similarity, i.e., they are items to solve. We then took a simple relation "human solves intelligence tests" in the base domain and translated it into the target domain. After a thorough analysis, we found everything went wrong. We just made the very mistake that is just pointed by structure-mapping theory. Thus, interpreting analogy correctly might not be trivial at all in practice.

The correct interpretation starts from studying the relations in the base domain, which can be clarified by a revisit to the origin of intelligence tests. Modern schooling is actually a new manner of eduction compared to the whole history of education. It does not exist until the 20th century. At the beginning, educators found that some children had a great deal of trouble learning in this manner. In order to select the students who were suitable for modern schooling, the French Education Ministry hired Alfred Binet. The solution Binet provided was to test children's ability to solve problems that could be commonly solved by children at certain ages, determining the children's mental ages. The ratio of mental age to chronological age was used as an index to select students for school education. This index is the prototype of today's intelligence quotient. Therefore, the origin of intelligence tests tells us that intelligence tests were developed to measure individual difference of learning ability under a certain circumstance (school education) relative to the average of a certain group of people (peers). This definition echoes our discussion of RPM in Section 3.2.

While this definition of intelligence tests seems complicated, it does accurately describe the relations in the base domain of Analogy C. Now, let us check the target domain for a similar relational structure. The general idea of the target domain is undoubtedly to test AI systems. We can try to extract from the target domain the counterparts of the concepts in the definition of intelligence tests. The most important two concepts in the definition of intelligence tests is definitely “learning ability” and “individual difference”. “Learning ability” of AI systems is a clear concept because it is native to the learning models. “Learning ability” has been considered as an integral part of AI systems (though the “learning ability” of AI systems might be the different from human learning ability). Thus, “learning ability” does not pose any problem to us. “Individual difference” of “learning ability” of AI system is less clearly defined because of the heterogeneous nature of various AI systems. Note that, in contrast to human intelligence testing, the inherent “learning ability” cannot be sufficiently reflected in the final outcome of learning. This problem can be solved if we considered the dual concept of ability—difficulty. Put simply, if we have items at various levels of difficulty, we can use human ability test items like a ruler to measure people’s ability. On the flip side, if we know people at different levels of ability, we can use these people’s response to these items to determine the difficulty of these items. That is, ability and difficulty are defined relative to each other. We are so familiar with difficulty in AI research because we have experienced so much of it. In particular, when evaluating AI systems’ learning ability, the concept of difficulty is reified as learning tasks. We would say that a learning task is difficult to a specific AI systems or to a class of AI systems. In practice, learning tasks can be defined differently, such as different datasets, different ways to present datasets, and access to other resources. A good example of learning tasks is the different generalization regimes of PGM and RAVEN datasets Barrett et al.; Zhang et al., which correspond to different conceptual distances between the abstract concepts in training and testing. The more distant, the harder the learning task. Now, we can look back at the the ruler to measure human intelligence, on which the marks are individual test items. Therefore, to interpret Analogy C, we can make the correspondence between human intelligence test items and learning tasks of AI systems. In contrast to previous interpretation of Analogy C, this correspondence is not based on literal similarity but derived from the relational structures in the base and the target domain. This correspondence is extremely important for us to establish the general testing theory of AI systems, but might not be obvious from literal meaning of Analogy C. We now can interpret Analogy C as human intelligence tests measure human intelligence as AI tests of learning tasks measure AI systems.

It is important to point out that this interpretation of Analogy C is not just a rhetoric or an arbitrary makeshift. It calls attention to two basic factors that one needs to consider to establish a test theory—what is being measured? what is used to measure it? For the first question, we definitely want to measure the “learning ability” of AI systems. For the second question, we have a great many existing learning tasks for AI systems. The context to answer the second question is subtly different from human testing and more complex. First, when we are evaluating an AI system on a learning task, we are interested in the overall performance rather than the response on a specific instance of this task. For example, for an image classification task, we would compare the overall accuracies of two AI systems to conclude that one is more capable than the other. We would not make such conclusion because one system gives a correct prediction for a specific image while the other does not, unless this instance (the image) is fundamentally different from other instances and possibly posing more demands for processing. In that case, this instance would make a separate learning task. In both cases, the correspondence between human intelligence test items and learning tasks for AI systems remains unchanged. The context of AI testing is more complex than human intelligence testing because there exist various learning tasks and various AI systems to solve them, but, for now, not every AI system is designed to solve every learning task. And for practical purposes, we need these specialized AI systems in our society rather only pursuing the ultimate goal of human-level AI. For human intelligence tests, although people might perform extremely well on some subtests but terribly on the others, the tests are valid measure for all human beings. But, currently, one cannot design an AI test that applies to all AI systems. What we can do now is to identify problem domains and fundamentally different learning tasks in the domain, which can be used to compose tests for AI systems. When AI technology enters the era of Artificial General Intelligence (AGI) in the future, we can design AI tests using learning tasks across multiple problem domains.

In general, this interpretation of Analogy C allows us to establish a testing framework for AI systems, which is similar to the testing theories in human intelligence testing. This framework requires extra efforts to study problem domains and, more importantly, study cognitive information processing to identify various learning tasks in the problem domain. Therefore, it is naturally a interdisciplinary research direction. this framework proposes a much higher standard than how AI systems are tested now. Although it requires extra efforts to implement, it will make sure that we are making concrete progress.

3.5.2 The Implication of Human Intelligence for Building AI systems

Although the history of human intelligence testing is much shorter (approximately 100 years) compared to the time intelligence exists, humans' intelligence test scores have shown a substantial increase (Flynn effect). Many efforts have been made to find what is responsible for this increase. These efforts are important not only for human development but also for AI systems from the perspective of AI testing. Specific social changes have been used to explain Flynn effect, such as television, computer games, changes in school education and so on. Most of these explanations do not hold up because these social changes are not accompanied by the changes in intelligence test scores. Interestingly, the change in testing scores does correlate with to the changes in human's height, birth weight, and infant mortality in a more than general sense. Thus, the increase in intelligence test scores might be attributed to the same factors responsible for height, birth weight and infant mortality—i.e., improved living conditions such as food and medical care(Raven, 2000).

When we are reviewing the development of AI, we are facing the same meta-question—what causes the development—that is not well answered. either. We could conclude that the recent improvement of AI is due to the increase of computational power and massive amount of data generated through internet. This explanation is not so different from attributing the increase of human intelligence to improvement of living conditions, which is not very operable for theoretical AI research. Apart from computational power and data, most of knowledge in basic science that are used in the cutting-edge AI technologies have been there for decades. Therefore, it is hard to find a theoretical factor that promoted the development of AI.

A hypothesis from the social studies that was proposed to explain humans' cognitive development can better explain the development of AI than other explanations. **The Challenge Hypothesis** (Hunt, 2010):

> Intelligence is developed by engaging in cognitively challenging activities. Environments vary in the extent to which they support such challenges, and individuals vary in the extent to which they seek them out.

The statement of the hypothesis is, though concise, but full of wisdom. In the last decades, the development of AI have been definitely accompanied by tasks that were initially challenging to AI systems, such as facial recognition and spam filtering, and later solved. These tasks did not exist before the era of AI. This argument echoes the emphasis in the last subsection on identifying and collating learning tasks for AI testing.

The second half of the challenge hypothesis—"individuals vary in the extent to which they seek them out"—is even more interesting. In the studies of human cognitive development, there is a somewhat surprising empirical result—eductive ability is more easily influenced by appropriate educational and developmental experience than reproductive ability. In particular, researcher found that educational self-direction, in which students are responsible for deciding what they need to learn, how they learn it, and what are goals, and complex educational activities (e.g., challenge and reasonable learning tasks) give rise to a cyclical development in cognitive ability (Raven, 2000). These studies shed light on a possible promising future trend in AI research, in which AI systems take the initiative to seek out learning tasks in the challenging environment that provide the most efficient development. This trend implies a fundamental change to the paradigm of AI systems by shifting from learning specific tasks to interacting with the environment(Laird et al., 2017)

CHAPTER 4

The Interplay between Perceptual and Conceptual Processing in Visual Abstract Reasoning

4.1 Introduction

For the second research question in Chapter 2, I propose to use the interplay between perceptual and conceptual processing to solve visual abstract reasoning tasks and investigate its effects on trivial and nontrivial generalization regimes. This is quite an overarching research question which might be studied from different perspectives and using methods in different areas. In this chapter, I will particularly explore the research question by realizing these abstract ideas—"the interplay between perceptual and conceptual processing", "trivial generalization", and "nontrivial generalization"—in machine learning models and experimenting on datasets that were reviewed in Chapter 3.

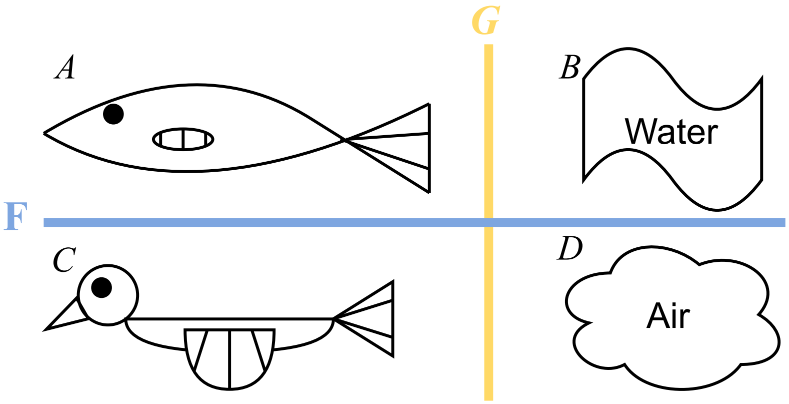


Figure 4.1: A is to B as C is to D. But in what ways?

The works in this chapter is deeply motivated by my works in human analogy-making tasks, which will be discussed in Chapter 5, and the systematic review on computational models for RPM and RPM-like tasks in Chapter 3. To explain it intuitively, consider the simple analogy—*A* is to *B* as *C* is to *D* in Figure 4.1. What relationships do you notice? Initially, one might recognize fish and birds as animals that move around in the water and air, respectively, and compare them fish and birds in terms of their appearance, i.e., similar body structures of their heads, fins/wings, and tails. However, to fully understand the analogy the analogy, one need to recall that birds get propulsion from their wings, whereas (most) fish do so using their tails. This alternate mapping (bird wings to fish tails, and bird tails to fish fins) is an outcome of conceptual processing of the initial perceptual representation of body structures, which then further influence perceptual representation of which similarities we emphasize and how we build our analogical representations. In generally, perceptual and conceptual processing can go on and on, changing each other, until consistent representations at both perceptual and conceptual levels are achieved.

Analogy-making—the process of comparing and contrasting two or more things to enable additional relational inferences of various kinds—has been argued to be one of the foundational aspects of human intelligence Hofstadter and Sander (2013). Theories of human perceptual and conceptual systems (e.g., Barsalou et al., 1999), including in the context of analogy-making (e.g., Carpenter et al., 1990), have made observations about this kind of bidirectional interplay between perceptual and conceptual processing, and forms of this interplay have also been explored in knowledge-based (i.e., symbolic) computational models of analogical reasoning Lovett and Forbus (2017). My works in this chapter and next chapter is an extension to them in the branch of this specific consistency-based analogy-making.

This chapter is organized in the following order:

- in Section 4.2, I present a comparison between two paradigms for solving visual abstract reasoning tasks—one following traditional image classification architectures and the other following the analogy-making process mentioned above;
- in Section 4.3, from the later one, I derive a cognitively-inspired neural architecture for solving visual abstract reasoning tasks;
- in Section 4.4, an specific implementation of this neural architecture is given for the following experiments;
- in Section 4.5, experiments on the RAVEN family are reported, including the RAVEN, I-RAVEN, and RAVEN-FAIR datasets;
- in Section 4.6, the implementation is ablated in two different ways to show its effectiveness;
- in Section 4.7, I report my investigation on an unexplained observation of experiments on the RAVEN family, including both mine and previous works, and disclose the root cause for it;
- in Section 4.8, experiments on the PGM dataset, including both trivial and nontrivial generalization regimes, are discussed;
- in Section 4.9, at last, I present an extended view of the neural architecture as a general approach for neural information processing.

4.2 Two Paradigms of Visual Abstract Reasoning

Again, we use Raven's Progressive Matrices (RPM) as our example task for visual abstract reasoning as in the previous chapters. As a reminder of what RPM looks like, we reproduce a figure of RPM from Chapter 3

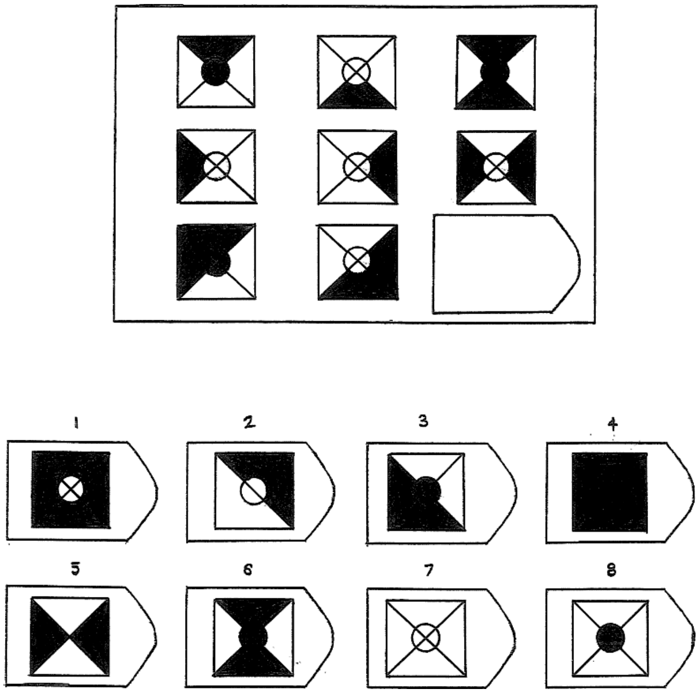


Figure 4.2: An example item of Raven's Progressive Matrices Kunda et al. (2013).

here in Figure 4.2. It consists of a matrix of images with the last entry missing and multiple (usually eight) answer choices. To solve such an item, the human subject needs to select an answer choice to complete the matrix so that the abstract patterns among rows and columns are consistent. For example, the abstract pattern in Figure 4.2 is that taking the union of the first two entries in a row (or a column) leads to the third entry in the row (or column), which leads to the correct answer of the fourth choice.

In the review in Chapter 3, we can see that most learning models for solving RPM follow a standard image classification paradigm, as shown in Figure 4.3. Given the images of the matrix entries and answer choices, this paradigm repeatedly applies feature extractions as in the famous AlexNet work Krizhevsky et al. (2012), decreasing spatial size but increasing the depth of the feature maps until a single vector is obtained for each image. These individual vectors are then combined in various ways and a MLP classification head is applied on the combined representation to predict the class label, i.e., the index of the correct answer choice.

It is not wrong to formulate RPM as an image classification task (in a sense, it is a image classification task); but the image classification paradigm in Figure 4.3 might not the most effective to solve RPM, given the complex nature of RPM that requires to reason about abstract relations/concepts and perceptual elements. We propose a more promising alternate paradigm motivated by human visual analogy-making, i.e., reasoning can often be enhanced by interleaving perceptual and conceptual processing, allowing each process to influence the other. Figure 4.4 illustrates this paradigm. Taking the same raw visual stimuli as in the image classification

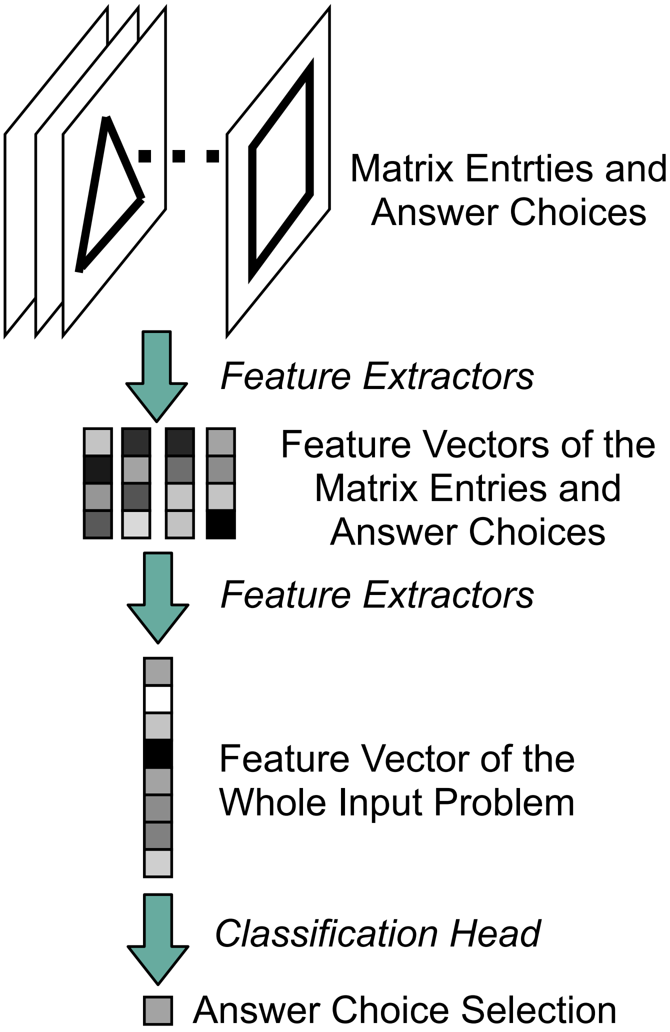

Matrix Entrties and Answer Choices
Feature Extractors
Feature Vectors of the Matrix Entries and Answer Choices
Feature Extractors
Feature Vector of the Whole Input Problem
Classification Head
Answer Choice Selection

Figure 4.3: Image Classification Paradigm for Solving RPM. Note that the sizes and numbers of tensors are diagrammatic, not representing the real implementation.

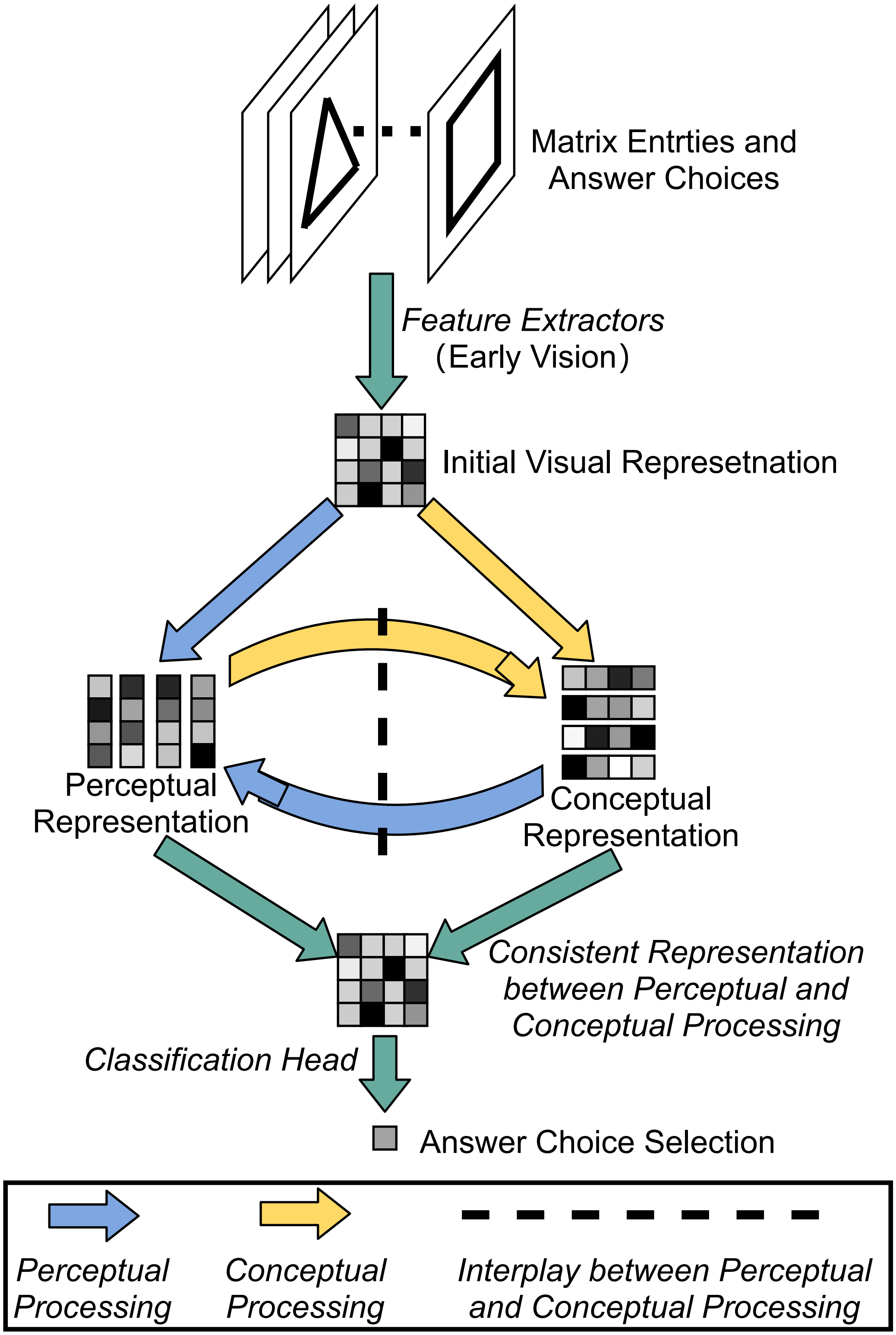


Figure 4.4: A Paradigm Simulating Human Cognition for Solving RPM. Note that the sizes and numbers of tensors are diagrammatic, not representing the real implementation.

paradigm, this alternate paradigm uses feature extractors, simulating early vision processing, to form an initial visual representation of input images. Then there follows two types of processing: (1) perceptual processing that refines the perceptual (visual) representation of input images, for example, refining blurry feature maps of lines and angles to clear feature maps of shapes, and (2) conceptual processing that enriches the representation of abstract concepts, i.e., the relations between entry images.

Then comes the main difference between the image classification paradigm and this paradigm—these two types of processing form a dynamic cycle, in which the perceptual and conceptual processing depend on each other's output. This cycle allows for direct interplay between perceptual and conceptual processing and is repeated for multiple steps until **consistency** between perceptual and conceptual processing is reached (thus adding a requirement for checking or somehow computing the consistency at every step). The resulting consistent representation thus takes into account both the perceptual information in individual images and the conceptual information across the different images. This combined representation can be used to predict the answer in a way that is meaningful to perception and conception both.

Figure 4.4 depicts reasoning on RPM-like problems as a complex, flexible, and dynamic process. This kind of entangled processing is frequently implied in theoretical cognition studies Barsalou et al. (1999); complementarily, behavioral studies Carpenter et al. (1990) involving eye tracking show that human subjects' attention move back and forth between the matrix entries, visiting each entry multiple times, rather than scanning the entries linearly (though other explanations also exist for such gaze phenomena).

As I indicated above, the rationale for the second paradigm is also deeply rooted in human analogy-making. Let us consider the simple visual analogy *"A is to B as C is to D"* in Figure 4.1 from the introduction in more depth. Suppose that a human subject has formed an initial visual representation for each analog in the analogy by looking at the figure for two seconds, but it is probably not the final, correct representation. According to the structure-mapping theory Gentner (1983), the subject needs to construct a mapping F between the base domain (A,B) and the target domain(C,D). This mapping depends on how the analogs are represented. Given the initial visual representations of analogs, the fish and the bird are probably mapped to each other according to their appearance, e.g., *head to head, fins to wings, and tail to tail*, and the air and the water are mapped in a holistic way. Then, if the subject's thinking moves to a higher level and tries to map the relations (i.e., G in Figure 4.1) in (A,B) to the ones in (C,D), she will find that they do not exactly match. In particular, (most) fish use tails for propulsion and fins for direction, whereas birds use wings for propulsion and tails for direction. This observation on G updates the mapping F and the representations of analogs—*fish fins to bird tails, fish tails to bird wings, fish heads to bird heads*, and *air to water holistically*. Given this more meaningful mapping F, if the subject moves up to a higher level again and compare the relations G, the mapping between B and D could be further refined to *air dynamics is mapped to fluid dynamics* (rather than their colors or tastes) and thus the representation of water and air are also updated to focus on their dynamics properties. Given F corresponding to the perceptual processing and G corresponding to the conceptual processing, this iterative process is equivalent to the interplay between perceptual and conceptual processing.

If the subject can give initial representations of analogs that can directly lead to the final correct mappings F and relations G, she may not need to go through this iterative process. However, in real-life situations, where stimuli are complex and ambiguous, the correct representations cannot always be formed immediately. This iterative process of working on F, G, and the representations of analogs is often needed to make and understand analogies.

About the preference for consistency, its rationale naturally follows from the fact that the analogy is fully interpreted or understood only if the iterative process has ended, i.e., no updates need to be made to representations of analogs anymore. In other words, it has been well recognized that analogical proportions enjoy central permutation as a characteristic property Dorolle (1949). That is, *A is to B as C is to D* if and only if *A is to C as B is to D*. This corresponds to interpretations of the analogy in Figure 4.1 in the horizontal or vertical direction. Two directions are equivalent. That one direction holds implies that the other direction also holds. Given this symmetry, G could also be regarded as a mapping between (A,C) and (B,D). If the interpretation of the analogy is unique, i.e., the mappings are unique, we will have $F \circ G = G \circ F$, i.e., F and G are commutative. *This equation is a very concise and beautiful description of analogy-making. By pursuing consistency between perceptual and conceptual processing in our neural architecture, we are actually pursuing equality in this equation, in a supervised data-driven way.* And we will revisit this analogy-making view in a more detailed way later in Chapter 5.

4.3 From the Paradigm to Neural Architecture

While it is not difficult to mechanically design a neural architecture that realizes the human-cognition paradigm in Figure 4.4, the real technical difficulty lies in how to design it so that it can be optimized properly——given its dynamic nature, how can we make sure the network steadily converges and more importantly, converges to the consistency to drive robust reasoning? This section describes a new neutral architecture—Contrastive Perceptual-Conceptual neural architecture (CPCNet)—that implements the human-cognition paradigm for solving visual abstract reasoning tasks and guarantees a stable convergence, and a specific CPCNet model for experimenting on the RPM-like datasets that were reviewed in Chapter 3.

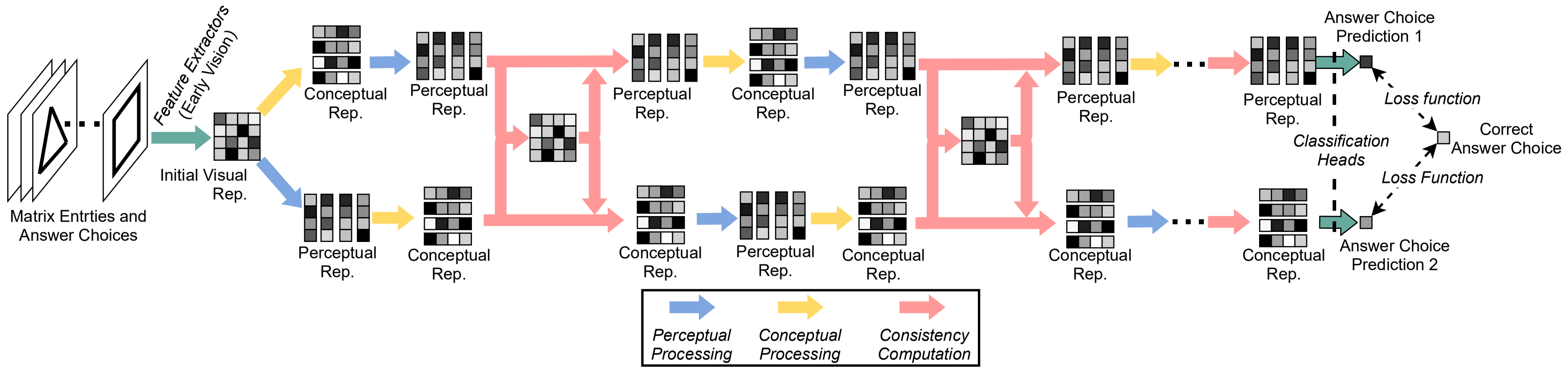


Figure 4.5: CPCNet: A Feed-Forward Architecture that unrolls the paradigm in Figure 4.4.

4.3.1 CPCNet Neural Architecture

In the traditional feed-forward architectures commonly used in deep neural nets, we do not have the convergence issue. I thus try to approximate the human-cognition paradigm with a feed-forward structure. Note that there are two *basic paths* (or, more precisely, cycles) in Figure 4.4 that give the human-cognition paradigm its dynamic nature: (1) *Path 1* starting from the perceptual representation through the conceptual representation and returning back to the perceptual representation, and similarly, (2) *Path 2* starting from the conceptual representation through the perceptual representation and returning back to the conceptual representation. Therefore, if we unroll the cycles with these two paths and add the consistency computation mentioned above, we will have a feed-forward architecture, as shown in Figure 4.5. It **approximates** the fully iterative, cognitively-inspired paradigm in a feed-forward manner. This solves the convergence issue.

Next, as indicated in the introduction section, the human cognition paradigm pursues consistency between perceptual and conceptual representations. There are two designs in the feed-forward architecture to make this happen.

First, after each iteration of *Path 1* and *Path 2*, the red paths in Figure 4.5 first compute the consistency information between the perceptual and conceptual representations. This consistency information could be computed as a shared component between the perceptual and conceptual representations through a linear mapping, or more complex non-linear operations could be used to have more representational power. Either way, the consistency information is then used to update the perceptual and conceptual representations, for example, deducting the consistency information from the perceptual and conceptual representations. This way, it would become easier and easier for later iterations to reach a "full" consistency because, intuitively, the job of eliminating inconsistency gets amortized over multiple iterations and the inconsistency should always be eliminated completely given sufficiently many iterations and proper supervising signals.

Second, the above computation structure only makes the consistency more likely to happen. But it does not guarantee that. Thus, we designed two classification heads at the end of the architecture, which classify the perceptual and conceptual representations, respectively. Then, during training, a loss function is used to pull predictions of the perceptual and conceptual representations from the two paths toward the same answer label. The supervising signal of classifying the same label will propagate back through the classification heads and pull the perceptual and conceptual representations from the two paths toward a consistent position. Here, the meaning of "consistent" becomes more clear—consistent representations could be mapped to the same correct answer label through some simple mappings, like a MLP. The design here is very similar to the idea of supervised contrastive learning Khosla et al. (2020), but it does not require data augmentation or pre-training as we commonly did for supervised contrastive learning. Instead, to achieve similar contrastive effect, our design relies on the delicate architecture design, which is inspired by the human cognition on analogy-making and VAR.

4.3.2 CPCNet Model for Solving RPM-Like Datasets

Based on the above discussion above, we can now describe a specific CPCNet model for solving RPM-like datasets in a formal way. In particular, we first describe what type of operations are used at each step. Then the coding-level implementation, hyper-parameters, and training procedures are provided in the next section.

At first, we use the *single-choice evaluation protocol*, which is more challenging than the commonly-used *multi-choice evaluation protocol*, because comparing answer choices gives the model advantage over evaluating each single answer choice independently (see more about this in Benny et al., 2021). By inserting each answer choice into the matrix and evaluating them individually, we can turn a multi-choice item into multiple binary classification items, where the input to our model is a real tensor x of shape $R \times C \times H_{input} \times W_{input} \times K_{input}$ (R and C are the rows and columns of the matrix. H_{input} and W_{input} are the height and width of each entry images. K_{input} is the image channel that equals 1 for grayscale images in our datasets) and a binary class label y indicates whether the inserted answer choice is correct.

For the feature extractor that simulates early vision in Figure 4.5, we adopt a convolution-based encoder f_E. Since the early vision usually does not involve forming abstract relations between entry images, f_E is applied on each matrix entry $x_{r,c}$ individually:

$$z_{r,c} = f_E(x_{r,c}) \forall (r,c) \in \{1,...,R\} \times \{1,...,C\} \tag{4.1}$$

$$z = [z_{r,c}]_{r=1,...,R,c=1,...,C} \in \mathbb{R}^{R \times C \times H \times W \times K} \tag{4.2}$$

where $H < H_{input}$, $W < W_{input}$, and $K > K_{input}$. Let $z_1^{(0)} = z_2^{(0)} = z$ for the Path 1 and 2 mentioned above, respectively.

For each iteration $i \in \{1,...,L\}$ after f_E, we need to define Path 1, Path 2, and the consistency computation between them. For Path 1, we define perceptual and conceptual processing as convolution-based modules $h_1^{(i)}$ and $g_1^{(i)}$, respectively. Similarly, for Path 2, we define the perceptual and conceptual processing as convolution-based modules $h_2^{(i)}$ and $g_2^{(i)}$. For the consistency computation, we define a MLP $q^{(i)}$. The hyper-parameters of these modules are set to values that preserve the input tensor shape (R,C,H,W,K), i.e., the output channels of $h_1^{(i)}$, $g_1^{(i)}$, $h_2^{(i)}$, and $g_2^{(i)}$ and the output units of $q^{(i)}$ are all set to K.

For RPM tasks, the abstract concepts lie in the row and column dimensions because the abstract concepts are represented by rows and columns by design. We thus apply the convolutions of conceptual processing $g_1^{(i)}$ and $g_2^{(i)}$ on the (R,C) dimensions of the input tensor, and apply the convolutions of perceptual processing

$h_1^{(i)}$ and $h_2^{(i)}$ on the (H,W) dimensions. And the consistency computation $q^{(i)}$ is applied on the channel dimension (K) to keep the spatial correspondence between the two paths. Note that dimensions when not being computed are considered transparent, i.e., like extended batch dimensions. Let the intermediate outputs from Path 1 and 2 of Iteration $i-1$ be $z_1^{(i-1)}$ and $z_2^{(i-1)}$, the computation of Iteration i is:

$$u_1 = h_1^{(i)} \circ g_1^{(i)} (z_1^{(i-1)}) \tag{4.3}$$

$$u_2 = g_2^{(i)} \circ h_2^{(i)} (z_2^{(i-1)}) \tag{4.4}$$

$$v_1 = q^{(i)}(u_1) \tag{4.5}$$

$$v_2 = q^{(i)}(u_2) \tag{4.6}$$

$$z_1^{(i)} = u_1 - v_2 \tag{4.7}$$

$$z_2^{(i)} = u_2 - v_1 \tag{4.8}$$

For the final output, we define two classification heads p_1 and p_2 for Path 1 and Path 2, respectively:

$$\hat{y}_1 = p_1(flatten(mean(z_1^{(L)}))) \tag{4.9}$$

$$\hat{y}_2 = p_2(flatten(mean(z_2^{(L)}))) \tag{4.10}$$

where the $mean$ is taken over the channel dimension K and the $flatten$ flattens its input to a vector of length $R \times C \times H \times W$. For training, we compute binary cross entropy losses for both $\hat{y}_1$ and $\hat{y}_2$ with respect to y and add them up as the final loss. For testing, we simply add up the logits of the two paths as scores for the input matrices and select the highest score (and the corresponding answer choice) as our answer.

4.4 Implementation of CPCNet and Training

This section describes a standard implementation of CPCNet that is used for the experiments in Section 4.5 and 4.8, whereas in other sections in this chapter, modifications to the standard implementation are needed for experimental purposes. I used Tensorflow to implement the model. In particular, all the tensor data used the data format of “channel_last" to boost running speed. For parameters not specified in this paper, they all took default values in Tensorflow 2.12.0. The K and L of CPCNet were set to 64 and 5, respectively. In the following, the coding-level implementation of CPCNet is given corresponding to Equation (4.1) through Equation (4.10).

F_E is a convolution-based entry encoder as shown in Table 4.1.

For each $i \in \{1,2,3,4,5\}$, $h_1^{(i)}, g_1^{(i)}, h_2^{(i)}$ and, $g_2^{(i)}$ follow the same residual structure as shown in Table 4.2. Since the tensor shape is not changed in the module, the residual link is implemented by directly adding the

Module Name	Specification
Conv2D	channels = 32, kernel = 7, stride = 2, padding = “same”, use_bias = False
BatchNormalization	axis = -1, momentum = 0.9 epsilon = 1e-5
Relu	-
MaxPool2D	pool_size = 3, stride = 2, padding = “same”
Conv2D	channels = 64, kernel = 3, stride = 1, padding = “same”, use_bias = False
BatchNormalization	axis = -1, momentum = 0.9 epsilon = 1e-5
Relu	-
MaxPool2D	pool_size = 3, stride = 2, padding = “same”

Table 4.1: The implementation of the entry encoder F_E.

input to the intermediate tensor before the last ReLU.

Module Name	Specification
Conv2D	channels = 64, kernel = 3, stride = 1, padding = “same”, use_bias = False
BatchNormalization	axis = -1, momentum = 0.9 epsilon = 1e-5
Relu	-
Conv2D	channels = 64, kernel = 3, stride = 1, padding = “same”, use_bias = False
BatchNormalization	axis = -1, momentum = 0.9 epsilon = 1e-5
Relu	-

Table 4.2: The implementation of the perceptual and conceptual processing $h_1^{(i)}$,$g_1^{(i)}$,$h_2^{(i)}$ and, $g_2^{(i)}$.

For each $i \in \{1,2,3,4,5\}$, $q^{(i)}$ is a two-layer MLP as shown in Table 4.3.

Module Name	Specification
Dense	units = 64, activation = “relu", use_bias = True
Dense	units = 64, activation = None, use_bias = True

Table 4.3: The implementation of the consistency computation $q^{(i)}$.

For each $j \in \{1,2\}$, p_j is a two-layer MLP as shown in Table 4.4.

Module Name	Specification
Dense	units = 128, activation = “relu", use_bias = True
Dense	units = 1, activation = None, use_bias = True

Table 4.4: The implementation of the classification head p_j.

For the experiments to run fast, extra configuration was made for training:

- I observed that setting the random seed for tensorflow would reduce the training speed in our experiments. Thus, I did not set random seeds. Nonetheless, according to my experience with the model and datasets, the experimental results can be easily reproduced with default initializers of tensorflow and the hyper-parameters in the following.

- Another key point is that we resized the matrix entry images from 160x160 (i.e., the original image size of the datasets) to 80x80 to save some computation.

These two choices make it possible to run the experiments on a single consumer-level GPU and each trial took only days rather than weeks.

Other hyper-parameters include:

- Batch size of 32. It is the largest batch size that my current GPU can handle. But it should be increased if larger GPU RAM is available.
- Adam Optimizer with a starting learning rate of 0.0025, which is gradually increased to 0.2∼0.4 and finally decreased to 0.0025. The purpose of doing so it to skip the local optima. Other advanced learning rate schedulers or optimizers can be used for the same purpose, but it is beyond the scope of this work.

For all the experiments, I monitored the validation accuracy. When it plateaued for several epochs, we took the checkpoints of the best validation accuracies for testing and reported the best test accuracy.

4.5 Experiments of CPCNet on the RAVEN Family

The CPCNet model is designed to avoid using meta-targets for auxiliary training because this extra hand-crafted supervising signal is not always available for human RPM tests and general VAR tasks. Instead, only the final score of each answer choice is predicted individually and the highest score is selected as the answer, i.e., using the single-choice evaluation protocol, which is more difficult than the alternative—the multi-choice evaluation protocol Benny et al. (2021).

Table 4.5 shows that our model achieve the best average accuracy compared to previous models and the best configuration accuracies on 6 out of 7 configurations. Although the accuracies of the runners-up—Rel-AIR and SAVIR-T—are close to CPCNet's, our model solves the datasets in a more difficult and general way—SAVIR-T uses the easier multi-choice evaluation protocol and is designed to utilize the inductive bias that is specific to the datasets Sahu et al. (2023), and while Rel-AIR uses the harder single-choice evaluation, Rel-AIR employs a separately-trained entry-encoder to explicitly extract values of size and position attributes, which are also specific to the datasets. All the compared models in the table (more or less) follow the image classification paradigm in Figure 4.3, whereas our model is the only one following the cognitively-inspired paradigm in Figure 4.4. The main difference is that our model simulates an iterative interplay between perceptual and conceptual processing, whereas the compared models work in a monotonic way from perceptual processing to conceptual processing, where their conceptual processing is usually based on inductive bias that is specific to RPM. In fact, many of the models in Table 4.5 more or less used inductive bias that is specific

	Model	Avg. Acc.	Center	2x2Grid	3x3Grid	L-R	U-D	O-IC	O-IG
Multi-Choice Evaluation Protocol	LEN	78.3%	82.3%	58.5%	64.3%	87.0%	85.5%	88.9%	81.9%
	MXGNet	83.91%	-	-	-	-	-	-	-
	CoPINet	91.42%	95.05%	77.45%	78.85%	99.10%	99.65%	98.50%	91.35%
	DCNet	93.58%	97.80%	81.70%	86.65%	99.75%	99.75%	98.95%	91.45%
	SAVIR-T	94.0%	97.8%	94.7%	83.8%	97.8%	98.2%	97.6%	88.0%
Single-Choice Evaluation Protocol	WReN	14.69%	13.09%	28.62%	28.27%	7.49%	6.34%	8.38%	10.56%
	ARNe	19.67%	-	-	-	-	-	-	-
	NCD	39.66%	45.45%	35.50%	39.50%	34.85%	33.40%	40.25%	30.00%
	PrAE	65.03%	76.50%	78.60%	28.55%	90.05%	90.85%	48.05%	42.60%
	ALANS	74.4%	69.1%	80.2%	75.0%	72.2%	73.3%	76.3%	74.9%
	MRNet	84.0%	-	-	-	-	-	-	-
	NVSA	87.7%	99.7%	93.5%	57.1%	99.8%	99.1%	98.1%	65.4%
	SCL	91.6%	98.1%	91.0%	82.5%	96.8%	96.5%	96.0%	80.1%
	AlgebraicMR	92.9%	98.8%	91.9%	**93.1%**	99.2%	99.1%	98.2%	70.1%
	Rel-AIR	94.1%	99.0%	92.4%	87.1%	98.7%	97.9%	98.0%	85.3%
	CPCNet	**96.92%**	**100.0%**	**96.70%**	86.05%	**100.0%**	**99.90%**	**99.90%**	**95.90%**
	Human	84.4	95.5%	81.8%	79.6%	86.4%	81.8%	86.4%	81.8%

Table 4.5: Accuracies on the original RAVEN. We report without-auxiliary-training accuracies if possible. Data source for each row: (Zheng et al., 2019), (Wang et al., 2020), (Zhang et al., 2019b), (Zhuo and Kankanhalli, 2021), (Sahu et al., 2023), (Zhang et al., 2019a), (Hahne et al., 2019), (Zhuo et al., 2021), (Zhang et al., 2021), (Zhang et al., 2022), (Benny et al., 2021), (Hersche et al., 2023), (Wu et al., 2021), (Xu et al., 2023), and (Spratley et al., 2020).

to RAVEN either in the model design or in the training procedure. On the contrary, the inductive bias in our model—if we consider it as a kind of inductive bias—is the interplay and consistency between perceptual and conceptual processing, which is more meaningful for solving and understanding general VAR. In particular, CoPINet and DCNet, which have been reported to utilize the backdoor of RAVEN Hu et al. (2021), also achieved lower accuracies than ours.

	Model	Avg. Acc.	Center	2x2Grid	3x3Grid	L-R	U-D	O-IC	O-IG
Multi-Choice Evaluation Protocol	SRAN	60.8%	78.2%	50.1%	42.4%	70.1%	70.3%	68.2%	46.3%
	SAVIR-T	98.1%	99.5%	**98.1%**	93.8%	99.6%	99.1%	99.5%	97.2%
Single-Choice Evaluation Protocol	NCD	48.22%	60.00%	31.20%	29.95%	58.90%	57.15%	62.35%	39.00%
	PrAE	77.02%	90.45%	85.35%	45.60%	96.25%	97.35%	63.45%	60.70%
	ALANS	78.5%	72.3%	79.5%	72.9%	79.2%	79.6%	85.9%	79.9%
	NVSA	88.1%	99.8%	96.2&	54.3&	100.0%	99.9%	99.6%	67.1%
	AlgebraMR	93.2%	99.5%	89.6%	89.7%	99.7%	99.5%	99.6%	74.7%
	SCL	95.0%	99.0%	96.2%	89.5%	97.9%	97.1%	97.6%	87.7%
	CPCNet	**98.5%**	**100.0%**	98.00%	**93.95%**	**100.0%**	**100.0%**	**100.0%**	**97.55%**

Table 4.6: Accuracies on the I-RAVEN. We report without-auxiliary-training accuracies of each model if possible. Data source for each row: (Hu et al., 2021), (Sahu et al., 2023), (Zhuo et al., 2021), (Zhang et al., 2021), (Zhang et al., 2022), (Hersche et al., 2023), (Xu et al., 2023), (Wu et al., 2021).

I also tested the CPCNet model on the other two variants of RAVEN—I-RAVEN and RAVEN-FAIR. Table 4.7 and Table 4.7 show the results on I-RAVEN and RAVEN-FAIR, respectively. Without exception,

	Model	Avg. Acc.	Center	2x2Grid	3x3Grid	L-R	U-D	O-IC	O-IG
Multi-Choice Evaluation Protocol	SAVIR-T	97.4%	-	-	-	-	-	-	-
Single-Choice Evaluation Protocol	MRNet	86.8%	97.0%	72.7%	69.5%	98.7%	98.9%	97.6%	73.3%
	CPCNet	**98.14%**	**100.0%**	**98.15%**	**90.80%**	**100.0%**	**100.0%**	**100.0%**	**98.00%**

Table 4.7: Accuracies on RAVEN-FAIR. We report without-auxiliary-training accuracies of each model if possible. Data source for each row: (Sahu et al., 2023), (Benny et al., 2021).

CPCNet achieved the best average accuracy and the best configuration-wise accuracies on almost all configurations.

Before,moving on to the next set of experiments, an important observation can be made by comparing the results across Table 4.5, Table 4.6, and Table 4.7. Although not discussed very often in literature, it has been shown by multiple works (Wei et al., 2023; Sahu et al., 2023; Wu et al., 2021; Benny et al., 2021; Xu et al., 2023) that when using single-choice evaluation, i.e., not allowing the model to comparing answer choices before scoring them and thus not allowing it to use the backdoor of the original RAVEN, the original RAVEN is more challenging than I-RAVEN and RAVEN-FAIR, that is, the same model always achieves a higher accuracy on I-RAVEN and RAVEN-FAIR than on RAVEN. This point is also reflect in the results of Table 4.5, Table 4.6, and Table 4.7.

This makes sense because the way the answer choices of RAVEN were generated makes the answer choices more similar to each other than in I-RAVEN and RAVEN-FAIR and thus more confusing to the model when evaluated individually; on the contrary, the ways answer choices were generated in I-RAVEN and RAVEN-FAIR make the distractors differ from the correct answer by more attributes and thus less confusing to the model when evaluated individually. Therefore, due to page limit, we report only the experiment on RAVEN here; experiments on I-RAVEN and RAVEN-FAIR can be found in the supplementary material.

4.6 Ablation Studies of CPCNet on RAVEN

4.6.1 Varying the number of CPC layers

Model	Avg. Acc.	Center	2x2Grid	3x3Grid	L-R	U-D	O-IC	O-IG
CPCNet (L=0)	12.99%	11.35%	19.95%	24.75%	5.85%	5.05%	9.30%	14.70%
CPCNet (L=1)	77.32%	87.05%	56.95%	57.20%	94.50%	95.35%	93.55%	56.65%
CPCNet (L=2)	92.68%	100.0%	86.50%	79.05%	100.0%	99.85%	99.90%	83.45%
CPCNet (L=3)	94.22%	100.0%	88.30%	79.55%	100.0%	99.90%	99.90%	91.90%
CPCNet (L=4)	95.55%	100.0%	95.35%	85.45%	99.95%	99.90%	99.85%	88.35%
CPCNet (L=5)	96.92%	100.0%	96.70%	86.05%	100.0%	99.90%	99.90%	95.90%

Table 4.8: Ablation Accuracies of CPCNet on RAVEN by Varying L.

Given the relatively complex structure of our model, we ablate the model in different ways. Since the iterations of Conceptual-Perceptual-Contrasting—CPC layers— are the basic building blocks of our model, we first ablate the model in terms of the number L of CPC layers. We vary L from 0 to 5, where 0 means no CPC layers at all and 5 is the value we used in the above experiments. Table 4.8 shows the result of this ablation. As L increases from 0 to 5, the accuracy increases from 12.99% to 96.92% (random guess baseline 12.5%). This shows that stacking CPC layers significantly contributes to CPCNet's accuracy and that setting L greater than 5 could potentially further increase the accuracy. Also, the accuracy gradually plateaus after $L \geq 3$. Thus, a better trade-off between accuracy and computation could be reached for L smaller than 5. Our initial choice of $L=5$ is purely empirical.

4.6.2 Ablating the consistency computation

Model	Avg. Acc.	Center	2x2Grid	3x3Grid	L-R	U-D	O-IC	O-IG
CPCNet-UP	93.05%	99.75%	87.95%	80.15%	100.0%	99.80%	99.80%	83.90%
CPCNet-LP	91.01%	99.90%	79.65%	75.45%	99.95%	99.85%	99.80%	82.45%
CPCNet-IC	94.19%	100.0%	90.45%	81.65%	99.95%	99.90%	99.95%	87.40%
CPCNet-UC	96.69%	100.0%	95.90%	89.15%	100.0%	99.80%	99.90%	92.05%
CPCNet-LC	95.11%	100.0%	93.85%	84.95%	100.0%	99.85%	99.95%	87.15%
CPCNet (L=5)	96.92%	100.0%	96.70%	86.05%	100.0%	99.90%	99.90%	95.90%

Table 4.9: Ablation Accuracies of CPCNet on RAVEN by Weakening Consistency in Different Ways.

Since the consistency between perceptual and conceptual processing is the core idea of our model, we ablate it in terms of designs for forcing the consistency. Taking the best performing model in the first ablation study—CPCNet with L=5—as the base model, we ablate it in five different ways:

- Removing the whole upper path in Figure 4.5 makes it more difficult for CPCNet to achieve the consistency. We name this kind of ablated CPCNet **CPCNet-UP**. But note that "difficult" does not mean impossible as there are still both perceptual and conceptual processing in the lower path. For completely removing the possibility of achieving the consistency, we would have to remove all CPC layers, i.e., CPCNet with L=0 in Table 4.8 is an example. Similarly, we remove the whole lower path and name the resulted model **CPCNet-LP**.
- Keeping the two paths, we can still weaken the consistency by removing the internal consistency computation (i.e., the red arrows in Figure4.5). We call this resulted ablated model **CPCNet-IC**.
- As mentioned above, the internal consistency computation just makes the consistency more likely to happen but does not guarantee that. We thus designed two classification heads to push the conceptual and perceptual representations toward a consistent position by letting them classifying the same input.

Therefore, we can have two more ablated model—**CPCNet-UC** and **CPCNet-LC**—by removing the upper and lower classification heads, respectively. Note that the input to the classification head is discarded when it is ablated.

Table 4.9 shows the results of ablating the consistency computations. Note that there is no way to remove the consistency computation completely (unless we set $L=0$ as we did for the first ablation study; but if we did so, our model would degrade to a simple ConvNet and it would be meaningless to discuss consistency.). Thus, partially ablating the consistency computation would only partially weaken the contribution of consistency computation. This can be told by the relatively smaller variation in accuracy compared to the first ablation study. However, we can still see that having all the consistency designs leads to the best accuracy.

4.7 The Gap Between Grid and Non-Grid Configurations of RAVEN

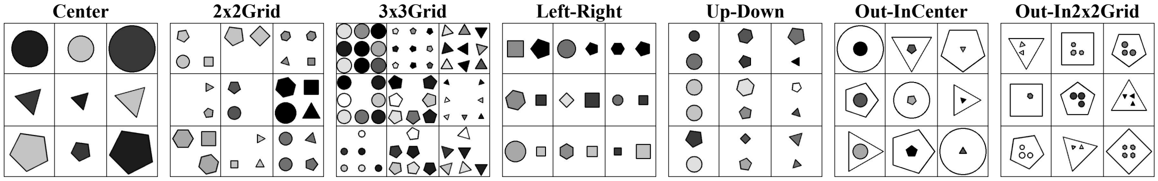


Figure 4.6: 7 spatial configurations of the RAVEN dataset. Each configuration is illustrated by a complete 3x3 matrix. In the **Center** configuration, each entry contains only one object located at the center. In **2x2Grid** and **3x3Grid**, objects can only be located on the grid positions in each entry. In **Left-Right** and **Up-Down**, each entry contains exactly two objects located at the fixed positions shown in the figure. **Out-InCenter** is a combination of two center configurations and **Out-InGrid** is a combination of a center configuration and a 2x2Grid configuration.

In the above experiments on RAVEN, I-RAVEN, and RAVEN-FAIR and almost all previous works on these datasets, a striking observation can be made about the configuration-wise performance. To explain it clearly, I reproduce the 7-configuration figure of the RAVEN family here in Figure 4.6. In particular, in Table 4.5, 4.6, and 4.7, all the models perform significantly worse on the 2x2Grid, 3x3Grid, and Out-InGrid configurations (the second, third, and last in Figure 4.6) than on the other non-grid configurations. It is probably not a coincidence which is resulted from the randomness of these experiments.

As the names of these configurations indicates, they all contain some grid spatial structures, which allow more variations in number and position of geometric elements than the non-grid configurations. Moreover, the grid and non-grid configurations represent two common types of human perceptual organization in VAR tasks—classical view and normal view (Arendasy and Sommer, 2005)—in which the correspondence between geometric elements are established very differently in human cognition. Given the differences in terms of abstract concepts and human perception, it would be necessary to investigate why and how our model and previous models treat the two grid and non-grid configurations differently, why they can't solve them equally well, and how to improve them.

As a heads-up, after I found the root cause of this issue, I realized that it was not complicated or profound at all (as I expected originally). However, the several months' journey for searching for the cause is quite typical for scientific exploration and personally a very interesting one. I thus decided to log it in my dissertation.

4.7.1 What Possibly Causes the Gap

When a neural net does not work well on a specific dataset, the first thought that would occur to the designer of the neural net is often that the model is not good enough. The designer would then keep working on the model to improve it. It is the same with me.

My model and most previous models for solving the RAVEN family are more or less influenced by the classical and popular computer vision models, which are all designed to tackle common computer vision tasks. But the RAVEN family, and more generally, visual abstract reasoning tasks, receive much less attention while those computation vision models were deveoped. Due to the difference between visual abstract reasoning and classical computer vision tasks (with different emphases on perception or on abstract reasoning), the design of classical computer vision model might not be the most suitable for solving visual abstract reasoning tasks.

Therefore, I made a series of changes to the **prototypical CPCNet model**, which is slightly different from the one described above in Section 4.3. Unfortunately, they all (relatively) failed on the grid configurations, i.e., failed to achieve accuracies on grid configuration that are comparable to accuracies on non-grid configurations. Here, I briefly listed the changes I made:

1. I tried to Keeping the position and number information of geometric elements while convoluting on images: the conceptual and perceptual processing of the prototypical CPCNet gradually decreases spatial dimensions and increase the size of channel dimension, through convolutions. However, this encoding process could be lossy for the position and number information of geometric elements in matrix entries. And the variations about position and number only exist in grid configurations. This might be the reason for the gap between grid and non-grid configurations. Therefore, I set the conceptual and perceptual processing in the prototypical CPCNet to be spatial-dimension preserving ones.

2. Changing the 4D convolution in consistency computation to 1D linear mapping: the prototypical CPCNet used very complex 4D convolutions in consistency computation. However, these 4D convolutions (on the row, column, height, and width) would disrupt the information of number and position. For the

same reason as above, I changed it from 4D to 1D linear mapping.

3. Changing the 1D linear mapping in consistency computation to nonlinear 2-layer MLP: the idea of consistency might not be able to be represented linearly. Inspired by the device in transformer, which is good at reasoning the relations between input tokens, 2-layer MLP could be a better choice for consistency computation.
4. Removing consistency computation completely.
5. Increasing or decreasing the size of channel dimension.
6. Increasing or decreasing the depth and width of classification heads.
7. Removing the feature extractor of early vision and using only CPC layers.

Although the changes above did not reduce the gap between grid and non-grid configurations, some of them, e.g., 1, 2, and 3 , did improve the overall accuracy of both grid and non-grid configurations. Thus, they were kept in the later CPCNet design, i.e., the one I described in Section 4.3.

After iterating on the model design in different ways, I started to shift my focus to another aspect of machine learning tasks—training procedure. There are different possible reasons that the training procedure could be a cause. First, it is possible all the previous models were trapped at some local optima where the grid accuracies are lower than the non-grid ones. Second, the gap between the grid and non-grid configuration can also be in interpreted in terms of one of the common phenomena—overfitting. The overfitting interpretation is particularly interesting for the prototypical CPCNet model. During training process, the validation/testing accuracies of non-grid and grid configurations will always respectively plateau at 90% plus and 50% plus, whereas training accuracies of all configurations will plateau at 90% plus, i.e., no clear distinction between grid and non-grid configurations for training. Thus, there is a paradox here:

1. for the non-grid configurations, there is no gap between training and validation/testing accuracies. This shows that the model did learn the underlying abstract concepts and visual elements, rather than the memorizing the training set.
2. for the grid configurations, there is a 40% gap between training and validation/testing accuracies. This means that the model did not learn the underlying abstract concepts and visual elements, but possibly memorizing the training set.

There are at least two hypotheses that we could make to explain the paradox:

1. Hypothesis 1: The model was just memorizing the items in the training set of the grid configurations, and due to the partial similarity between training and validation/testing sets of grid configurations, the prototypical CPCNet achieved the 50% plus validation/testing accuracy on grid configurations. It could be caused by a local optimum.

2. Hypothesis 2: The model was able to learn some abstract concepts and some visual elements in the training set of the grid configuration, but not all of them. In this case, it memorized the ones that it was not able to learn. This could also give the 50% plus validation/testing accuracy on grid configurations. In particular, this could be caused by an optimization difficulty due to the competition between different types of abstract concepts and visual elements. And some abstract concepts and visual elements won out in the competition.

To test Hypothesis 1 and other general overfitting causes (rather than the ones specific to RAVEN), the following experiments were conducted:

1. I equipped the prototypical CPCNet with traditional regularization methods, such as dropout in the MLP modules.

2. In our case of conceptual and perceptual contrasting, extra regularization was also applied to reduce the residual of contrasting.

3. I also resized the input images to smaller size so that larger batch size could be used.

4. Warmup training.

5. Fully shuffling the dataset for every epoch, which was not originaly used for training the prototypical CPCNet to save memory.

6. Trying different loss functions, such as cross entropy, binary cross entropy, binary focal cross entropy, and other customized losses for the prototypical CPCNet.

Unfortunately, none of them worked.

Hypothesis 2 implied that the abstract concepts and visual elements—that are specific to grid configurations, i.e., the variations about position and number of goemetric objects—were the ones that did not won out. But why didn't they win out? In other words, why were they difficult to learn compared to others? Perhaps, before answering these questions, a very basic question was whether the variations about position and number of geometric objects were learnable by a neural network or not. Therefore, I created a new number matrix dataset, which ideally encoded the position and number information of goemetric objects explicitly in numbers. That is, each item is a matrix of numbers instead of images, where numbers correspond to the number and position of geometric objects in the original RAVEN, and there were no other variations in the number matrix dataset. An MLP was trained and tested on this number matrix dataset and the result showed that, when represented in this explicit way, the variation about number and position of goemetric objects could be learned EXTREMELY easily.

Given all the above experiments about model design and training, I started to suspect that the problem arose from the dataset itself, rather than the methods we used to solve the dataset. There might be something special about the RAVEN dataset that caused this special distinction between grid and non-grid configurations. I thus conducted following experiments on the RAVEN dataset:

1. Data leak check. It turns out that there is no data leak from the training set to the validation and testing set.
2. Balanced training in terms of correct and wrong answer choices. There are eight answer choices and only one of them is correct. The negative supervising signal is much stronger than the positive one. I thus reformulated each training items to have only two answer choices—one correct and one wrong. However, this did not improve the situation at all.
3. Noise-Free RAVEN: in grid configuration, there exist noisy visual attributes (e.g., angle and orientation) that are not governed by any abstract concepts/rules. But non-grid configurations do not have such noisy attributes. I thus regenerated the items of grid configurations without any noisy attribute. However, training on this noise-free RAVEN did not reduce the gap between the grid and non-grid configuration at all. Thus, the noise is not the reason for the gap.

So far, all the experiments I tried to explain the gap between the grid and non-grid configurations has failed. However, when I was inspecting the results of these experiments, a glimmering of correlations between these numbers eventually helped me solve this mystery. In particular, the root cause for the gap was verified through creating a new variant of RAVEN, which will be discussed in the next subsection.

4.7.2 AB-RAVEN

The most important feature of RAVEN or other visual abstract reasoning tasks is that each item is characterized by some abstract concepts and visual attributes, and the whole dataset is backed by a system of abstract concepts and visual attributes. That is, the meaning of each item is not just a positive or negative label to be predicted, or belonging to certain perceptual category; instead, the meaning is more conceptually complex. This allows us to analyze the situation for each abstract concept and visual attribute. Table 4.10, 4.11, and

4.12 show the testing accuracy of the prototypical CPCNet for each combination of abstract concept and attribute in the grid configurations.

	Number	Position	Number/Position	Type	Size	Color
Constant	-	-	89.43%	61.99%	58.48%	61.97%
Progression	65.22%	37.34%	-	73.07%	70.96%	70.7%
Arithmetic	63.87%	66.29%	-	-	66.73%	64.83%
Distribute-Three	68.34%	57.74%	-	64.12%	68.35%	68.49%

Table 4.10: Testing accuracy of the prototypical CPCNet for each abstract concept and attribute in the 2x2Grid configuration of RAVEN.

	Number	Position	Number/Position	Type	Size	Color
Constant	-	-	91.67%	61.04%	57.29%	57.8%
Progression	73.67%	23.37%	-	65.38%	68.56%	69.71%
Arithmetic	62.71%	65.82%	-	-	65.14%	65.77%
Distribute-Three	74.25%	59.52%	-	66.15%	65.84%	63.2%

Table 4.11: Testing accuracy of the prototypical CPCNet for each abstract concept and attribute in the 3x3Grid configuration of RAVEN.

	Number	Position	Number/Position	Type	Size	Color
Constant	-	-	75.95%	76.29%	76.3%	75.95%
Progression	85.23%	58.9%	-	77.09%	75.68%	77.78%
Arithmetic	73.94%	72.07%	-	-	80.93%	76.32%
Distribute-Three	78.99%	73.41%	-	75.02%	75.28%	81.14%

Table 4.12: Testing accuracy of the prototypical CPCNet for each abstract concept and attribute in the OutInGrid configuration of RAVEN.

Meanwhile, Table 4.13, 4.14, and 4.15 show the training item number for each abstract concept and attribute in the grid configurations. We can see a moderate correlation between the three accuracy tables and the three training item number tables. This implies that, for each combination of abstract concept and attribute, to some extent, its testing accuracy is determined by the number of its training items. To be comprehensive, Table 4.16, 4.17, 4.18, and 4.19 show the training item number for each abstract concept and attribute in the non-grid configurations

	Number	Position	Number/Position	Type	Size	Color
Constant	0	0	970	1957	1527	1463
Progression	556	501	0	1999	1512	1469
Arithmetic	989	1038	0	0	1503	1507
Distribute-Three	986	960	0	2044	1458	1561
Sum by Color	5030		18970			

Table 4.13: The Training Item Number for Each Abstract rule and Attribute in the 2x2Grid Configuration of RAVEN.

	Number	Position	Number/Position	Type	Size	Color
Constant	0	0	833	1999	1494	1544
Progression	828	849	0	1991	1552	1480
Arithmetic	872	884	0	0	1553	1484
Distribute-Three	879	855	0	2010	1401	1492
Sum by Color	5167		18833			

Table 4.14: The Training Item Number for Each Abstract rule and Attribute in the 3x3Grid Configuration of RAVEN.

	Number	Position	Number/Position	Type	Size	Color
Constant	0	0	6000	3272	3432	6000
Progression	473	459	0	3365	1894	1563
Arithmetic	1011	1011	0	0	1707	1456
Distribute-Three	1060	976	0	3385	3414	1490
Sum by Color	4990		36978			

Table 4.15: The Training Item Number for Each Abstract rule and Attribute in the Out-In2x2Grid Configuration of RAVEN.

	Number	Position	Number/Position	Type	Size	Color
Constant	0	0	6000	1977	1520	1524
Progression	0	0	0	1998	1465	1454
Arithmetic	0	0	0	0	1505	1535
Distribute-Three	0	0	0	2025	1510	1487
Sum by Color	0		24000			

Table 4.16: The Item Number for Each Abstract rule and Attribute in the Center Configuration of RAVEN.

	Number	Position	Number/Position	Type	Size	Color
Constant	0	0	6000	3378	3243	6000
Progression	0	0	0	3344	2412	1511
Arithmetic	0	0	0	0	1460	1536
Distribute-Three	0	0	0	3320	3332	1509
Sum by Color	0		37045			

Table 4.17: The Item Number for Each Abstract rule and Attribute in the Out-InCenter Configuration of RAVEN.

	Number	Position	Number/Position	Type	Size	Color
Constant	0	0	6000	3392	2660	2676
Progression	0	0	0	3327	2623	2617
Arithmetic	0	0	0	0	2605	2563
Distribute-Three	0	0	0	3378	2601	2616
Sum by Color	0		36958			

Table 4.18: The Item Number for Each Abstract rule and Attribute in the Left-Right Configuration of RAVEN.

	Number	Position	Number/Position	Type	Size	Color
Constant	0	0	6000	3365	2575	2610
Progression	0	0	0	3362	2591	2703
Arithmetic	0	0	0	0	2679	2605
Distribute-Three	0	0	0	3285	2639	2612
Sum by Color	0		37026			

Table 4.19: The Item Number for Each Abstract rule and Attribute in the Up-Down Configuration of RAVEN.

To further explain this idea, I aggregated the data from configuration training item number tables into Table 4.20. There are two types of combinations in this table—the red ones that exist only in the three grid configurations and the green ones that exist mainly in the other four non-grid configurations and also exist in roughly 65% of the grid items. As we can see in the table, the sum of the green ones is roughly 15 times the sum of the red ones. Therefore, the red ones are much less represented in RAVEN, in their own configurations and globally. This argument also applies to I-RAVEN and RAVEN-FAIR because they share the same underlying system of abstract rules and attributes with RAVEN.

	Number	Position	Number/Position	Type	Size	Color
Constant	0	0	31803	19240	16451	21817
Progression	1857	1809	0	19386	14049	12797
Arithmetic	872	2933	0	0	13012	12686
Distribute-Three	2925	2791	0	19447	16355	12767
Sum by Color	15187		209810			

Table 4.20: Numbers of RAVEN (I-RAVEN and RAVEN-FAIR) Training items containing each combination of abstract rules and attributes.

Since the less represented ones in RAVEN only exist in the three grid configurations mentioned above, I hypothesize that it is because the reds ones in RAVEN in Table 4.20 are less represented that the accuracy is worse on the grid configurations than on the non-grid ones. **To verify this hypothesis**, I revised RAVEN to make it more **B**alanced in terms of **A**bstract rules and attributes (I thus name this revised version **AB-RAVEN**). Table 4.21 shows the statistics of AB-RAVEN. It was made more balanced by decreasing the number of non-grid training items and increasing the number of grid training items while keeping the overall size of training set unchanged. The validation and test sets remain the same as RAVEN's. If the hypothesis is true, we will observe a smaller gap between grid and non-grid testing accuracies on AB-RAVEN than on original RAVEN. Meanwhile, this dataset can also check if previous models' high accuracies on non-grid configurations are a result of excessively many training items of non-grid configurations.

As we see in Table 4.20, AB-RAVEN is not perfectly balanced, just more balanced than RAVEN. Perfectly balancing the dataset will violate the basic arrangement of 7 configurations, i.e., need to remove all non-grid configuration items, making it not comparable with previous works on RAVEN.

	Number	Position	Number/Position	Type	Size	Color
Constant	0	0	19574	17507	15263	21220
Progression	4058	4040	0	17641	11998	11010
Arithmetic	6329	6307	0	0	11508	10932
Distribute-Three	6335	6421	0	17579	15080	10907
Sum by Color	33490		180219			

Table 4.21: Numbers of AB-RAVEN Training items containing each combination of abstract rules and attributes.

Model	Avg. Acc.	Center	2x2Grid	3x3Grid	L-R	U-D	O-IC	O-IG
CPCNet on AB-RAVEN	98.84	99.75	99.20	94.95	99.70	99.80	99.50	98.95
Compared to CPCNet on RAVEN	+1.92	-0.25	+2.5	+8.9	-0.3	-0.1	-0.4	+3.05

Table 4.22: Accuracies on the AB-RAVEN, using single-choice evaluation protocol. The numbers in the second row reflect the improvement over the orginal imbalanced RAVEN dataset.

Table 4.22 shows the CPCNet model's accuracies on AB-RAVEN. Compared to Table 4.5, the accuracy gap between grid and non-grid configurations has been reduced from 7.07% to 1.98%. **This verifies our hypothesis about the imbalanced issue**. Moreover, this experiment also shows that overfitting can happen at a finer level of abstract concepts and attributes, rather than the level of the entire dataset.

For the readers might be curious about the exact distributions of abstract rules and attributes in RAVEN and AB-RAVEN and how they differ from each other, I provide more detailed statistics of them in Table 4.23 through 4.30. These tables would provide reference for studying the effect of distribution of abstract concepts on learning models.

The AB-RAVEN was generated using the original RAVEN generation code[1]. The number of items of each configuration in AB-RAVEN is given in Table 4.23. AB-RAVEN is available here[2].

		RAVEN (I-RAVEN and RAVEN-FAIR)				AB-RAVEN			
	Configuration	Training	Validation	Test	All Splits	Training	Validation	Test	All Splits
Grid Configurations	2x2Grid	6000	2000	2000	10000	12400	2000	2000	16400
	3x3Grid	6000	2000	2000	10000	12400	2000	2000	16400
	O-IG	6000	2000	2000	10000	12400	2000	2000	16400
Non-Grid Configurations	Center	6000	2000	2000	10000	1200	2000	2000	5200
	L-R	6000	2000	2000	10000	1200	2000	2000	5200
	U-D	6000	2000	2000	10000	1200	2000	2000	5200
	O-IC	6000	2000	2000	10000	1200	2000	2000	5200
Sum		42000	14000	14000	70000	42000	14000	14000	70000

Table 4.23: The number of items of each spatial configuration in RAVEN (I-RAVEN and RAVEN-FAIR) and AB-RAVEN.

WARNING: It needs to be pointed out that since I generated the new items of AB-RAVEN using the original RAVEN source code, AB-RAVEN has the same backdoor solution as the original RAVEN, i.e.,

---

[1] https://github.com/WellyZhang/RAVEN

[2] The link is to be released soon.

	Number	Position	Number/Position	Type	Size	Color
Constant	0	0	1200	399	305	303
Progression	0	0	0	416	277	293
Arithmetic	0	0	0	0	306	295
Distribute-Three	0	0	0	385	312	309
Sum by Color	0		4800			

Table 4.24: The Item Number for Each Abstract rule and Attribute in the Center Configuration of AB-RAVEN.

	Number	Position	Number/Position	Type	Size	Color
Constant	0	0	1277	4109	3193	3065
Progression	1151	1124	0	4165	3147	3072
Arithmetic	2178	2253	0	0	3127	3099
Distribute-Three	2149	2268	0	4126	3033	3164
Sum by Color	11123		38477			

Table 4.25: The Item Number for Each Abstract rule and Attribute in the 2x2Grid Configuration of AB-RAVEN.

	Number	Position	Number/Position	Type	Size	Color
Constant	0	0	1097	4162	3133	3170
Progression	1798	1850	0	4114	3126	3123
Arithmetic	1902	1912	0	0	3212	3071
Distribute-Three	1916	1925	0	4124	2929	3036
Sum by Color	11303		38297			

Table 4.26: The Item Number for Each Abstract rule and Attribute in the 3x3Grid Configuration of AB-RAVEN.

	Number	Position	Number/Position	Type	Size	Color
Constant	0	0	1200	650	653	1200
Progression	0	0	0	697	495	295
Arithmetic	0	0	0	0	325	315
Distribute-Three	0	0	0	669	640	313
Sum by Color	0		7452			

Table 4.27: The Item Number for Each Abstract rule and Attribute in the Out-InCenter Configuration of AB-RAVEN.

	Number	Position	Number/Position	Type	Size	Color
Constant	0	0	12400	6818	7040	12400
Progression	1109	1066	0	6929	3920	3190
Arithmetic	2249	2142	0	0	3518	3084
Distribute-Three	2270	2228	0	6950	7082	3050
Sum by Color	11064		76381			

Table 4.28: The Item Number for Each Abstract rule and Attribute in the Out-In2x2Grid Configuration of AB-RAVEN.

	Number	Position	Number/Position	Type	Size	Color
Constant	0	0	1200	671	554	538
Progression	0	0	0	660	522	519
Arithmetic	0	0	0	0	495	537
Distribute-Three	0	0	0	671	540	515
Sum by Color	0		7422			

Table 4.29: The Item Number for Each Abstract rule and Attribute in the Left-Right Configuration of AB-RAVEN.

	Number	Position	Number/Position	Type	Size	Color
Constant	0	0	1200	698	485	544
Progression	0	0	0	690	511	518
Arithmetic	0	0	0	0	525	531
Distribute-Three	0	0	0	654	544	520
Sum by Color	0		7390			

Table 4.30: The Item Number for Each Abstract rule and Attribute in the Up-Down Configuration of AB-RAVEN.

comparing (embeddings of) answer choices before scoring them would lead to a high accuracy. However, as discussed in the experiment section, this so-called backdoor is a double-edged sword. One one hand, it gives a shortcut (i.e., using multi-choice evaluation protocol) to achieve a high accuracy, which, however, does not really say anything about the abstract reasoning ability of the model being tested; on the other hand, combining the single-choice evaluation protocol with AB-RAVEN and RAVEN provides a more challenging and reasonable test to evaluate our models than using multi-choice evaluation protocol and/or I-RAVEN and/or RAVEN-FAIR. Thus, I encourage our readers to use the positive edge of this sword to evaluate the models and restrain the idea of using the negative edge to just achieve high accuracies.

4.8 Experiments of CPCNet on PGM

With exactly the same implementation as in RAVEN experiments, the CPCNet model was tested on the other major matrix reasoning dataset—PGM (Barrett et al., 2018). PGM is very different from the RAVEN family in that, one one hand, PGM is much larger (about 20 times larger) than RAVEN, on the other, PGM provides multiple generalization regimes to evaluate learning models.

As discussed in the Chapter 3, because the visual abstract reasoning datasets are built upon systems of abstract concepts and visual elements, the datasets can be presented to learning models in either trivial or nontrivial manner, depending on the how the abstract concepts and visual elements in training differ from the ones in testing. In particular, in trivial generalization regimes, the combinations of abstract concepts and visual elements are the same across training and testing; on the contrary, nontrivial generalization regimes mean that the combinations of abstract concepts and visual elements in training are different from the ones in testing. There are different ways to combine abstract concepts and visual elements for training and testing, thus leading to different levels of generalization difficulty.

The terms—"trivial" versus "nontrivial"—used here are, in a sense, misleading. In fact, the trivial generalization regimes can be not trivial at all in terms of the difficulty levels as learning tasks. And in some evaluation settings, the trivial generalization regimes can be quite challenging. Similarly, the nontrivial generalization regimes can be quite trivial, depending on how the learning task is formulated in terms of input, output, prior knowledge, and heuristics (for example, see the auxiliary training in Chapter 3). Thus, "trivial" and "nontrivial" are used only to describe the difference between the two situations in terms of the abstract concepts and visual elements. From the perspective of human cognition, these two terms indicate whether extra human-like high-level reasoning steps are needed to make decisions.

One abstract concept can be applied to many different visual elements and one visual element can be used to represent many different abstract concepts. Each visual abstract reasoning item is characterized by one or more pairs of abstract concepts and visual elements, rather than just abstract concepts or just visual elements separately. The existence of abstract concepts are always accompanied by the existence of visual elements. In the case of PGM, each item is defined by one or more triplets of $[r,o,a]$ where r is a relation, o is a object, and a is an attribute, and that a triplet $[r,o,a]$ means the relation r holds between the values of attribute a of the objects o in the matrix entries. The detailed description of r, o, and a can be found in Section 3.3.2. Here, the abstract concepts of PGM are represented by r. Thus, abstract concepts, relations, and rules are used interchangeably in the following. The visual elements of PGM are a bit complex and represented by o and a together. By varying the triplets $[r,o,a]$ between training/validation and testing sets, PGM provides eight different generalization regimes:

1. Neutral: training/validation and testing sets share the same set of 29 triplets $[r,o,a]$. The values of each attribute a are taken from a finite list V_a.

2. Interpolation: training/validation and testing sets share the same set of 29 triplets $[r,o,a]$. In the training/validation set, the values of a are restricted to even-indexed members of the finite list V_a, whereas in the test set only odd-indexed values were permitted.

3. Extrapolation: training/validation and testing sets share the same set of 29 triplets $[r,o,a]$. The values of a were restricted to the lower half of the finite list V_a for training/validation, whereas in the test set they took values in the upper half of V_a.

4. Held-Out Pairs of Triplets: there exist 400 combinations of two different triplets $([r_1,o_1,a_1],[r_2,o_2,a_2])$.

Note that it is not $\binom{29}{2}$, because some triplets are conflicting with each other. 40 out of the 400 pairs are held out for testing.

5. Held-Out Pairs of Attributes: 4 out of the 20 attribute pairs (a_1,a_2) are held out for testing.

6. Held-Out Triplets: 7 out of the 29 triplets $[r,o,a]$ are hold out for testing. In particular, there are 7 $[o,a]$ in total, and for each of the 7 $[o,a]$, a randomly selected $[r,o,a]$ is held out for testing.

7. Held-Out Shape-Color: the triplets with $[o{=}shape,a{=}color]$ are held out for testing. Each testing item contains at least one triplet with $o{=}shape$ and $a{=}color$.

8. Held-out Line-Type: the triplets with $[o{=}line,a{=}type]$ are held out for testing. Each testing item contains at least one triplet with $o{=}line$ and $a{=}type$.

According to our definitions of trivial generalization, only Neutral and Interpolation can be definitely classified as trivial generalization regimes. The Extrapolation regime lies on the boundary between being trivial and nontrivial because, on one hand, the training and testing sets share the same set of triplets, and, on the other, a special form of generalization is needed for handling unseen numerical values. But, unfortunately, this kind of generalization is not encoded in the triplets and PGM thus does not allow us to control and study it systematically,for example, whether learned extrapolation for one attribute help extrapolate for other attributes or whether extrapolations learned through some abstract concepts help extrapolate for other abstract concepts. When it is not studied as a learning task, i.e., learning for extrapolation of numerical values, this kind of generalization can be achieved easily through special design or tricks (Webb et al., 2020a), which does not really help us understand how extrapolation is realized generally in human cognition and AI.

According to our definition of nontrivial generalization, the last five generalization regimes can be classified as nontrivial generalization regimes. Depending on the extent to which the training triplets differ from the testing triplets, these five generalization regimes have different levels of difficulty; the more the training and testing triplets differ, the more difficult the generalization is. The difficulty levels of these five generalization regimes roughly follow this order: Held-Out Pairs of Triplets $<$ Held-Out Pairs of Attributes $<$ or $\approx$ Held-Out Triplets $<$ Held-Out Shape-Color $\approx$ Held-out Line-Type.

The basic nontrivial generalization task is Held-Out Pairs of Triplets, in which the overall triplets actually remain the same across training and testing but the combinations of triplets are different. Thus, success of a model on this regime implies that the model indeed learns the triplets $[r,o,a]$, i.e., the model indeed "understands" that the variation of attribute a of object o follows the rule r, no matter how many different variations happen at the same time in the input.

The Hold-Out Pairs of Attributes moves the difficulty to a higher level by changing the combination of attributes, but not really changes the triplets across training and testing. Success of a model on this regime implies that the model not just have learned the triplets $[r,o,a]$, but also have started to look into the triplet by separating the component a from the triplet.

Different from the previous regimes, the Hold-Out Triplets changes triplets from training and testing by holding out some combinations of abstract concepts and visual elements for testing. In particular, the following seven triplets are held out:

1. $[r=AND,o=line,a=type]$
2. $[r=progression,o=line,a=color]$
3. $[r=progression,o=shape,a=color]$
4. $[r=XOR,o=shape,a=position]$
5. $[r=XOR,o=shape,a=type]$
6. $[r=OR,o=shape,a=size]$
7. $[r=consistent_{u}nion,o=shape,a=number]$

Success on this generalization regime implies that the model has learned that the variations are governed by two components—abstract concepts and visual elements and been able to apply abstract concepts on visual elements in new ways and interpret visual elements with abstract concepts in new ways. This indicates that the abstract concepts and visual elements are represented by the model in some disentangled way that allows flexible combinations of learned representations.

The next two regimes—Hold-Out shape-color and Hold-Out line-type— are probably the most challenging nontrivial generalization regimes. They require to generalize to visual elements—shape-color and line-type respectively—that are irrelevant during training. Generally, this kind of generalization is impossible for current learning model because current learning models, as we all know, are always "lazy" in their learning. If they can find a short-cut solution, they will never to make an extra effort to try to "understand" stimuli that are irrelevant to the current task. Instead, they would simply consider irrelevant stimuli as random noise and just ignore them. However, the situation for Hold-Out shape-color and Hold-Out line-type are subtly different from generalizing to completely irrelevant visual elements. In appearance, the training on $[r,o=\text{shape},a=\text{not color}]$ and $[r,o=\text{not shape},a=\text{color}]$ would possibly help generalize to $[r,o=\text{shape},a=\text{color}]$. Similarly, the training on $[r,o=\text{line},a=\text{not type}]$ and $[r,o=\text{not line},a=\text{type}]$ would possibly help generalize to [*r*,*o*=line,*a*=type]. However, in our experience with this dataset, this kind of generalization is very questionable because *o* and *a* are too much entangled with each other and probably cannot be separated as too components. That is, *o* and *a* can only be learned together at the same time and treated as a whole. Therefore, the combination game would probably not work here.

Due to the possibly entangled between *o* and *a*, PGM might not be a perfect dataset for studying different nontrivial generalizations that could have been defined more clearly by disentangled conceptual and perceptual components.

4.8.1 Trivial Generalization Regimes

Most previous works using PGM as a benchmark only evaluated their models on the Neutral generalization regimes. Other generalization regimes, whether trivial or nontrivial, received much less attention from the previous research. Table 4.31 shows the models' accuracies on the trivial generalization regimes. Note that here we consider Extrapolation as a trivial one, though it lies on the boundary between trivial and nontrivial ones.

	Model	Neutral	Interpolation	Extrapolation
Multi-Choice Evaluation Protocol	CoPINet	56.37%	-	-
	MXGNet	66.7%	65.4%	18.9%
	LEN	68.1%	-	-
	DCNet	68.57%	59.7%	17.8%
	SRAN	71.3%	-	-
	SAVIR-T	91.2%	-	-
Single-Choice Evaluation Protocol	ARNe	12.55%	-	17.76%
	NCD	47.6%	47.0%	**24.9%**
	WReN	62.6%	64.4%	17.2%
	ViT	72.7%	67.7%	16.4%
	NI	77.0%	70.5%	19.4%
	Rel-Base	85.5%	-	22.05%
	SCL	88.9%	55.8%	17.3%
	MRNet	93.4%	68.1%	19.2%
	RS-TRAN	97.5%	**77.2%**	19.2%
	MLRN	98.03%	57.8%	14.9%
	CPCNet	**98.4%**	74.08%	16.75%

Table 4.31: Accuracies on the Trivial Generalization Regimes of PGM. We report without-auxiliary-training accuracies of each model if possible. Data source for each row: (Zhang et al., 2019b), (Wang et al., 2020), (Zheng et al., 2019), (Zhuo and Kankanhalli, 2021), (Hu et al., 2021), (Sahu et al., 2023), (Hahne et al., 2019), (Zhuo et al., 2021), (Barrett et al., 2018), (Rahaman et al., 2021), (Rahaman et al., 2021), (Spratley et al., 2020), (Wu et al., 2021; Małkiński and Mańdziuk, 2020), (Benny et al., 2021), (Wei et al., 2023), (Jahrens and Martinetz, 2020).

.

Table 4.31 shows that the CPCNet model performs well on the Neutral generalization regimes. However, the accuracies on the Interpolation and Extrapolation regimes are not as good as on the Neutral one. It is understandable that the CPCNet does not work well on Extrapolation because the CPCNet does not have any special design to support this kind of generalization. Like most learning models, it should not be able to generalize to numerical values of attributes that are two far away from the values on which it is trained.

CPCNet's accuracy on Interpolation should have been close to its accuracy on the Neutral. But Table 4.31 shows that the CPCNet's accuracy on Interpolation is significantly lower than its accuracy on the Neutral. The reason for this could be that the CPCNet model was not sufficiently trained. Due to the limited computational resource, systematic exploration for hyper-parameters and sufficiently many random trials could not be conducted for the generalization regimes other than Neutral.

4.8.2 Nontrivial Generalization Regimes

	Model	H.O. Triplet Pairs	H.O. Attribute Pairs	H.O. Triplets	H.O. Line-Type	H.O. Shape-Color
Multi-Choice Evaluation Protocol	MXGNet	43.3%	33.6%	19.9%	16.7%	16.6%
Single-Choice Evaluation Protocol	WReN	41.9%	27.2%	19.0%	14.4%	12.5%
	ViT	44.1%	34.1%	15.9%		
	NI	45.2%	36.6%	20.0%		
	SCL	64.5%	**40.8%**	**27.0%**	15.1%	12.7%
	MRNet	55.3%	38.4%	25.9%	30.1%	**16.9%**
	RS-TRAN	43.6%	28.4%	22.2%	24.7%	12.9%
	CPCNet	**97.14%**	33.00%	24.62%	**30.98%**	13.32%

Table 4.32: Accuracies on the nontrivial Generalization Regimes of PGM. We report without-auxiliary-training accuracies of each model if possible. Data source for each row: (Wang et al., 2020), (Barrett et al., 2018), (Rahaman et al., 2021), (Rahaman et al., 2021), (Wu et al., 2021; Małkiński and Mańdziuk, 2020), (Benny et al., 2021), (Wei et al., 2023).

Table 4.32 show the experimental results of nontrivial generalization regimes. According to the implication of Held-Out Pairs of Triplets discussed above, it can be said that the CPCNet model indeed learned the triplets $[r,o,a]$, i.e., the model indeed "understands" that the variation of attribute a of object o follows the rule r, no matter how different variations, including noisy random variation, happen at the same time in the input; moreover, according to the implication of Held-Out Pairs of Attribute mentioned above, our model made an initial effort to look into the internal structure of triplets and was able to separate the component a from the triplet in some cases, but in most cases, it failed. Given its performance on the last three regimes, the CPCNet model show an initial sign of attempting to discover the internal structure of triplets and to make nontrivial generalizations by re-organizing the components of learned triplets. However, it can be seen that the CPCNet model has plenty of room for improvement when the abstract concepts and visual elements during training and testing are distant from each other. It also needs to be pointed out that, again, due to the limited computational resource, systematic exploration for hyper-parameters and sufficiently many random trials could not be conducted for the nontrivial generalization regimes.

4.9 An Extended View of CPCNet : Deep Non-Monotonic Reasoning

In this section, I will present an slightly different view of CPCNet that extends the original view of perceptual and conceptual contrasting to a more general view. CPCNet can be considered as a special case of this new view, in which the input is assumed to be general multi-dimensional tensor, where each dimension has its own semantics and the whole tensor input presents a complex reasoning problem involving interaction between all the dimensions. This view not only help us understand how CPCNet works on RAVEN an PGM, but also it provides a guidance for extending CPCNet for solving more complex and flexible visual abstract reasoning tasks.

Most existing deep learning models for processing this kind of multi-dimensional tensor input operate in a monotonic way, i.e., aggregating/eliminating each dimension of the input gradually in a fixed order. In contrast, CPCNet represents a non-monotonic way for processing, in which multiple processing orders are explored and visiting the same dimensions for multiple times are allowed.

Again, take RPM items as an example. If we insert each answer choice into the matrix, we have eight 3×3 matrices with each matrix entry a 160×160 image. Since there are a finite number of items in RPM-like datasets, we can assume that we have N underlying patterns and each underlying pattern has M instances (i.e., items) in this dataset. Then, this RPM-like dataset can be represented by a 7D array of shape $(N,M,a=8,r=3,c=3,h=160,w=160)$, with a for answer choices, r for rows, c for columns, h for height of matrix entry images, and w for width of them.

A monotonic AI system would probably do the processing from the right to the left of $(N,M,c=8,r=3,c=3,h=160,w=160)$, without backtracking. The monotonic assumption here means that correctly deciphering a dimension is conditioned on correctly deciphering the ones on its right. Now, imagine a situation where all the entry images, and thus all the items, in the RPM-like dataset are the same. In this case, processing the first five dimensions first, rather than the last two, would be more efficient and easier for the AI system to detect the underlying pattern—everything is constant. This might be a very extreme example, but the general idea is clear. There should not be any fixed order to process input dimensions; at a certain point of processing for solving certain items, some dimensions are more informative than others. Therefore, the non-monotonic approach has its advantage over the monotonic approach as it sees all dimensions and allows to select the informative ones at each step.

Now, we formalize the non-monotonic approach. Suppose that we have an input array D of shape $(d_1,d_2,\ldots,d_l)$. Let $\Omega_1,\Omega_2,\ldots,\Omega_T\subset\{1,2,\ldots,l\}$ be subsets of dimensions that we are going to process sequentially. (in different orders). There exist $K=T!$ possible processing paths and we thus create K copies $\{D_1^{(0)},D_2^{(0)},\ldots,D_K^{(0)}\}$ of D, each copy being processed by a distinct path. A non-monotonic step, say Step

j, transforms $\{D_1^{(j)}, D_2^{(j)}, ..., D_K^{(j)}\}$ to $\{D_1^{(j+1)}, D_2^{(j+1)}, ..., D_K^{(j+1)}\}$ through Equation (4.11), (4.12), (4.13), and (4.14). For the i-th path where $i \in \{1, 2, ..., K\}$, we have a sequence of dimension subsets $\Omega_{i_1}, \Omega_{i_2}, ..., \Omega_{i_T}$, which is a permutation of the dimension subset $\{\Omega_1, \Omega_2, ..., \Omega_T\}$:

$$f_i^{(j)} = f_{i,\Omega_{i_T}}^{(j)} \circ f_{i,\Omega_{i_{T-1}}}^{(j)} \circ \cdots \circ f_{i,\Omega_{i_1}}^{(j)} \tag{4.11}$$

$$P_i^{(j)} = f_i^{(j)}\left(D_i^{(j)}\right) \tag{4.12}$$

$$C^{(j)} = g^{(j)}\left(P_1^{(j)}, P_2^{(j)}, ..., P_K^{(j)}\right) \tag{4.13}$$

$$D_i^{(j+1)} = P_i^{(j)} - C^{(j)} \tag{4.14}$$

where the subscript Ω_{i_k} indicates the function $f_{i,\Omega_{i_k}}^{(j)}$ processes dimensions in Ω_{i_k}. $f^{(j)} = \left(f_1^{(j)}, f_2^{(j)}, ..., f_K^{(j)}\right)$ and $g^{(j)}$ are the processing paths and the contrasting module of Step j, respectively. The processing paths $f^{(j)}$ and the contrasting module $g^{(j)}$ can be implemented using any standard deep learning modules, such as convolutions, linear, and pooling, or other custom operations, as long as the input and output dimensions are compatible with neighboring modules.

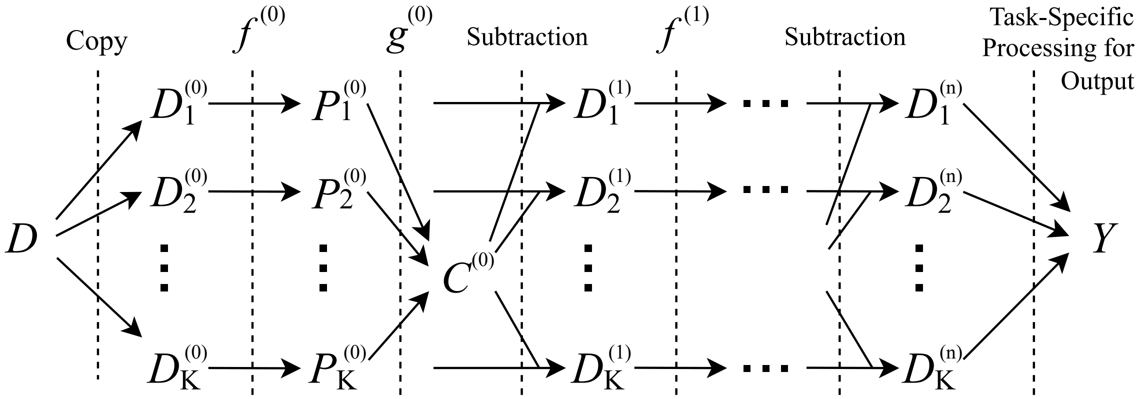


Figure 4.7: Non-monotonic processing of a multi-dimensional array D. This figure visualizes the computation of Equation (4.11), (4.12), (4.13), and (4.14).

Figure 4.7 depicts the non-monotonic approach in a more extended way. In practice, we can stack multiple non-monotonic steps together, as we stack multiple convolutional layers in monotonic models. At Step 0, $f^{(0)}$ extracts information from certain dimensions of the input data array D and optionally reduces the sizes of the dimensions; $g^{(0)}$ then contrasts the outputs from different paths. Since the parallel paths of $f^{(0)}$ are processing the same dimensions in different orders, the outputs of different paths are expected to be consistent (or even identical) with each other, as we have discussed for CPCNet.

4.10 Concluding Remarks

Through RPM, we can see some common characteristics of visual abstract reasoning. First, there must be multiple parts in an visual abstract reasoning item because the abstract concepts are in nature relations between multiple parts. In most cases, it requires using multiple dimensions to organize the multiple parts. The more complex the abstract conceptual system, the more additional dimensions needed. This nature of visual abstract reasoning tasks calls for effective multi-dimensional processing methods. The proposed CPCNet and the non-monotonic approach are such methods.

Second, the name "visual abstract reasoning" has been misleading for building AI systems as visual abstract reasoning tasks in intelligence tests are in no way pure reasoning problems. RPM tests and other visual abstract reasoning tests are for testing fluid intelligence, which is largely a superset of reasoning ability, or a precondition for the reasoning ability that we considered in our daily life. This point can be seen through the administration procedures of RPM tests, in which very sparse instructions are given. The human participant only knows that she needs to point at one of the answer choices to complete the matrix, but has no idea of what "complete" means and what is the criterion of being correct or wrong. Therefore, visual abstract reasoning as intelligence tests is primarily about discovering and secondarily about reasoning. In contrast, the AI systems for solving visual abstract reasoning is mainly working on the reasoning part, with the discovering part ideally designed by human designers. The proposed approach in this chapter intends to capture the discovering part by pursuing the consistency between different processing paths. This approach is by no means the only way to do so, and it might not be a very efficient one. But the crux of building capable AI is definitely on the "discovering" part.

CHAPTER 5

Computational Imagery

5.1 Introduction

The research question for this chapter is whether imagery, when implemented computationally in AI systems, is sufficient for solving visual abstract reasoning tasks. This Chapter incorporates finished works for this research question. In particular, the computational models in these works solve RPM and RPM-like tasks through only pixel representation of testing items and image operations. These works show that the imagery-based approach is sufficient to solve RPM tests and other similar tasks. They further show that different imagery abilities, such abilities manipulate mental images, can cause great variation in the task performance, which is similar to results of human intelligence testing. To clearly examine the effect of mental imagery, we use items from real intelligence tests to conduct experiments in this chapter.

Previous computational models have explored many important dimensions of solving RPM and related tasks, including the capacity for subgoaling (Carpenter et al., 1990; Kunda, 2015), pattern matching (Cirillo and Ström, 2010), rule induction (Rasmussen and Eliasmith, 2011), and dynamically re-representing and re-organizing visual elements (Lovett and Forbus, 2017). However, an integrated computational view of the solving process has been lacking. In Section 5.2, I present a systematic examination of how an imagery-based system can solve RPM by combining imagery representation, similarity metrics, analogy construction, and high-level strategies. As my base model, I use the Affine and Set Transformation Induction (ASTI) model, which operates on scanned, pixel-based images from the RPM test booklet and uses affine transformations and set operations to reason about image differences (Kunda et al., 2013; Kunda, 2013). In addition to presentation and evaluation of the new, expanded ASTI+ model, the contributions of this work include:

- A three-level search and reasoning framework for solving RPM problems. First, at the level of images, the agent can search across a known set of image *transformations* to interpret relationships within a given image pair or triplet (e.g., to explain the variation across a row, column, or diagonal). Second, at the level of a matrix, the agent can search across different *analogies* to find transfers of relationships across different image pairs or triplets. Third, the agent can use different *integration strategies* that combine results from the first two levels in different ways to produce the final answer.
- A finer taxonomy of *option-usage strategies* for solving RPM problems. Option usage, i.e., how an agent uses the given answer choices to guide its reasoning, are traditionally categorized into constructive matching and response elimination strategies (Bethell-Fox et al., 1984). We further divide

constructive matching into *option-free* and *option-informed* constructive matching.

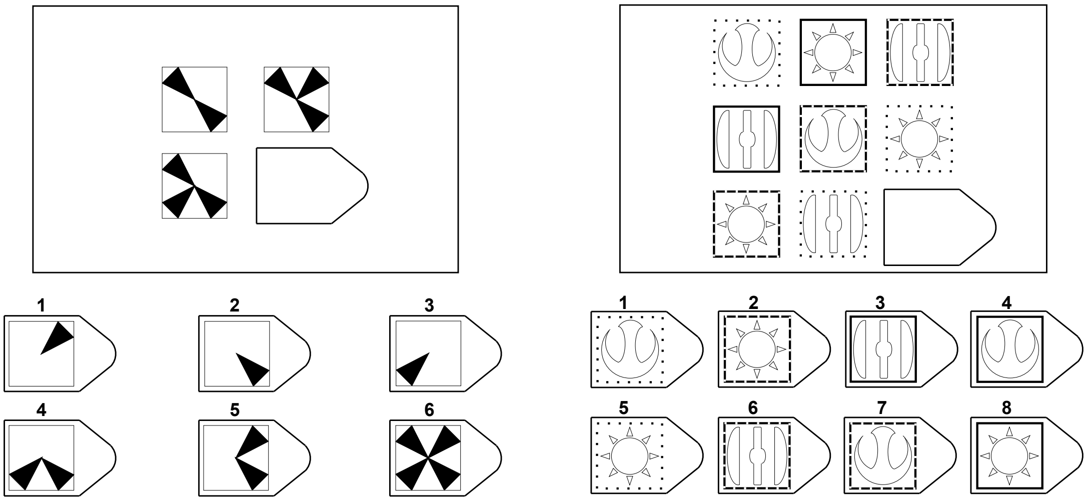
- A demonstration that an ASTI+ agent using a certain combination of transformations, analogies, and integration strategy can solve 57/60 items on the Raven's Standard Progressive Matrices test, which shows that these representations and inference mechanisms are sufficiently expressive and effective for driving successful performance on this class of geometric matrix reasoning tasks.
- Systematic ablation experiments that show how test performance varies as a function of agent knowledge and analogical construction strategy. The range of performance we obtain in these ablation experiments covers almost the entire range of human performance reported in normative studies of the Standard RPM test.

In the practice of using imagery-based systems to solve visual abstract reasoning tasks, I found that the performance of these systems are highly sensitive to visual similarity metrics. In particular, when there are hand-drawing and hand-writing in testing items, which are quite common in human intelligence tests, distortion and noises cause by hand-drawing and writing would pose a great challenge to traditional similarity metrics, such Jaccard index. In this case, it is possible that the metrics would give a very low score to extremely similar visual stimuli in human perception. Therefore, in Section 5.3, I propose a new visual similarity metric—soft Jaccard index—which is based the original Jaccard index. The advantage of this similarity metric is not only the robustness to distortion and noises but that it inherits the clear semantics of the traditional Jaccard index, which makes it applicable in all scenarios where the traditional Jaccard index applies.

Besides similarity metrics, analogy making is another core cognitive factor behind the solving process of RPM. Thus, Section 5.4 presents an analogy-making view toward the solving process of visual abstract reasoning. It focuses on the how analogy is established through constructing mappings between visual stimuli and organizing these mappings in a consistent way, which is meaningful for analogy-making. Similar to the ASTI+ model, the entire analogy making is based on imagery-based representation and the new soft Jaccard index.

5.2 ASTI+: an Imagery-Based Model for Solving RPM

In this section, the technical details of ASTI+ is described, including: 1) input and output representations, 2) similarity metrics, 3) image transformations, 4) matrix analogies, and 5) integration strategies. A comparison to its predecessor—ASTI (Kunda et al., 2013; Kunda, 2013)—is also provided. The experimental results and discussion of testing ASTI+ on the standard RPM tests are presented at the end of this section.



(a) 2×2 RPM-like problem (b) 3×3 RPM-like problem

Figure 5.1: Sample problems similar to those from the Standard version of the Raven's Progressive Matrices (RPM) test (Raven et al., 1998). Real RPM problems are not shown in order to protect the confidentiality of the test.

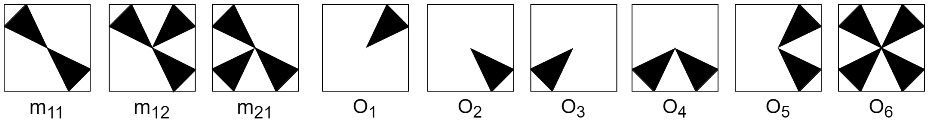


Figure 5.2: Illustration of input to our model for the 2×2 example problem in Figure 5.1, where m_{ij} is the matrix entry in row i and column j, and O_k is the k-th answer option.

5.2.1 Model Details

5.2.1.1 Input and Output Representations

Since the Standard RPM is in black and white, we represent each problem as a binary (i.e. pure black and white) image. Note that this is equivalent to representing an image as a set of black (or foreground) pixels, with each pixel represented by its coordinates in the image. Throughout this section, we use these two representations interchangeably.

Binary images are generated from grayscale scans of a paper copy of the Standard RPM test booklet. We select a threshold manually to convert grayscale values to binary values. We use an RPM-specific automated image-processing pipeline (Kunda, 2013) to decompose each full test page into images of individual matrix entries and answer options, as shown in Figure 5.2. We then feed these individual images as inputs to the ASTI+ model.

5.2.1.2 Similarity Metrics

One core reasoning component within the ASTI+ model specifies how to measure similarity between images. The model incorporates a similarity function that takes two images as input, and returns a real-valued number as output. For this purpose, we use the Jaccard index and the asymmetric Jaccard index, as shown in Equations (5.1) and (5.2), respectively:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{5.1}$$

$$J_A(A,B) = \frac{|A \cap B|}{|A|} \tag{5.2}$$

where sets A and B each represent a binary image. These indices essentially compute a measure of spatial overlap between two binary images. Equation (5.2) is asymmetric because $J_A(A,B)$ measures the extent to which A is inside (i.e., a subset of) B, and thus $J_A(A,B) \neq J_A(B,A)$.

A problem with Equations (5.1) and (5.2) is that the measures are sensitive to the alignment of pixels between A and B, i.e., if A has a black circle in the upper left, and B has a black circle in the lower right, then $J(A,B)$ and $J_A(A,B)$ might both be zero, even though there is considerable similarity between the pixel arrangements across those two image regions. However, in many geometric matrix reasoning problems, the images of matrix entries and options come in various shapes and sizes, and pixel elements within each image can vary in spatial location. We take a simple but robust approach to this problem (as in the original ASTI model): slide one image over the other, calculate a similarity value at every relative position, and select the maximum. In the process of sliding, images are padded to have the same shape and size.

$$S(A,B) = (J(A,B), pos_{AB}) \tag{5.3}$$

$$S_A(A,B) = (J_A(A,B), pos_{AB}, pos_{DA}, D) \tag{5.4}$$

As a result, similarity calculation in our model is defined by Equations (5.3) and (5.4), where $J(A,B)$ and $J_A(A,B)$ are the maximum similarity values at the relative position pos_{AB} of A to B. In Equation (5.4), $D = B - A$ is the difference between A and B when the maximum is reached, and pos_{DA} is the relative position of D to A.

When the maximum is achieved at multiple relative positions, we take the least shifted one. If multiple such least shifted positions exist, the agent breaks ties by always selecting the first one that is returned by the sliding window operation. Of course, there exist other methods for breaking ties, but we do not examine such variations here.

5.2.1.3 Transformations

The second component in the ASTI+ model is the specification of low-level visuospatial domain knowledge. ASTI+ represents this knowledge as a discrete set of image transformations that map from one or more input images to an output image, akin to operations of visual mental imagery done by humans such as mental rotation, mental image composition, etc. These functions operate on images at the pixel level, without re-representing visual information in terms of higher-order features. Although these functions were defined manually, based largely on inspections of the Raven's test, important directions for future work include expanding them to include higher-order features and concepts, as well as learning this knowledge from perceptual experience (Michelson et al., 2019).

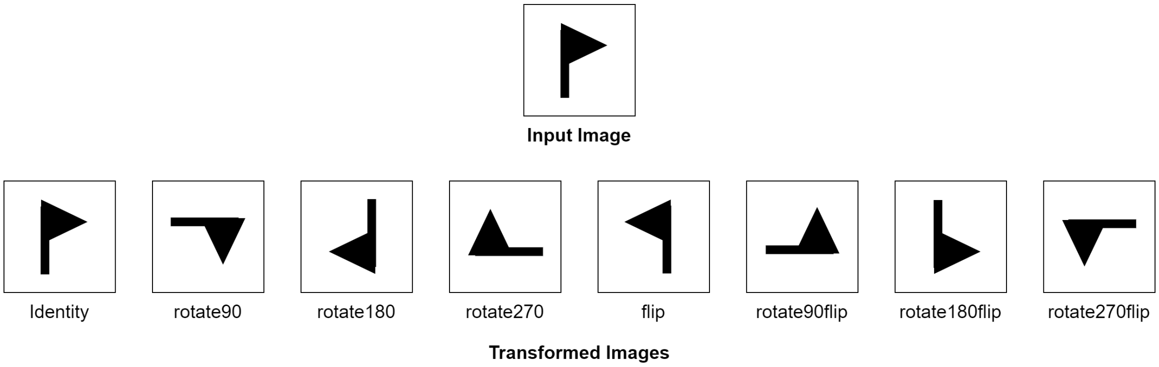


Figure 5.3: Illustrations of affine transformations used in our model.

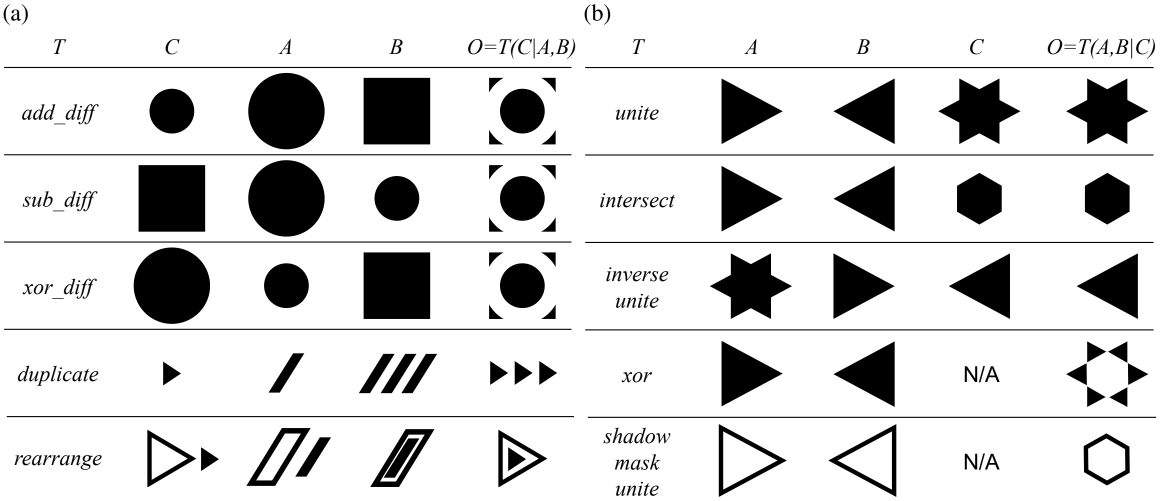
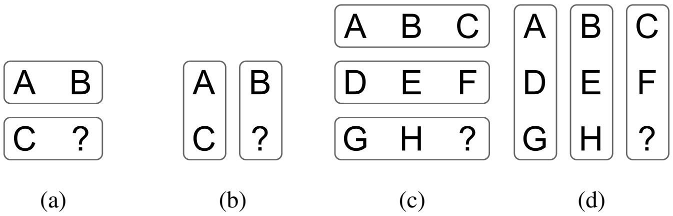


Figure 5.4: Illustrations of set transformations used in our model: (a) Given an analogy *A:B::C:?* and an unary set transformation T, the output image is $O{=}T(C|A,B)$, where C is the input, and B and C are parameters of T; (b) Given an analogy *A:B:C::D:E:?* and a binary set transformation T, the output image is $O{=}T(A,B|C)$ when T is applied on *A:B:C*, where A and B are the inputs, and C is a parameter of T, or $O{=}T(D,E|O')$ when T is applied on *D:E:?*, where O' is an option of the RPM problem.



(a) (b) (c) (d)

Figure 5.5: Illustrations of simple analogies in RPM problems. Simple analogies reflect how a matrix layout is naturally perceived as rows or columns. Particularly, given 2×2 matrices in (a) and (b), the row analogy is *A:B::C:?* and the column analogy is *A:C::B:?*; similarly, given 3×3 matrix in (c) and (d), the row analogies include *A:B:C::G:H:?* and *D:E:F::G:H:?*, and the column analogies include *A:D:G::C:F:?* and *B:E:H::C:F:?*.

ASTI+ includes two types of image transformations: unary and binary, which take a single input image and two input images, respectively, and then return one or more images as output. All ASTI+ transformations are based on fundamental affine transformations and set operations. These extend the original collections proposed in earlier ASTI research (Kunda et al., 2013; Kunda, 2013).

ASTI+ includes nine unary affine transformations: eight rectilinear rotations/reflections, as shown in Figure 5.3, and a ninth scaling transformation that doubles the area of the input image. There are also 11 additional set transformations: five unary and five binary, as shown in Figure 5.4a and 5.4b respectively, and one hybrid unary/binary transformation.

Table 5.1 gives details of each transformation. Unary transformations are defined relative to analogies between pairs of images, such as *A:B::C:D* for images *A*, *B*, *C* and *D*. Binary transformations are defined relative to analogies between trios of images, such as *A:B:C::D:E:F* for images *A*, *B*, *C*, *D*, *E* and *F*.

5.2.1.4 Analogies

The third element specifies how analogies are defined within a given RPM problem next. ASTI+ posits that an RPM analogy is composed of relations between matrix entries and that all parallel relations should be instantiated by the same transformation. This assumption seems adequate for most problems on the Standard Raven's test, but items on the Advanced test or other geometric analogy tests may require considering multiple transformations (Carpenter et al., 1990; Kunda, 2015).

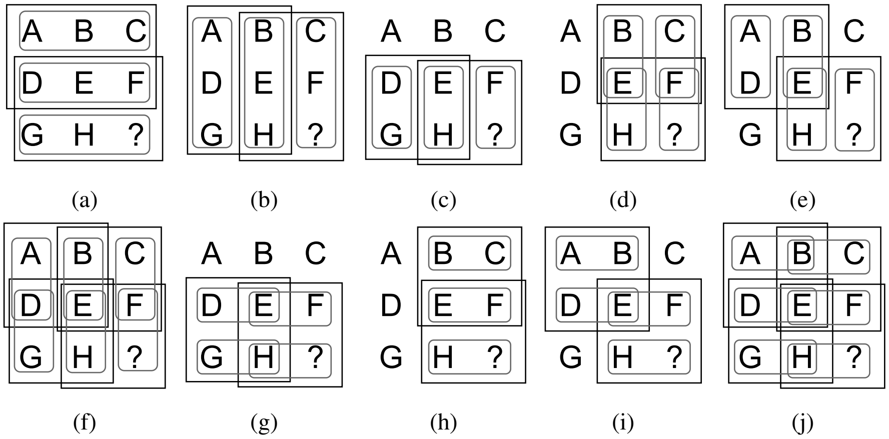
Figure 5.5 illustrates simple analogies that one could draw in any given RPM problem, where the images are represented by characters. These analogies are either between rows (Figure 5.5a and 5.5c) or between columns (Figure 5.5b and 5.5d), implying that the rows or columns share the same underlying relation among entries.

In addition to the simple analogies in Figure 5.5, the ASTI+ model also expands these analogies in two ways. First, for 3×3 matrices, the model further considers several subproblems, as shown in Figure 5.6. For

$add\_diff(C \mid A,B)$	Calculate $S_A(A,B)=(\cdots,pos_{DA},D)$. Align C and D using $pos_{DA\mid A=C}$. Output $O=C\cup D$.
$sub\_diff(C \mid A,B)$	Calculate $S_A(B,A)=(\cdots,pos_{BA},pos_{DB},D)$. Align C and D using $pos_{BA\mid A=C}$ and pos_{DB}. Output $O=C-D$.
$xor\_diff(C \mid A,B)$	Calculate $S(A,B)=(\cdots,pos_{AB})$. Align A and B using pos_{AB}, and calculate $D=A\oplus B$ and pos_{DA}. Align C and D using $pos_{DA\mid A=C}$. Output $O=C\oplus D$.
$duplicate(C \mid A,B)$	Let O be an empty image of the same size as B. Calculate $S_A(A,B)=(\cdots,pos_{AB},\cdots)$ and $B=B-A$ aligned by pos_{AB}, and copy C to the position of $pos_{AB\mid A=C}$ in O. Repeat this until nothing is left in B. Output O.
$rearrange(C \mid A,B)$	Let O be an empty image of the same size as B. Decompose C, A and B into connected components $C_1,C_2,\cdots,C_l$, $A_1,A_2,\cdots,A_m$ and $C_1,C_2,\cdots,C_n$. If $l=m=n$ is false, output a value indicating failure. Otherwise, find a permutation f of $\{1,2,\cdots,n\}$ that maximizes $\sum_{i=1}^{n} J(A_i,B_{f(i)})$ by calculating $S(A_i,B_j)=(J(A_i,B_j),pos_{A_iB_j})$ for each i and each j. Find another permutation g of $\{1,2,\cdots,n\}$ that minimizes $\sum_{i=1}^{n} distance(C_i,A_{g(i)})$. Generate O by copying C_i to position of $pos_{A_{g(i)}B_{f(g(i))}\mid A_{g(i)}=C_i}$ in O for all i.
$unite(A,B \mid C)$	Calculate $S_A(A,C)=(\cdots,pos_{AC},\cdots)$ and $S_A(B,C)=(\cdots,pos_{BC},\cdots)$. Align A and B with pos_{AC} and poc_{BC}. Output $O=A\cup B$.
$intersect(A,B \mid C)$	Calculate $S_A(C,A)=(\cdots,pos_{CA},\cdots)$ and $S_A(C,B)=(\cdots,pos_{CB},\cdots)$. Align A and B with pos_{CA} and poc_{CB}. Output $O=A\cap B$.
$IU(A,B \mid C)^{*}$	Calculate $S_A(B,A)=(\cdots,pos_{BA},\cdots)$ and $S_A(C,A)=(\cdots,pos_{CA},\cdots)$. Align A, B and C using pos_{BA} and pos_{CA}. Output image $O=A-(B-C)$.
$xor(A,B)$	Calculate $S(A,B)=(\cdots,pos_{AB})$. Align A and B by pos_{AB}. Output $O=A\oplus B$.
$SMU(A,B)^{*}$	Let X and Y be the shadows of A and B, where “shadow” is defined to be a copy of an image where any white area surrounded by black in the original image is colored black. Calculate $S(X,Y)=(\cdots,pos_{XY})$. Align X and Y using pos_{XY}, and calculate $M=X\cap Y$. Align A and B using $pos_{XY\mid X=A,Y=B}$. Output $O=M\cap(A\cup B)$.
$PSD(D,E \mid A,B,C)^{*}$	Given analogy $A{:}B{:}C{::}D{:}E{:}?$, $preserving\_sub\_diff$ works as $sub\_diff(E \mid B,C)$. But it requires that $A\subset B\cap C$ and $D\subset E\cap O$, where O is an option. Otherwise, output a value indicating failure. (This transformation is NOT shown in Figure 5.4.)

* $IU=inverse\_unite$, $SMU=shadow\_mask\_unite$, $PSD=preserving\_sub\_diff$

Table 5.1: Details of unary, binary, and hybrid unary/binary transformations.



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figure 5.6: Illustrations of recursive analogies in 3×3 RPM problems: (a) are (b) are trio analogies and (c) through (j) are pair analogies.

example, consider the simple analogies in Figure 5.5c, *A:B:C::G:H:?* and *D:E:F::G:H:?*, which use only two of the three rows. We then combine them into a larger recursive[1] format, *A:B:C::D:E:F:::D:E:F::G:H:?* as in Figure 5.6a, which use all rows. In this recursive analogy, two subproblems are created — the first subproblem is *A:B:C::D:E:?*, with *F* as the only option, and the second subproblem is *D:E:F::G:H:?*, with options from the original RPM problem. All subproblems should be solved equally well by the correct transformation.

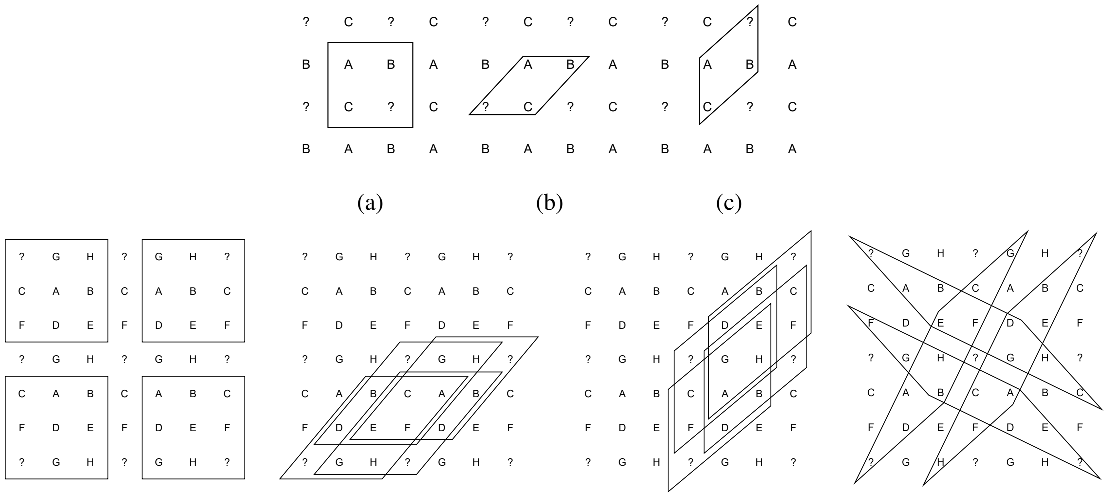
Second, ASTI+ captures more sophisticated spatial regularities by expanding the matrix in a way that the adjacency between matrix entries is preserved everywhere in the expanded version. Then it encloses different parts of the expanded matrix with quadrilaterals, as shown in Figure 5.7. The entries in each quadrilateral form a new matrix, whose rows and columns constitute analogies that can not be systematically constructed by rows and columns in the original matrix. ASTI+ follows two reasonable heuristics to enclose these matrices: (1) the quadrilateral should contain a permutation of the original matrix, and (2) the quadrilateral should have a *?* at one of its corners. We do not necessarily expect that humans use this strategy to search through this analogy space, but it provides a systematic and parsimonious way to capture regularities within a matrix that humans might perceive and reason about, albeit in different ways.

5.2.1.5 General Integration Strategy

The fourth element concerns the general strategy used to integrate transformations, analogies and similarity metrics to solve an RPM. The integration can be generally divided into three stages. In Stage 1, ASTI+ attempts to explain the variations in the incomplete matrix with some analogies and transformations. In Stage 2, it verifies the explanations by checking if there exists an option that can be generated from the analogy and

---

[1] Recursive in that it is an analogy of analogies.



(a) (b) (c)

(d) (e) (f) (g)

Figure 5.7: Expanded matrices to generate analogies: (a) through (c) are expanded from the 2×2 matrix in Figure 5.5, and (d) through (g) are expanded from the 3×3 matrix in Figure 5.5.

transformation. In Stage 3, it uses the best explanation—the best analogy and the best transformation—to select an answer option.

To quantify "how well" an analogy and a transformation explain the variations across matrix entries, we introduce three scores corresponding to the three stages, which are realized through different ways to assemble Jaccard similarity measurements: (1) the MAT score measures how well an analogy and a transformation explain the variations in the matrix in Stage 1; (2) the O score measures how well an analogy and a transformation explain the variations involving the options in Stage 2; and (3) the MATO score, which is used as the final metric to select the answer, is computed from the MAT and O scores. For example, given the matrix in Figure 5.5a, analogy *A:B::C:?* and transformation $flip(X)$, we have $\mathsf{MAT}=J(flip(A),B)$, $\mathsf{O}=J(flip(C),O)$ and $\mathsf{MATO}=(\mathsf{MAT}+\mathsf{O})/2$. Score calculation depends on what types of analogy and transformation are used, as described below.

MAT Scores. For transformations in forms of $T(A)$ or $T(A,B)$ (without extra parameters), MAT scores are calculated in the same way as $flip(X)$. For transformations with extra parameters, they cannot be computed in this way because the model does not know the extra parameters. For example, for $add\_diff(I|S,T)$ and *A:B::C:?*, it cannot use $\mathsf{MAT}=J(add\_diff(A|S,T),B)$ because it does not know S and T, but it can use $add\_diff(C|A,B)$ to calculate O score. In this case, the MAT score is calculated as $\mathsf{MAT}=J_A(A,B)$ for $add\_diff(I|S,T)$. Although the model takes transformation-specific approaches to calculate MAT scores, they are simply different ways to assemble similarity measurements (symmetric and asymmetric Jaccard indices) of the same known matrix entries.

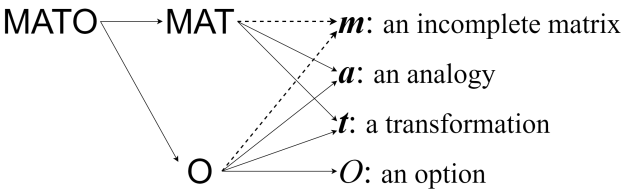


Figure 5.8: The dependencies of scores: The dashed lines denote partial dependence. Given the relations in an analogy, MAT relies on the entries that are not related to the missing entries while O relies on the entries that are related to the missing entries.

O Scores. For transformations whose MAT scores are calculated through the Jaccard index, so are their O scores. For transformations using the asymmetric Jaccard index, for example *add_diff* and *sub_diff*, the asymmetric Jaccard index is always higher than the Jaccard index given the same input (see Equation (5.1) and (5.2)). As a result, transformations measured by asymmetric Jaccard index tend to have higher scores even if their explanations are poor. To fix this issue, the model calculates multiple Jaccard and asymmetric Jaccard indices, each of which characterizes a distinct aspect of the transformation, and average them to get an O score. For example, for $add\_diff(C|A,B)$ and *A:B::C:?*, three aspects of the transformation are considered: (1) how much C is a subset of O, where O is an option, (2) how the difference between A and B compares to the difference between C and O and (3) how similar the predicted image is to O. This leads to $\mathsf{O}=(J_A(C,O)+J(D,D')+J(add\_diff(C|A,B),O)))/3$, where $D=B-A$ and $D'=O-C$ after A, B, C and O are properly aligned.

MATO Scores. Finally, every combination of an analogy, a transformation and an option is evaluated by a weighted average of its MAT and O scores, where the weight is proportional to the number of variations that the score measures. For recursive analogies in 3×3 matrices, scores of the original problem are derived from the scores of subproblems. For instance, suppose that there are n subproblems in a recursive analogy, and let MAT_k and O_k be the MAT score and O score of the k-th subproblem. In this case, the final MAT score is $\mathsf{MAT}=[\sum_{k=1}^{n-1}(\mathsf{MAT}_k+\mathsf{O}_k)+\mathsf{MAT}_n]/(2n-1)$ and the final MATO score is $\mathsf{MATO}=[\sum_{k=1}^{n}(\mathsf{MAT}_k+\mathsf{O}_k)]/2n$.

5.2.1.6 Specific Integration Strategies: When and What to Maximize

ASTI+ implements the general integration strategy as several alternative specific strategies that systematically explore different design choices in each stage of the general strategy. Given the dependencies of scores in Figure 5.8, the general strategy boils down to an optimization in which MATO score is maximized over the analogy $\boldsymbol{a}$, the transformation $\boldsymbol{t}$, and the option O for a problem-specific matrix $\boldsymbol{m}$. An heuristic for solving the optimization can be drawn from an observation on high-achieving human solvers — they often first form a good understanding of the incomplete matrix before attending to the options. This observation, translated into our scoring system, says that a good MAT score implies a good O score and thus a good MATO score. However, as most heuristics in intelligent systems, this heuristic might become invalid in some cases, for example, it will not work if the system does not have adequate capability to fully "understand" or explain the incomplete matrix (e.g. lacking appropriate transformations or analogies), or if the matrix contains distracting noisy features that cause the system to "over-explain" the content that should have been ignored.

For this reason, we introduced specific integration strategies (summarized in the first part of Table 5.2) that range from relying entirely on the heuristic to ignoring it. In particular, given an RPM matrix $\boldsymbol{m}$, an analogy $\boldsymbol{a}$, a transformation $\boldsymbol{t}$ and an option O, the MAT score is a function $\mathsf{MAT}(\boldsymbol{m},\boldsymbol{a},\boldsymbol{t})$, O score is a function $\mathsf{O}(\boldsymbol{m},\boldsymbol{a},\boldsymbol{t},O)$, and MATO is a function $\mathsf{MATO}(\mathsf{MAT},\mathsf{O})$. We formulate the three strategies as optimization processes, as shown below in (I), (II) and (III):

$$\begin{aligned} \mathsf{MATO}^{*} &= \max_{O} \mathsf{MATO}(\mathsf{MAT}(\boldsymbol{m},\boldsymbol{a}^{*},\boldsymbol{t}^{*}),\mathsf{O}(\boldsymbol{m},\boldsymbol{a}^{*},\boldsymbol{t}^{*},O)) \\ \boldsymbol{a}^{*},\boldsymbol{t}^{*} &= \operatorname*{argmax}_{\boldsymbol{a},\boldsymbol{t}} \mathsf{MAT}(\boldsymbol{m},\boldsymbol{a},\boldsymbol{t}) \end{aligned} \tag{I}$$

$$\begin{aligned} \mathsf{MATO}^{*} &= \max_{\boldsymbol{a},O} \mathsf{MATO}(\mathsf{MAT}(\boldsymbol{m},\boldsymbol{a},\boldsymbol{t}^{*}),\mathsf{O}(\boldsymbol{m},\boldsymbol{a},\boldsymbol{t}^{*},O)) \\ \boldsymbol{t}^{*} &= \operatorname*{argmax}_{\boldsymbol{t}} \mathsf{MAT}(\boldsymbol{m},\boldsymbol{a},\boldsymbol{t}) \end{aligned} \tag{II}$$

$$\mathsf{MATO}^{*} = \max_{\boldsymbol{a},\boldsymbol{t},O} \mathsf{MATO}(\mathsf{MAT}(\boldsymbol{m},\boldsymbol{a},\boldsymbol{t}),\mathsf{O}(\boldsymbol{m},\boldsymbol{a},\boldsymbol{t},O)) \tag{III}$$

where (I) completely relies on the heuristic, (III) completely ignores the heuristic, and (II) lies in between. We thus refer to optimizations (I), (II) and (III) as **M-confident**, **M-neutral** and **M-prudent** strategies, respectively, in the following discussion.

Since the O score also depends on the option O in Figure 5.8, it can also serve as the objective function to select an answer from the options. Therefore, ASTI+ has three analogous integration strategies for maximizing O, which we refer to as **O-confident**, **O-neutral** (IV) and **O-prudent** (V)) strategies:

$$\begin{aligned} \mathsf{O}^{*} &= \max_{\boldsymbol{a},O} \mathsf{O}(\boldsymbol{m},\boldsymbol{a},\boldsymbol{t}^{*},O) \\ \boldsymbol{t}^{*} &= \operatorname*{argmax}_{\boldsymbol{t}} \mathsf{MAT}(\boldsymbol{m},\boldsymbol{a},\boldsymbol{t}) \end{aligned} \tag{IV}$$

$$\mathsf{O}^{*}=\max_{\boldsymbol{a},\boldsymbol{t},O}\mathsf{O}(\boldsymbol{m},\boldsymbol{a},\boldsymbol{t},O) \tag{V}$$

Note that MATO is simply a weighted average of MAT and O, so the O-confident strategy is equivalent to M-confident (I). Thus we do not need a separate optimization for it.

5.2.2 From ASTI to ASTI+

In this subsection, we compare ASTI+ to its predecessor ASTI. The ASTI model (Kunda et al., 2013; Kunda, 2013) introduced a visual-imagery framework for solving geometric reasoning problem that based analogical reasoning on a pixel-level representation, transformations, and metrics. This framework remains unchanged in the ASTI+ model. From ASTI to ASTI+, we gave enhancements to the core dimensions of the framework.

5.2.2.1 Analogy

For 2×2 matrices, ASTI and ASTI+ share the same analogy set, which could be manually enumerated given the small size of matrices. In contrast, 3×3 matrices provide many more choices of analogies. We thus developed the systematic approach in Section 5.2.1.4 to enumerate analogies, which led to analogies that ASTI supported. We adopted this approach because analogues in matrix reasoning tasks are usually arranged in spatial parallelism. Another enhancement was the introduction of recursive analogy, which was inspired by the recursive and incremental nature of human solving reported in the literature (Carpenter et al., 1990; Kunda, 2015).

5.2.2.2 Transformation

ASTI+ inherits all the affine transformations of ASTI. Meanwhile, ASTI+ has extra complex set operations, such as *inverse unite* and *shadow mask unite*, that combine basic set operations in ASTI.

5.2.3 Integration Strategy

Compared to ASTI, ASTI+ has more choices of integration strategy representing different degrees of reliance on the heuristic mentioned in Section 5.2.1.6. In contrast, ASTI implements only one strategy that roughly equals the **M-prudent** strategy in ASTI+.

5.2.3.1 Option-Usage Strategy

Two general option-usage strategies for solving RPM problems and other multiple-choice reasoning problems have been reported in human studies: constructive matching and response elimination (Snow, 1980;

Bethell-Fox et al., 1984). Constructive matching proceeds as $infer(T) \to A = apply(T) \to test(A)$, where T is a transformation and A is an answer constructed by applying T. Response elimination proceeds as $infer(T_1) \to infer(T_2|O) \to compare(T_1, T_2)$, where O is an option used to infer T_2 and, if $compare(T_1, T_2)$ fails, O will be eliminated. The strategy choice observed in human experiments was found to relate to subject's intellectual ability, item type and difficulty. Cognitive models have been constructed based on both strategies (Evans, 1964; Sternberg, 1977; Mulholland et al., 1980).

ASTI strictly follows the constructive matching strategy, where options are never used before generating the missing entry. We refer to this constructive matching as **option-free constructive matching**. In contrast, ASTI+ adopts a slightly different approach that we refer to as **option-informed constructive matching**, which lies between constructive matching and response elimination. It follows the pattern $infer(T, p_1) \to infer(p_2|T, O) \& A = apply(T, p_2) \to evaluate(A, O, p_1, p_2)$, where p_1 and p_2 are parameters of T and p_2 is inferred from the option O. For example, options are used for calculating alignment parameters of the transformations in ASTI+. This strategy gives ASTI+ the flexibility to represent the relations that cannot be represented by single-direction transformations.

5.2.4 Experimental Studies of the ASTI+ Model

To study how different analogical constructions affect the performance on the RPM test, we equip ASTI+ with different configurations of analogies and transformations, and integration strategies, and test its performance on the standard RPM test, which consists of five sets of problems with 12 problems each. In our experiments, the analogical constructions are implemented as different configurations of analogies, transformations, and integration strategies. We further aggregated them into the groups summarized in Table 5.2. Each configuration has one or more groups of analogies and transformations, whereas it has only one integration strategy. We hypothesized that, by varying the configuration, the performance would change accordingly.

To study how each dimension of the configuration affects performance, we conducted two experiments. In the first one, we varied only the integration strategy and fixed the configuration of analogies and transformations (using the full set of analogies and transformations). In the second one, we selected the best integration strategy in the first one and varied the configurations of analogies and configurations.

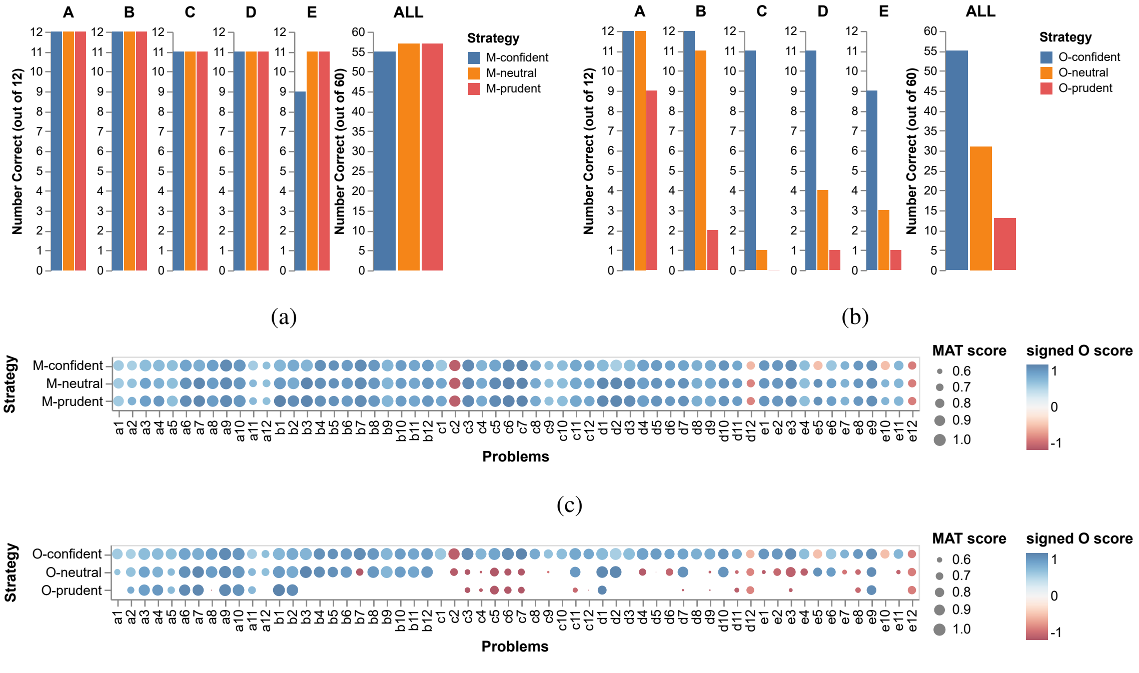
The first experiment compares the integration strategies. Figure 5.9 shows the set-wise and problem-wise performance of each one. The M-neutral strategy always ties with the M-prudent strategy, solving 57/60 problems, whereas the M-confident strategy performs slightly worse, solving 55/60 problems. The O strategies are far less capable, especially in the last three sets (C, D and E), where the problems are 3×3 (Set A and B contains only 2×2 problems). Thus, the M strategies, by considering both MAT and O scores, are more robust to increases in the matrix dimension.

Integration Strategies	
M-Confident	Find an analogy and a transformation that best explain the incomplete matrix; and then select an option that best matches the analogy and the transformation.
O-Confident	Mathematically equivalent to M-confident.
M-neutral	For each analogy, find a transformation that best explains the incomplete matrix; and then select an option such that there exist an analogy and its best transformation that match the option well and explain the incomplete matrix well.
O-neutral	For each analogy, find a transformation that best explains the incomplete matrix; and then select an option such that there exist an analogy and its best transformation that match the option well.
M-prudent	Select an option such that there exist an analogy and a transformation that match the option well and explain the incomplete matrix well.
O-prudent	Select an option such that there exist an analogy and a transformation that match the option well.
Transformation Groups	
Affine	All the affine transformations.
Diff	*add_diff*, *sub_diff*, *xor_diff*, and *preserving_sub_diff*.
Match	*duplicate* and *rearrange*.
Set	*unite*, *intersect*, *inverse_unite*, *xor* and *shadow_mask_unite*.
Analogy Groups	
S	The analogies in Figure 5.7a and 5.7d.
H	The analogies in Figure 5.7b and 5.7e.
V	The analogies in Figure 5.7c and 5.7f.
R	The analogies in Figure 5.7g.

Table 5.2: Configurations of integration strategies, analogy groups and transformation groups.

While the M-confident comes in last in Figure 5.9a by maximizing MATO, the O-confident fares best by maximizing O in Figure 5.9b. Furthermore, O-neutral and O-prudent strategies in Figure 5.9b contrast sharply with their counterparts in Figure 5.9a. In particular, the more a strategy relies on the heuristic from Section 5.2.1.6, the more performance drops when switching from maximizing MATO to maximizing O. We surmise that this is because the RPM is designed to have distractors with high O and low MAT. In other words, these distractors work like traps for strategies that maximize only O scores, which is consistent with observations that people often make errors of "repetition" while solving RPM problems (Kunda et al., 2016).

Figure 5.9c and 5.9d depict the scores for each strategy's answer to each problem as disks. MAT and O scores are encoded as size and color intensity, while the correctness of the answer is denoted by colors (blue for correct and red for incorrect). Note that the "signed" O score in Figure 5.9c and 5.9d is only to distinguish between correct and incorrect answers, and the real scores always fall in [0,1]. Figure 5.9c and 5.9d show



(a) (b)

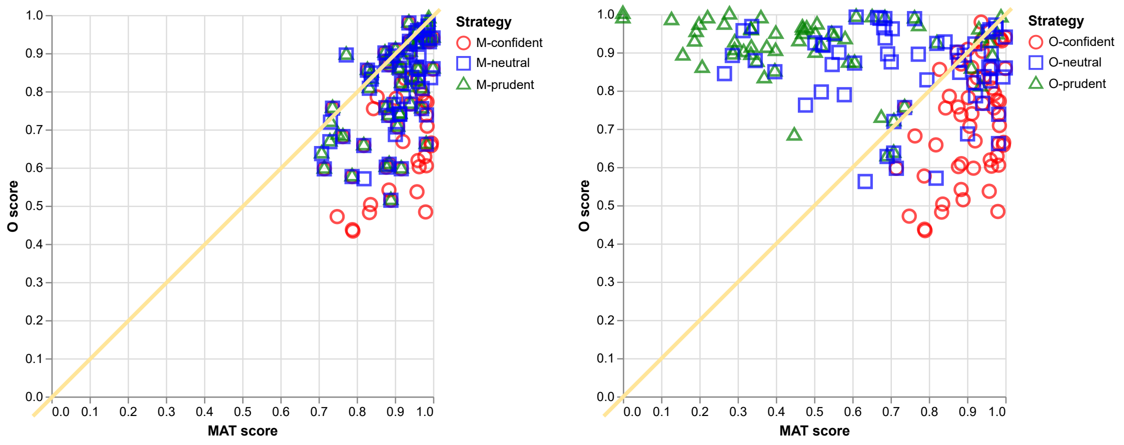
(c)

(d)

Figure 5.9: Performance of each strategy on the standard RPM test: (a) and (b) show numbers of problems correctly solved by each strategy in every set (A—E) and the entire test; (c) and (d) visualize MAT and O scores of each strategy's answer to each problem as disks of various sizes and colors, where red disks indicates incorrect answers and blue disks indicate correct answers. Note that the "signed" O score in (c) and (d) is only to distinguish visually between correct and incorrect answers, and the real scores always fall in [0,1].

a subtle difference: different strategies can have the same correct answer to a problem, but the answer may result from different analogies and transformations. Otherwise blue disks in any column would have the same size and color.

Figure 5.10a and 5.10b present the strategies' answers to every problem in scatter plots drawn with respect to the MAT and O scores, which show more difference between strategies. Note that most data points in Figure 5.10a, corresponding to the blue disks in Figure 5.9c, denote problems that are correctly solved. Since these data points in Figure 5.10a are mostly located near or below the diagonal, we could hypothesize that, for a "naive" participant or computational model (with little prior knowledge about RPM), a good explanation for the known matrix entries matters more than how an option can be matched. Recall that MAT and O are measurements of these two explanations. On the flip side, many more points, representing incorrect answers according to Figure 5.9d, fall above the diagonal in Figure 5.10b, which further supports this idea. The hypothesis is consistent with observations in previous human studies that high-achieving test takers usually take a more constructive approach, which requires a clear explanation of the matrix rather than perceptually matching the options (Bethell-Fox et al., 1984; Carpenter et al., 1990; Lovett and Forbus, 2017).



(a) (b)

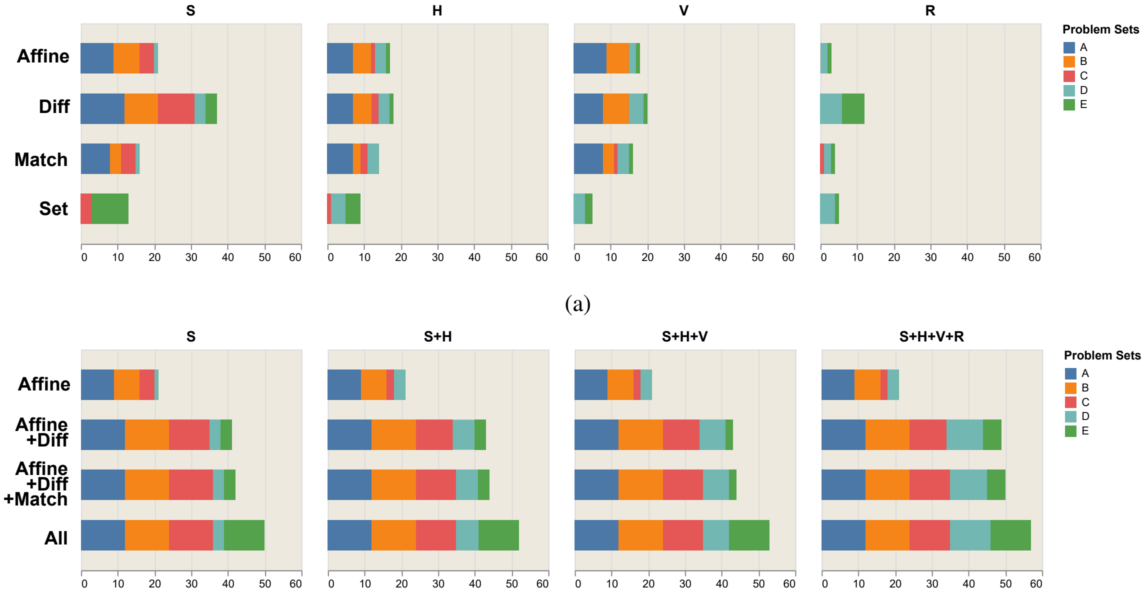
Figure 5.10: Scatter plots of each strategy's answer to each problem in the standard RPM test drawn with respect to the MAT and O scores.

In the second experiment, we compared different configurations of analogies and transformations while setting the integration strategy to the M-prudent strategy. Figure 5.11 shows the performance of different combinations of analogy and transformation groups in this situation. In particular, each analogy group is combined with each transformation group in Figure 5.11a, and analogy groups and transformation groups are combined in an incremental way in Figure 5.11b. In Figure 5.11a, we can see the strength and weakness of each analogy group and each transformation group. S analogies plus Diff transformations are good at problems in Set A, B, and C, whereas R analogies and Set transformations work well on Set D and E but work poorly on Set A, B, and C. Figure 5.11b shows increases in both the vertical and horizontal directions. The former are more substantial than the latter. This does not mean that transformations are more important than analogies, because, as seen in Figure 5.11a, the S group outperforms H, V, and R for every transformation group and most problems in Set A, B and C solved by H, V, and R can also be also solved by S with a different transformation. We might expect more variation across analogy groups if they were defined at a finer-grained level.

To conclude our analysis, we compare ASTI+'s performance with human performance (Raven et al., 1998) (i.e., the normative data of RPM). Figure 5.12 shows 95th, 50th and 5th percentiles of human performance (age from 6 to 19) in green curves and the performance of different configurations of ASTI+ used in our experiments in horizontal blue lines[2]. The ranges of ASTI+'s and human performance overlap substantially, suggesting that analogical construction has a great effect on the performance of the RPM test and that it probably has the same effect on the performance in other geometric reasoning tasks.

---

[2] Note that we did not use all the possible configurations, which would have resulted in wider and more even distribution of blue lines



(a)

(b)

Figure 5.11: Bar charts of numbers of problems correctly solved by M-prudent strategy using different analogy groups and transformation groups. (This figure should be viewed in color.)

5.2.5 Summary

This section described a framework of solving matrix reasoning tasks, including variations in transformations, analogies, and integration strategies. The experimental study shows that task-specific language of representations and inference mechanisms is quite expressive on the Raven's Standard Progressive Matrices test and that test performance varies not only as a function of transformations and analogies used, but also with the higher-level integration strategy: when and how, across analogies and transformations, the model performs its maximization calculations.

In tasks such as the RPM, where *eductive ability* (Spearman, 1923; Raven et al., 1998) is required to extract information from a new situation, redundant information often exists; otherwise, ambiguity cannot be eliminated because little prior knowledge is available. Methods for representing, identifying, and exploiting such redundancies are crucial to solving the problem. Analogy is often used for this purpose. By varying the configuration of the ASTI+ model, we alter its ability to identify and represent these redundancies and control the extent to which it can exploit them to solve the task.

This work has two main implications. First, for artificial intelligence, analogical ability might be needed for systems in new unseen situations. Second, for human intelligence, understanding analogical ability helps us understand eductive ability. ASTI+ demonstrates that analogical reasoning can be implemented in AI systems as exhaustive search on a predefined analogy space. Humans' analogical ability is far more sophisticated

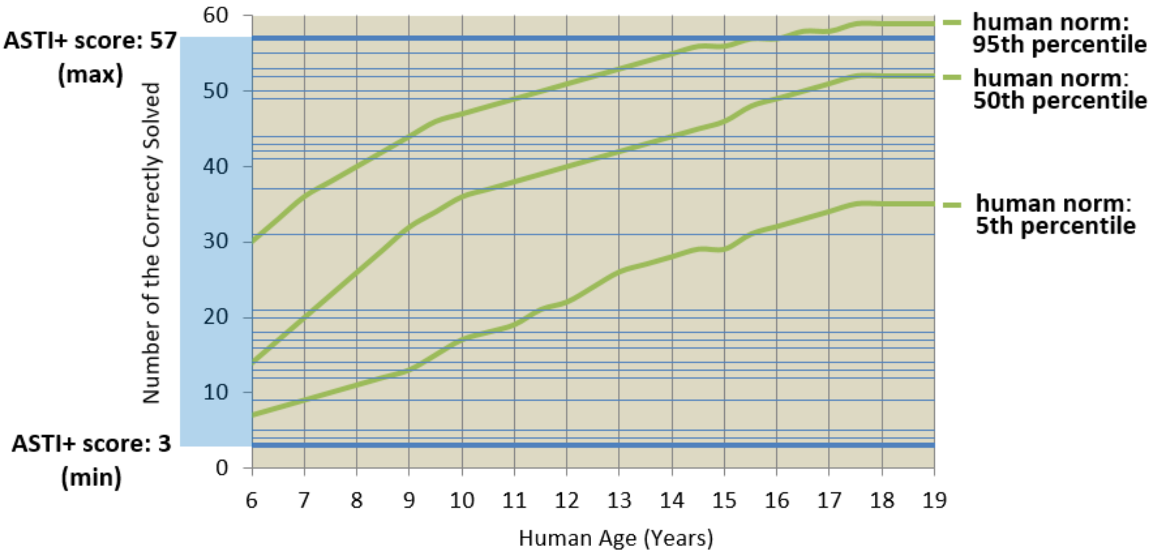


Figure 5.12: Comparison between ASTI+ and human subjects. The blue horizontal lines denote the performance of the configurations used in our experiments. The green curves represent the percentiles of human data.

than explicit search: it adapts to different complexity levels and task domains (Bethell-Fox et al., 1984), it involves goal management and selective attention in working memory (Carpenter et al., 1990; Primi, 2001), and it requires synergy between perception and cognition that works in a bidirectional and recursive way (Barsalou et al., 1999; Hofstadter, 2001). These features present a huge challenge to any existing analogy-making AI system.

The ASTI+ model uses only one analogy and one transformation to solve problems in the standard Raven's test. However, multiple analogies and transformations are required for problems beyond the standard test (Carpenter et al., 1990; Kunda, 2015) and, thus, adding methods that coordinate multiple reasoning pathways of different analogies and transformations.

Going one step further, virtually all extant computational RPM models, including ASTI+, employ a single strategy to solve every problem. However, there is ample evidence that people change strategies on Raven's problems, sometimes within a single testing session. For example, studies have found behavioral (DeShon et al., 1995) and neural (Prabhakaran et al., 1997) differences across test items linked to visual versus verbal problem-solving strategies, and other dimensions of strategy may exist. How do people manage these strategies and, possibly meta-cognitively, select options appropriate for problems? And how might an intelligent agent benefit from similar flexibility during complex problem solving?

Finally, although analogies and strategies are predefined in this research, there is the question of how humans learn such strategies, which, to our knowledge, no AI systems have accomplished for the Raven's test (Hernández-Orallo et al., 2016). Even RPM models that use learning still require the system designer to define the function to be maximized. Research in program induction may provide one path to tackle this thorny question (Schmid and Kitzelmann, 2011), including how strategies might be learned in the first place and adapted to new problems.

5.3 Soft Jaccard Index

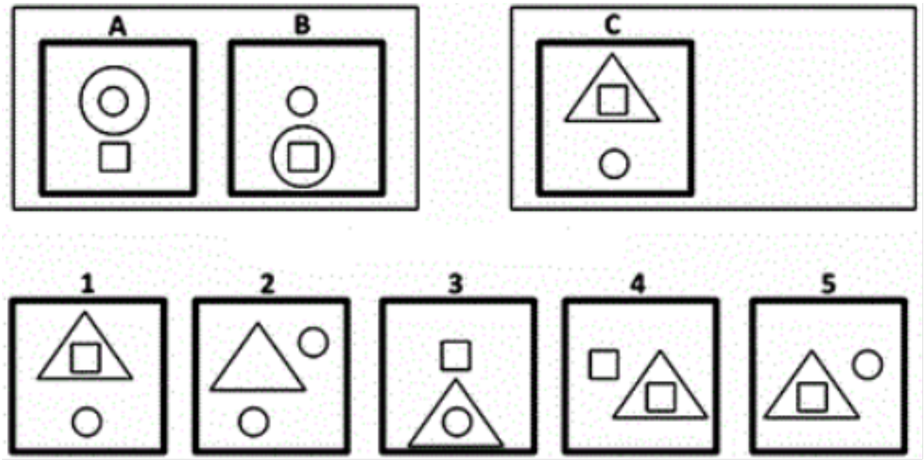


Figure 5.13: A simple example of geometric analogy problems (Lovett et al., 2009).

Imagery-based models are sensitive to the choice of similarity metrics. A basic formulation of similarity metric is the Jaccard index (Equation (5.5)), which measures the similarity between two finite sets (Kunda et al., 2013). In our works, these two sets consist of black pixels representing two geometric objects. Another useful variant of the Jaccard index is the asymmetric Jaccard index (Equation (5.6)) that measures the extent to which one set is a subset/inside of the other set.

$$J(A,B)=\frac{|A\cap B|}{|A\cup B|}=\frac{|A\cap B|}{|A\cap B|+|A\setminus B|+|B\setminus A|} \tag{5.5}$$

$$\vec{J}(A,B)=\frac{|A\cap B|}{|A|}=\frac{|A\cap B|}{|A\cap B|+|A\setminus B|} \tag{5.6}$$

The Jaccard index works well for geometric objects that are ideally drawn, such as those generated through vector graphics. But it is not as effective for geometric objects that human subjects would see in real psychological tests and in daily life. These visual stimuli are subject to distortion and noise, which pose a problem for imagery models using the Jaccard index. For example, applying Equation (5.5) on the scanning image of Figure 5.13, which contains distortion and noise that are imperceptible to human vision, the Jaccard index between the two large circles in *A* and *B* is only 0.25185; the Jaccard index between the small square in *A* and the small circle in *B* is 0.48649 — these measurements violate the correspondences implied by the verbal description.

$$A_b = \underset{x \in A}{\operatorname{argmin}}\, d(x,b) \text{ for each } b \in B \tag{5.7}$$

$$B_a = \underset{x \in B}{\operatorname{argmin}}\, d(a,x) \text{ for each } a \in A \tag{5.8}$$

$$M_0 = \{(a,b) \in A \times B \mid a \in A_b \wedge b \in B_a\} \tag{5.9}$$

$$A_0 = \{a \in A \mid \exists b \in B \text{ s.t. } (a,b) \in M_0\} \tag{5.10}$$

$$B_0 = \{b \in B \mid \exists a \in A \text{ s.t. } (a,b) \in M_0\} \tag{5.11}$$

$$T_a = \{(b,a') \in B \times A \mid b \in B_a \wedge a' \in A_b\} \text{ for each } a \in A \setminus A_0 \tag{5.12}$$

$$T_b = \{(a,b') \in A \times B \mid a \in A_b \wedge b' \in B_a\} \text{ for each } b \in B \setminus B_0 \tag{5.13}$$

$$M_1 = \{(a,b,a') \in (A \setminus A_0) \times B \times A \mid (b,a') \in \underset{(b,a') \in T_a}{\operatorname{argmin}} d(a,a')\} \tag{5.14}$$

$$M_2 = \{(b,a,b') \in (B \setminus B_0) \times A \times B \mid (a,b') \in \underset{(a,b') \in T_b}{\operatorname{argmin}} d(b,b')\} \tag{5.15}$$

$$d_0 = \frac{1}{|M_0|} \sum_{(a,b) \in M_0} |d(a,b)|^p \tag{5.16}$$

$$d_1 = \frac{1}{|M_1|} \sum_{(a,b,a') \in M_1} |d(a,a')|^p \tag{5.17}$$

$$d_2 = \frac{1}{|M_2|} \sum_{(b,a,b') \in M_2} |d(b,b')|^p \tag{5.18}$$

$$D(A,B) = d_0 + d_1 + d_2 \tag{5.19}$$

$$\vec{D}(A,B) = d_0 + d_1 \tag{5.20}$$

$$S(A,B) = e^{-\alpha D(A,B)} \tag{5.21}$$

$$\vec{S}(A,B) = e^{-\alpha \vec{D}(A,B)} \tag{5.22}$$

Figure 5.14 and Table 5.3 give a clearer example of this issue. The first row of the table is the Jaccard indices of the objects in Figure 5.14. According to these values, the square on the left is more similar to the triangle than to another square on the right, and, similarly, the circle on the left is more similar to the semicircle than to another circle on the right. Note that the sides of the two squares differ by only 1 pixel, and so do the radii of the two circles.

We designed another similarity metric, which inherits the general idea of the Jaccard index and is more robust to distortion and noise. The new metric does not require strict recurrences of elements in the two sets; instead, two elements can be considered “recurring" to some extent depending on the distance between them. Therefore, we name it soft Jaccard index.

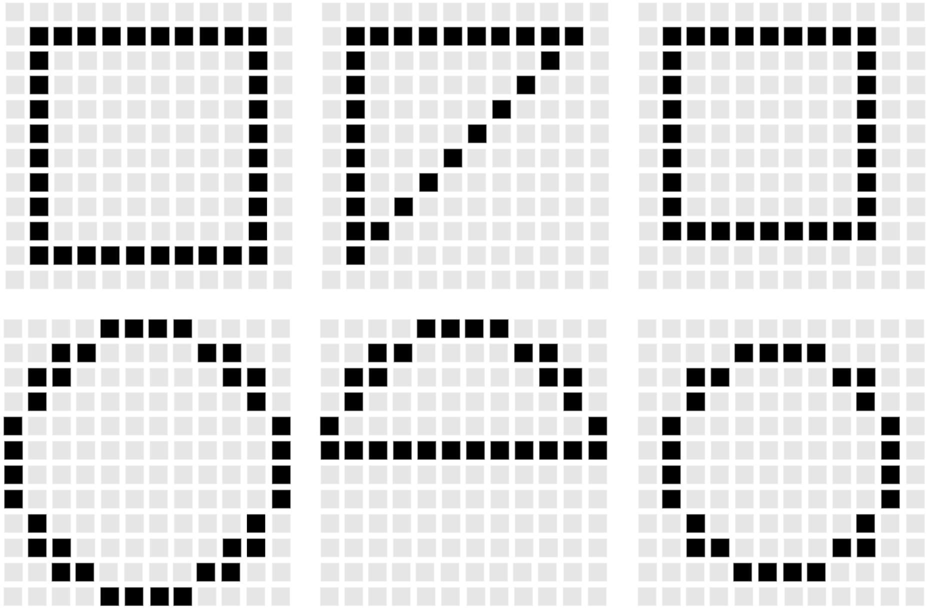


Figure 5.14: Geometric objects. Each cell in a grid denotes a pixel.

	□vs ◸		□vs □	○vs ⌓		○vs ○
Jaccard	0.4318	$>$	0.3333	0.3913	$>$	0.2307
Soft Jaccard	0.2076	$<$	0.9564	0.6461	$<$	0.9449

Table 5.3: Similarities between geometric objects in Figure 5.14. The Jaccard index is calculated using Equation (5.5). The soft Jaccard index is calculated using Equation (5.21) with α=0.03, d of one-norm and p=3.

Given two sets $A=\{a_1,a_2,\cdots,a_m\}$ and $B=\{b_1,b_2,\cdots,b_n\}$ from a metric space with a metric d, the soft Jaccard index of A and B is defined by Equation 5.7 through 5.22. Note that the argmin gives a set of values that equally minimize the objective function. Equation 5.21 and 5.22 are the symmetric and asymmetric versions. Like the Jaccard index, the soft Jaccard index also consists of three terms — corresponding to $|A \cap B|$, $|A \backslash B|$ and $|B \backslash A|$ in the Jaccard index — subscripted by 0, 1 and 2 in the equations. The difference is that every term's contribution is calculated from the metric d instead of set cardinality. To compare with the Jaccard index, the soft Jaccard indices for the objects in Figure 5.14 are in the second row of Table 5.3. The soft Jaccard indices are more consistent with human perception than the Jaccard ones.

5.4 An Analogy-Making View

If various cognitive gifts are the jewels in the crown of human intelligence, the analogy-making ability, as the core of cognition (Hofstadter, 2001), is undoubtedly one of the brightest ones. Analogy problems have always been an irreplaceable chapter in intelligence tests since they were first invented. In this section, we take a closer look at the analogy-making aspect of visual abstract reasoning. Particularly, we will investigate the correspondence-finding mechanism based on mapping and optimization. The work in this section is based on geometric analogy problems in human intelligence tests, which are also visual abstract reasoning tasks but more relevant to analogy making than RPM. Figure 5.13 gives a simple example of geometric analogy problems. To solve this problem, a subject needs to select an answer from the five options so that the analogy—A is to B as C is to the answer—makes sense.

Imagine that a human subject solved the problem in the figure. She would probably describe it this way: in the first two images, the large circle surrounding the small circle moves down to surround the small square; thus, in the last two images, the large triangle surrounding the small square should move down to surround the small circle, which gives us Option 3 as the answer. This simple description perfectly explains what happens in the analogy, and most people would accept it as a reasonable answer. However, excessively relying on verbal protocols is inappropriate because the verbal description after the subject already solved the item tends to disguise the complexity of geometric analogy problems as a cognitive task. In the first place, the verbal description is more of a consequence of the solving process rather than the solving process per se. Second, the solving process might involve cognitive components that are not consciously accessible to the subject and thus barely reflected in the verbal description. Last but not least, the verbal description uses high-level concepts and ignores the potential difficulty of how these concepts are formed or chosen given the visual stimuli. This part is probably far more complicated and influential in geometric analogy tasks than one would expect (Barsalou et al., 1999; Hofstadter, 1979).

While the work in this section is not intended as a high-fidelity cognitive model of human analogy making, it does provide insight into the representational and computational power (Thagard, 1996) of our specific formulation of analogy-making mechanism with implications for advancing the reasoning capabilities of artificial systems and for developing new hypotheses about mechanisms of visual imagery in human cognition.

5.4.1 Intuitions Behind the Proposed Approach

Before we go into the technical details, it is a better idea to sketch the intuition behind the approach. Again, taking the item in Figure 5.13 as an example, recall that the verbal description of it entails high-level concepts such as “circle”, “square”, “triangle”, “surrounding” and “moving down”, and why and how these high-level concepts end up in the verbal description is not so self-evident, yet very crucial to the complete solving process.

Imagine that you see only one image, say the first image, of the item in Figure 5.13, without any contextual information. How would you describe it? Perhaps still using the same set of concepts. But, more probably, different people might describe it differently. For example, it looks like a symbol of lollipop. This leads us to consider the context-dependent nature of concept individuation. In our case of geometric analogy problems,

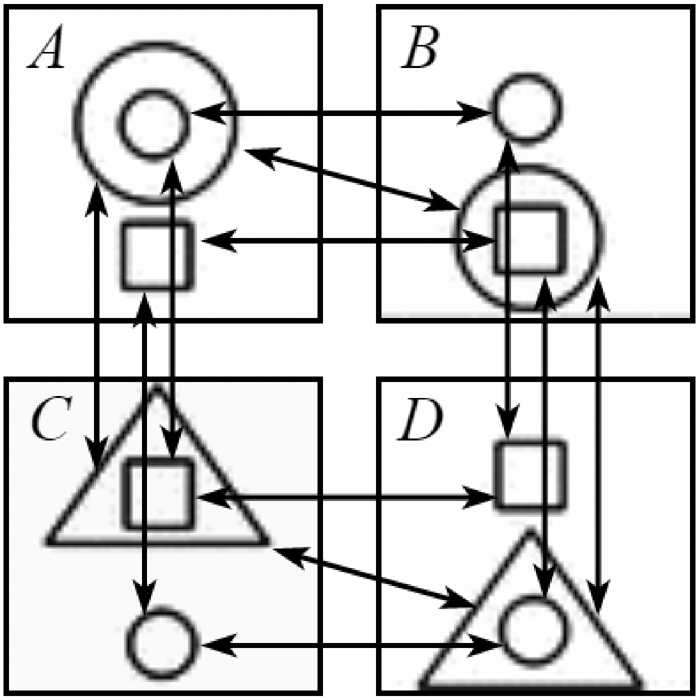


Figure 5.15: Correspondences between geometric objects. The correspondences between *A* and *B* and between *C* and *D* are shape mappings; the mappings between *A* and *C* and between *B* and *D* are derived from inside/outside spatial relations.

the contextual information is the correspondence among geometric objects in the four images. Moreover, the concept individuation and the correspondence finding in solving process are better to be regarded as two viewpoints toward the same thing. For example, correspondences in our example problem can be depicted as in Figure 5.15, in which the horizontal ones are based on similarity and the vertical ones are based on spatial relation. One could say that the conceptual role of each object gives the correspondences, or, the other way around, that the correspondences determine the conceptual role of each object.

From a problem-solving perspective, when embedded into the incomplete analogy, a correct option would induce a self-consistent set of correspondences, or, equivalently, a self-consistent set of conceptual roles. This type of self-consistency can be formally verified by a process of consistency check: given the correspondences, two pathways exist between two diagonal images; for each starting object in each image, whichever pathway is followed, it should lead to the same ending object in the diagonal image. For example, in Figure 5.15, on one hand, the large circle in *A* corresponds with the large circle in *B*, which corresponds with the large triangle in *D*; on the other, the large circle in *A* corresponds with the large triangle in *C*, which corresponds with the large triangle in *D*. The choice of using diagonal images in consistency check is because each pathway contains the correspondences in both directions.

There are two general analogy-making theories. The first theory assumes a base domain and a target domain and, by comparing the relational structures in these two domains, mappings between them are inferred (Gentner, 1983). When the relational structures or domains are not clearly defined, analogy-making is usually performed through the second theory where a dynamic process is employed, in which structures and correspondences between structures adapt to each other and settle on an equilibrium (Barsalou et al., 1999; Hofstadter, 1979; Mitchell, 1993). For the purpose of end-to-end modeling of the solving geometric analogy problems, the second theory is preferable. In particular, the domains are not clearly defined and the desirable equilibrium is realized as a self-consistent set of correspondences. Note that given the proportional format of geometric analogy, the base and target domains are not clearly defined (i.e., central permutation property (Prade and Richard, 2009, 2010, 2013)); so are relational structures and mappings between structures. In the rest of this section, we will discuss the technical details of the end-to-end modeling of solving geometric analogy problems, which bear resemblance to the second theory of analogy-making.

5.4.2 Correspondence Finding

The example analogy in Figure 5.13 can be characterized by a consistent set of correspondences in two analogical directions. In this section, we discuss how these correspondences can be found and used to interpret an analogy in a broader sense. We formulate conceptual correspondences in analogies as mathematical mappings. Thus, treatments of mathematical mappings could help understand analogy-making and modeling. We first consider two independent dimensions of mappings:

Qualitative vs Quantitative. A mapping can be derived from either qualitative relations or quantitative relations. A qualitative mapping depends on whether there is a good match between two qualitative relational structures. Thus, the validity of a qualitative mapping is considered binary. In contrast, a quantitative mapping is associated with a continuous score, say between 0 and 1, to indicate the extent of its validity. To let them work together, we give every qualitative mapping a score of 1 if it is valid or 0 if not.

Simple vs Complex. A mapping can also be derived either directly from geometric attributes of objects or from other mappings. Let us call them simple and complex mappings, respectively. In a sense, a complex mapping represents an isomorphism between two structures defined by two groups of mappings.

General Quantitative Mapping. Qualitative mappings are relatively easy to determine through structure matching, whereas quantitative mappings require additional considerations to coordinate multiple factors: (a) strong relations are preferable to weak ones; (b) the derived mapping should be unambiguous (i.e., injective) in that any two mapped objects should mutually be each other's best match; (c) the size of the mapping should be as large as possible to capture the largest isomorphism. Thus, we designed a template method to derive quantitative mappings as shown in Equation (5.23), where, given two sets $U=\{u_1,u_2,\cdots,u_m\}$ and $V=\{v_1,v_2,\cdots,v_n\}$ of objects, whether u_i and v_j are mapped to each other is denoted by $x_{ij}=1$ or 0, and

$s_{ij} \in \mathbb{R}$ denotes a measurement of the relation between u_i and v_j, for example, similarity. The above factors are thus integrated into the optimization in Equation (5.23), where x_{ij} and t are variables. Note that, in this formulation, we assume larger values of s_{ij} indicate stronger relations. If smaller values of s_{ij} indicate stronger relations, the equations need to be accordingly negated.

$$\begin{aligned} &\max \sum_{i,j} x_{ij} \\ \text{s.t. } &1 \geq \sum_{j} x_{ij} \text{ for all } i \\ &1 \geq \sum_{i} x_{ij} \text{ for all } j \\ &x_{ij} = x_{ji} \text{ for all } i,j \\ &(x_{ij} - 0.5)(s_{ij} - t) > 0 \text{ for all } i,j \\ &x_{ij} \in \{0,1\} \text{ for all } i,j \text{ ,and } t \in \mathbb{R} \end{aligned} \tag{5.23}$$

Given the two dimensions of mappings and general quantitative mapping, we introduce the specific mappings:

Simple Quantitative: Shape Mapping. When the soft Jaccard index is used as the strength of relation in Equation (5.23), we obtain a mapping reflecting shape similarity. Since the soft Jaccard index gives values between 0 and 1, we use the minimum strength of the selected relations as the score of the mapping.

Simple Quantitative: Location Mapping. When Euclidean distance between objects is used as the strength of relation in Equation (5.23), we obtain a mapping based on the locations of objects. In this case, the lower strength values indicate stronger relations. The score of this mapping is calculated as the normalized maximum strength of the selected relations.

Complex Quantitative Mappings. Let $M_1 : A \rightarrow B$ and $M_2 : C \rightarrow D$ be two injective mappings. A **delta shape mappings** is a complex quantitative mapping constructed from M_1 and M_2, representing the idea that the same shape change happens from A to B and from C to D. Similarly, a **delta location mappings** based on M_1 and M_2 represent the idea that the same location change happens from A to B and from C to D. These two complex quantitative mappings are thus between A and C and between B and D, orthogonal to the directions of M_1 and M_2. Specifically, we require that M_1 and M_2 have to be of the same size, and all the pairs in M_1 and M_2 have to be mapped in these two types of delta mappings.

Complex Quantitative: Delta Shape Mapping. When the difference between the soft Jaccard index of each M_1 pair and the soft Jaccard index of each M_2 pair is used as the strength of relation in Equation (5.23), we obtain the delta shape mapping. The score of this mapping is calculated from the maximum strength of the selected relations.

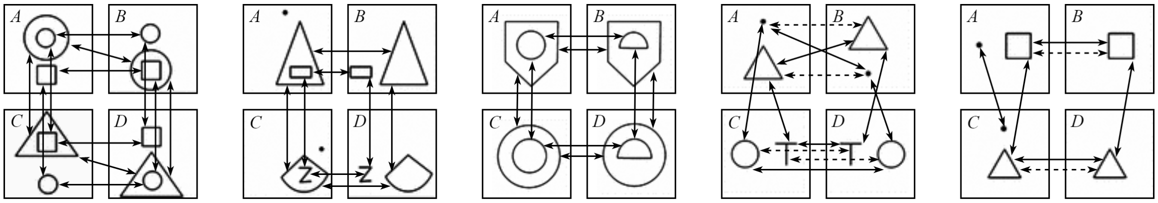
Complex Quantitative: Delta Location Mapping. When the difference between the distance of each M_1 pair and the distance of each M_2 pair is used as the strength of relation in Equation (5.23), we obtain the delta location mapping. The score of this mapping is calculated from the normalized maximum strength of the selected relations.

Simple Qualitative: Inside/Outside Mapping. Relations such as regional connection calculus were supposed to be used here. But for rapid prototyping, we use only the inside/outside relation. An inside/outside mapping exists if the relational structures of one set can strictly match the relational structure of the other set, and thus has a binary score of 0 or 1. The internal area of a geometric object can be labeled by pixel-level algorithms and the inside/outside relations can thus be determined by set operations between the internal area and other objects.

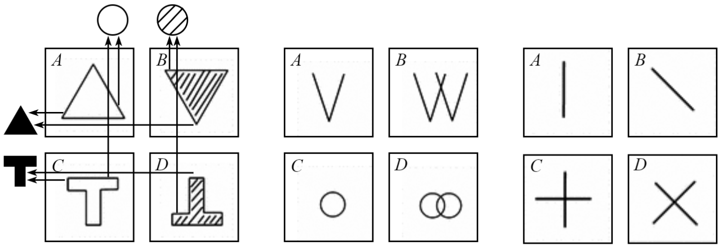
Complex Qualitative: Edge-Labeled Isomorphism Between Bipartite Multigraphs. Let $f_1, f_2, \cdots, f_n$ be injective mappings between sets A and B, and $f'_1, f'_2, \cdots, f'_n$ be injective mappings between sets C and D. These two groups of mappings form two edge-labeled bipartite multigraphs with labels in $\{1, 2, \cdots, n\}$. We can derive two new mappings between A and C and between B and D from any label-preserving isomorphism between these two multigraphs. The score of the mapping is 1 if such isomorphism exists; otherwise 0.

There are cases when mappings are theoretically workable, but cumbersome. For example, when a geometric object is rotated or mirrored, we can certainly map every point of the object to where they are moved to. But a more efficient solution is to consider the transformation of the whole object. Therefore, we also include common affine transformations in our toolbox, and, using the soft Jaccard index, we score the validity of these transformations, as we did for mappings.

What if there is no good mapping? No useful mapping can be derived in some cases. For example, given only one object in each item, we have no choice but to map them to one another even if the scores are quite low; or an object maps to multiple objects equally well (not injective mapping), thus crippling the reasoning through the mapping. In these cases, mapping is no longer suitable to represent the relations in an analogy and we fall back to the approaches of the ASTI+ model. For example, in our experiment, Problem 2 requires a 45-degree rotation to represent the relation; Problem 16 requires duplication to represent the relation; Problem 13 requires pixel density to represent the relation. We derive these non-mapping relations still at the pixel level and calculate a score for each of them. Therefore, these mapping and non-mapping relations can work together to interpret an analogy. However, further research on more coherent integration and unification of these conceptual approaches is needed. These mapping and non-mapping relations are combined to form interpretations for geometric analogies. In particular, each interpretation is to be applied to the analogy completed by each option, and a score is calculated for each pair of interpretation and option



Interp. 1, Problem 5 Interp. 2, Problem 1 Interp. 3, Problem 8 Interp. 4, Problem 20 Interp. 5, Problem 15



Interp. 6, Problem 13 Interp. 7, Problem 16 Interp. 8, Problem 2

Figure 5.16: The typical problems solved by each interpretation.

by aggregating the scores of mapping and non-mapping relations in the interpretation. The interpretation and the option are selected by the highest score. Candidate interpretations are constructed from these mapping and non-mapping relations (see Table 5.4).

5.4.3 Experimental Studies

Interpretation	*A:B::C:?*	*A:C::B:?*	Mapping/Transformation	Consistency Check	Solved Problems
1	shape	inside/outside	Mapping	yes	3, 5, 7, 9, 11,17
2	shape	delta location	Mapping	no	1, 4
3	inside/outside	delta shape	Mapping	no	8
4	shape & location	isomorphism	Mapping	no	10, 20
5	shape = location	shape	Mapping	yes	15
6	density change	shape change	Transformation	no	13
7	duplicate	N/A	Transformation	no	16
8	affine	N/A	Transformation	no	2, 6, 12, 14, 18, 19

Table 5.4: Experimental Results on Geometric Analogy Problems. The second and third columns are two analogical directions. The last column shows the problems that were solved by that interpretation.

Besides the aforementioned mappings, we construct another conceptual layer in our modeling — interpretation — by assigning mappings or transformations to the two analogical directions, i.e., each interpretation is a combination of specific mappings or transformations. To solve a geometric analogy problem, each interpretation is scored for the geometric analogy obtained by inserting each option into the incomplete analogy, by aggregating the scores of its mappings or transformations. The interpretation and option of the highest score are selected as the answer to the problem. Following this outline, we implemented a computational model and ran it on a classical set of 20 geometric analogy problems (details found in (Lovett et al., 2009)), which was published in the 1942 edition of the *Psychological Test for College Freshmen of the American Council on Education*.

The experimental results are summarized in Table 5.4. All the 20 geometric analogy problems were solved by 8 different interpretations. Table 5.4 lists each interpretation's mappings or transformations in the two analogical directions, and the solved problems. The first five interpretations are mapping interpretations, among which Interpretation 1 and 5 require a successful consistency check because their mappings in the two analogical directions are independently derived; in contrast, in Interpretation 2, 3 and 4, the complex mappings in one direction are built upon the simple mappings in the other direction with the consistency assumed to be true. Although the consistency holds in both cases, the corresponding analogies and how these analogies are processed are different. Interpretation 6, 7 and 8 are transformation interpretations, which apply to a large portion of the problems. This implies that, in addition to consistent mappings, visual imagery and mental transformation are another important facet of analogy-making.

It is worth pointing out that Problem 19 can be solved by two affine transformations—rotation and reflection—leading to different options. The rotation option won out marginally in our experiment, but the reflection option is more human-preferred.

To give a straightforward description of how the model works, we select for each interpretation a problem to describe the details. These problems and interpretations are visualized in Figure 5.16.

Interpretation 1: Problem 5 shows an analogy of topological variation between the two rows. Horizontally, two shape mappings $A \rightarrow B$ and $C \rightarrow D$ are constructed. Vertically, two inside/outside mappings $A \rightarrow C$ and $B \rightarrow D$ are constructed. These four mappings are consistent and characterize the repetition of the same topological change in the two rows.

Interpretation 2: Problem 1 shows an analogy of location change between the two rows. Horizontally, two shape mappings $A \rightarrow B$ and $C \rightarrow D$ are constructed. Vertically, two delta location mappings $A \rightarrow C$ and $B \rightarrow D$ are constructed on the basis of the horizontal shape mappings. The repetition of the same location change in the two rows is characterized by these four mappings.

Interpretation 3: Problem 8 shows an analogy of shape change between the two rows. This illustration is parallel to Interpretation 2's except that it describes shape change instead of location change, using inside/outside mapping instead of shape mapping in the horizontal direction.

Interpretation 4: Problem 20 shows an analogy of location exchange between the two rows. Note that location exchange is different from location change in that an object can only move to a previously-occupied place, and thus the movement is relative, whereas location change is the absolute movement in the global coordinate system. Horizontally, two types of mappings are constructed, where the dashed line indicates location mappings and the solid line indicates shape mappings. Vertically, an edge-labeled isomorphism is constructed on the basis of the horizontal mappings, where different mapping types serve as edge labels. The repetition of the same location exchange in the two rows is characterized by these mappings.

Interpretation 5: Problem 15 shows an analogy of adding or removing objects between the two rows. Horizontally, like Interpretation 4, shape mappings and location mappings are constructed but these two types of mappings are required to agree with each other. Vertically, shape mapping is constructed. Therefore, the same change of adding or removing objects in the two rows is described by these mappings. A consistency check is needed.

Interpretation 6: Problem 13 shows an analogy of texture change and shape change between the two rows. Horizontally, the texture change was supposed to be measured, but due to the lack of a general computational representation for texture, the density change of black pixel is used to approximate texture change. In particular, we measure the density of black pixels in the area occupied by the geometric object. Given d_A, d_B, d_C and d_B as the densities of A, B, C and D, they are expected to be $d_A{=}d_C$, $d_B{=}d_D$ and $d_A{-}d_B{=}d_C{-}d_D$. Vertically, shape change is represented by the change in soft Jaccard Index. The changes are supposed to be equal in rows and columns. The score depends on how much all the expectations are satisfied.

Interpretation 7: Problem 16 shows an analogy of duplication between the two rows. Horizontally, objects are duplicated in the same way (same location arrangement) from A to B and from C to D. The location arrangement is determined by repeatedly calculating the asymmetric soft Jaccard index at different relative locations and subtracting the best-match part from what is left. Vertically, a general quantitative mapping between the two sets of locations is computed using Euclidean distance as the strength of relations. The score of this interpretation is calculated from the minimum asymmetric soft Jaccard index and the maximum significant level of selected correspondences. Note that, to differentiate intersection, tangency and closeness, the location coordinates have to be normalized by the dimensions of the original object.

Interpretation 8: Problem 2 shows an analogy of affine transformation, a 45-degree rotation in this case, between the two rows. The soft Jaccard index is used to determine which affine transformation best matches the variation.

5.4.4 Summary

In this section, we proposed a model for solving geometric analogy problems. For each problem, the model selects an option and an interpretation, which is based on the mappings or transformations in the two analogical directions. Making or interpreting an analogy is a very delicate dynamic process in human cognition, in which multiple aspects and levels of perception and cognition are closely cooperating with one another, and gradually, magically, settling on a perfect harmony that make sense to all the aspects and levels. The intricacy behind this cognitive process is far more complicated than an object mapping or a numerical optimization.

CHAPTER 6

Future Work

There is more exciting work to be done on all the research questions that I raised in Chapter 2. For example, for the second research question about the interplay mechanism between perceptual and conceptual processing, I explored only one specific way to implement such interplay. Although the proposed model performed well on the RAVEN family and the trivial generalization regimes of PGM, it showed just a weak sign of generalization in the nontrivial generalization regimes of PGM, which are, however, more valued in the sense of human intelligence testing. Therefore, to fully understand the effect of the interplay on nontrivial generalization regimes of visual abstract reasoning tasks, other possible implementations of the interplay of perceptual and conceptual processing need to be investigated. For example, unlike CPCNet's implicit way to implement the interplay, implementing it in an explicit way might be a better choice for it probably allows more delicate control over the processing, e.g., regularization.

On the flip side, PGM might not be a perfect dataset for studying nontrivial generalization of visual abstract reasoning because the triplets $[r,o,a]$ of PGM are not defined in a systematic way that allows us to evaluate different types, different dimensions, and different levels of nontrivial generalization. For example, although the entangled components o and a of PGM make the held-out shape-color and line-type regimes quite challenging for current learning models, we are not sure if they are really meaningful nontrivial generalization regimes and where they lie in the map of visual abstract reasoning. Therefore, another future work for Research Question 2 is to construct a more clearly-defined dataset for the purpose of evaluating visual abstract reasoning ability.

In addition, my work on Research Question 4 was limited due to the unexpected technical difficulties for the first three research questions, for example, the imbalanced dataset and insufficient computational resources for solving RPM-like datasets. Therefore, in the rest of chapter, I will elaborate on Research Question 4 and possible future works for it. The research question is how generative models can be extended to imagery-based production systems that are able to produce a flexible reasoning trajectory in computational imagery. To clearly depict this research question, I set three specific goals for it.

6.1 Goal 1

This research question could be considered as a continuation of the works in Chapter 5. In that chapter, the discussion was centered around the idea of computational imagery. In particular, the input, the intermediate results, and the output of imagery-based approach are all pixel images; the reasoning and solving processes are all realized through image operations, similarity computation, and analogy construction. The experimental results demonstrated the effectiveness of imagery-based approach on RPM and geometric analogy problems. However, an obvious defect of these imagery-based models is that they are hardcoded and rigid compared to learning models, for example the model described in Chapter 1. **Therefore, the first goal is to construct a flexible, robust learning model that simulates the behavior of imagery-based models on RPM and other visual abstract reasoning tasks.**

6.2 Goal 2

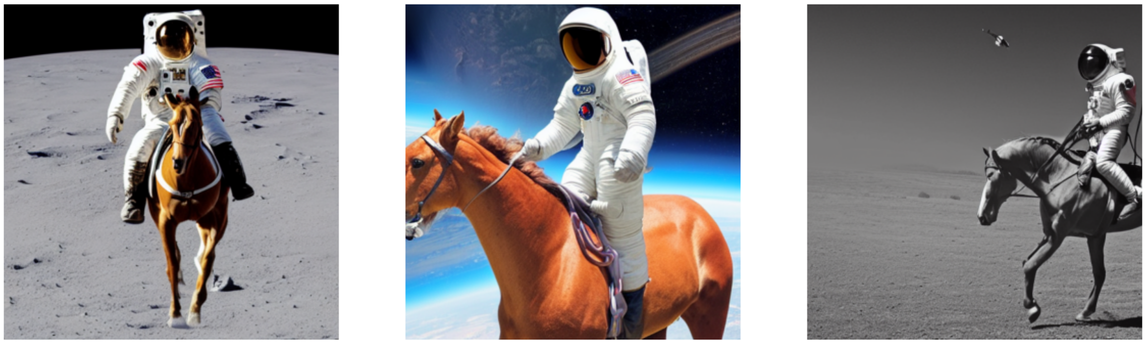


Figure 6.1: Images generated by a Stable Diffusion model given the prompt “photograph of an astronaut riding a horse”.

A key word in the research question is “generative”. By “generative” models, I mean the deep learning generative models, such as autoencoder, GAN, and variants of them, that can be used to generate images. In Chapter 2, I argued that these generative models could be considered as very prototypical example of imagery-based production systems. This argument could be illustrated by Figure 6.1, which includes the famous example of an astronaut riding a horse. This figure was generated by a stable diffusion model given the text prompt “photograph of an astronaut riding a horse”.

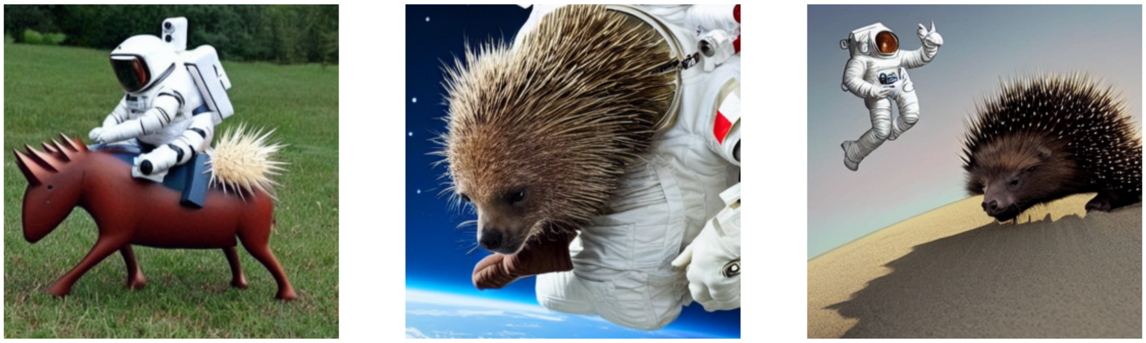


Figure 6.2: Images generated by a Stable Diffusion model given the prompt “An astronaut riding a porcupine”.

Despite the amazing progress of image generation models in recent years, I have to be quite cautious about my arguments here. That is the reason why I used the words “very prototypical example”. First, for the part that amazed me, assuming that the huge training sets, which are impossible for me to scrutinize, do not contain a picture or drawing of an astronaut riding horse, the generative models do exhibit certain level of imagery ability. In particular, it is highly likely that the model “knows” what horses look like, what astronauts look like, and what a man riding a horse looks like, but it never saw an astronaut riding a horse during training (according to our assumption). In this case, the model needs to imagine this unrealistic scenario in its “mind”. On one hand, it corresponds to the characteristic of human mental imagery that it can be unrealistic and full of creativity, as we mentioned in Chapter 3; on the other hand, it achieves a kind of nontrivial generalization, which is similar to the one discussed in Chapter 4 but in a less abstract context, i.e., the literal meaning of “riding” is less abstract than the concepts in previous chapters, such as arithmetic and analogy. The word “riding” is very much a spatial arrangement, in which one object is on another object, which is moving and taking the first object with it. To first show that how less abstract the generative model is, I would have to be vicious to the model and the astronaut, by giving the model a prompt “An astronaut riding a porcupine”. And it gave the output in Figure 6.2. In the left image of Figure 6.2, the animal does not really look like a porcupine; nor does the astronaut looks like a normal astronaut for his or her legs are kind of missing. In the middle one, there is even no astronaut, but a porcupine astronaut (I guess the model has a good sense of humor when it is not able to generate expected images). In the right one, there are indeed an astronaut and a porcupine, but neither of them is riding the other. I understand that no human, not even astronauts wearing the everything-proof space armor, would want to ride a porcupine, and thus there should not be too many, if any, images of humans riding porcupines in the training set. But the images of astronauts and porcupines must be representative enough in the training set. That is why I chose this prompt and the result implies that the generative model cannot even capture the literal meaning of the verb “ride”, let along the more abstract concepts.

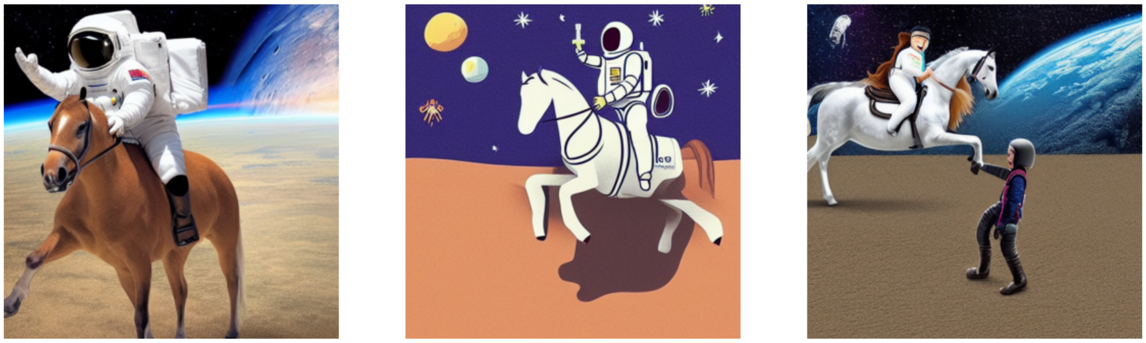


Figure 6.3: Images generated by a Stable Diffusion model given the prompt “A horse riding an astronaut”.

Another example of rigidness of the generative model is using the prompt "A horse riding an astronaut". This is the first prompt that came to my mind when I read some news advocating the Stable Diffusion model and its astronaut-riding-horse example. This example was also used by Gary Marcus in his criticism of the claim that these image generation models are significant steps toward Artificial General Intelligence (AGI). Figure 6.3 shows the output of the stable diffusion model given the prompt "A horse riding an astronaut". I cannot help relating this prompt to a famous school of Chinese martial arts—Taiji (also translated as Tai Chi)—which is characterized as using the minimum maneuvers to direct your enemy's force to fight himself. Taiji requires a deep understanding of human body and physics of forces to make the best maneuvers. Similarly, correctly responding to the prompts like "A horse riding an anstronaut" requires deep a understanding of natural languages (e.g., a small change to a sentence can completely change its meaning) and physics of the world. At the foundation of such understanding is the understanding of abstract concepts and how abstract concepts relate to each other and to real world physics. **Therefore, the second goal is to construct a learning model that can understand abstract concepts.**

6.3 Goal 3

As I indicated above, the current generative models are only very prototypical examples of learning models of computational imagery. They are very prototypical in two senses. The first one is the limitation of the current generative models discussed above, i.e., the incapability of nontrivial generalization and understanding and utilizing abstract concepts. The second is that it is generative but not productive. Although these two English words—generative and productive–have very similar meanings and can be used interchangeably in most situations, "From Generative to Productive" is not empty rhetoric in terms of AI systems.

In particular, being productive implies being generative; but being generative does not imply being productive if we take the traditional meaning of productive in the history of AI. The most typical example of being a productive AI system is production systems. The basic idea of production systems is a set of production rules which consist of preconditions and actions. Given initial states of a working memory, if the precondition of a production rule is satisfied, then the system can execute the action of the rule to update the states of the working memory. This procedure is repeated until the system cannot proceed because, for example, there is no satisfied rules or the goal state is achieved in the working memory.

In terms of formality, different production systems vary in the ways of how the working memory is matched to the rules and how the match rules are followed (e.g., executing all satisfied rules or one of the satisfied rules). In terms of content, i.e., the rules and all possible states of working memory, different production systems have different expressive powers; for example, some production systems are Turning complete, e.g., Post Production System, while many production systems, though useful, are not Turing complete. The production systems that are relevant to the research question are a subset of production systems that have variables in their rules and thus allows to construct formal languages, for example, the formal languages reducible to some Chomsky-typed formal languages. There are two obvious differences between these production systems and current deep learning image generation models:

- Iterative versus One-Shot: the production system needs to repeatedly applies its rules, generating more and more valid results, until the goal is achieved or it cannot proceed; popular image generation deep learning models, such as GAN and VAE, generate the images in an one-shot way. This one-shot way is not suitable for visual abstract reasoning which usually requires a long reasoning chain.
- Entangled versus Disentangled: There are indeed some generative deep learning models that involve an iterative process, for example, the diffusion model we used to generate images in this chapter. But these iterative processes, like the reverse-diffusion process in diffusion models, do not apply a distinct rule at each step as production systems does. In other words, the operations in a reverse-diffusion process are all entangled with each other and one cannot tell what specific effects are achieved at each step. In image generation, disentangled latent variables are often desired to achieve a better generative effect. Similarly, in visual abstract reasoning, the disentanglement between rules applied at different steps is not only desirable but necessary. The disentanglement of visual abstract reasoning can be considered as a temporal or logical disentanglement at a higher level than the latent variables in the current generative models, because abstract rules about attributes are disentangled in the former while attributes are disentangled in the latter.

The research question is “how generative models can be extended to imagery-based production systems”. The idea of such a production system is relatively clear now: the states in working memory are images and the production rules are operations on images; and the production rules are learned rather than handcrafted. But given these two differences between generative models and production systems, it seems quite difficult to extend generative models to production systems, let along imagery-based production systems. But in our case of visual abstract reasoning, being imagery-based is an advantage for us to build such production system compared to other tasks, e.g., text-to-image generation, because:

- the images in visual abstract reasoning are much simpler than the images that current generative need to generate;
- clearly distinct and disjoint rules in visual abstract reasoning problems allows the disentanglement of production rules of production system;

- applying different rules results in various images and thus allows us to fully utilize the strong generative power of current generative models.

In a word, visual abstract reasoning tasks provide proper data for us to build the imagery-based production system, especially to learn the abstract concepts. **Given the differences in the execution between generative models and production systems, the only issue, which is also the last goal in this chapter, is how to modify the architectures of generative models in order to simulate the execution of production systems.**

6.4 Summary

In this chapter, I described the context of the research question—"how generative models can be extended to imagery-based production systems that is able to produce a flexible reasoning process in computational imagery"—and discussed the motivation and purposes of this research questions. In particular, I decomposed this research question into three goals:

- **Goal 1**: construct a flexible and robust learning model that simulates the behavior of imagery-based models on RPM and other visual abstract reasoning tasks.
- **Goal 2**: construct a learning model that can understand abstract concepts.
- **Goal 3**: modify the architectures of generative model in order to simulate the execution of production systems.

These three goals are by no mean independent. On the contrary, they are targeting different aspects of the same imagery-based production system and realizing any one of them depends on realizing the other two.

References

An, J. and Cho, S. (2020). Hierarchical transformer encoder with structured representation for abstract reasoning. *IEEE Access*, 8:200229–200236.

Arendasy, M. (2002). Geomgen—ein itemgenerator für matrizentestaufgaben. *Wien: Eigenverlag*.

Arendasy, M. and Sommer, M. (2005). The effect of different types of perceptual manipulations on the dimensionality of automatically generated figural matrices. *Intelligence*, 33(3):307–324.

Barrett, D., Hill, F., Santoro, A., Morcos, A., and Lillicrap, T. (2018). Measuring abstract reasoning in neural networks. In *International conference on machine learning*, pages 511–520. PMLR.

Barsalou, L. W. et al. (1999). Perceptual symbol systems. *Behavioral and Brain Sciences*, 22(4):577–660.

Benny, Y., Pekar, N., and Wolf, L. (2021). Scale-localized abstract reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12557–12565.

Bethell-Fox, C. E., Lohman, D. F., and Snow, R. E. (1984). Adaptive reasoning: Componential and eye movement analysis of geometric analogy performance. *Intelligence*, 8(3):205–238.

Blum, D. and Holling, H. (2018). Automatic generation of figural analogies with the imak package. *Frontiers in psychology*, 9:1286.

Bohan, A. and O'Donoghue, D. (2000). Ludi: A model for geometric analogies using attribute matching. In *11th Irish Artificial Intelligence and Cognitive Science Conference (AICS 2000))*, pages 110–119.

Bringsjord, S. (2011). Psychometric artificial intelligence. *Journal of Experimental & Theoretical Artificial Intelligence*, 23(3):271–277.

Bringsjord, S. and Schimanski, B. (2003). What is artificial intelligence? psychometric ai as an answer. In *IJCAI*, pages 887–893. Citeseer.

Burke, H. R. (1958). Raven's progressive matrices: A review and critical evaluation. *The Journal of Genetic Psychology*, 93(2):199–228.

Carpenter, P. A., Just, M. A., and Shell, P. (1990). What one intelligence test measures: a theoretical account of the processing in the raven progressive matrices test. *Psychological review*, 97(3):404.

Cattell, R. B. (1941). Some theoretical issues in adult intelligence testing. *Psychological Bulletin*, 38(7):592.

Cattell, R. B. (1943). The measurement of adult intelligence. *Psychological bulletin*, 40(3):153.

Cattell, R. B. (1950). *Handbook for the individual or group culture fair intelligence test*. Institute for Personality and Ability Testing.

Cattell, R. B. (1963). Theory of fluid and crystallized intelligence: A critical experiment. *Journal of educational psychology*, 54(1):1.

Cattell, R. B. (1987). *Intelligence: Its structure, growth and action*. Elsevier.

Cirillo, S. and Ström, V. (2010). An anthropomorphic solver for raven's progressive matrices. Master's thesis, Chalmers University of Technology.

Das, J. P., Naglieri, J. A., and Kirby, J. R. (1994). *Assessment of cognitive processes: The PASS theory of intelligence.* Allyn & Bacon.

Davies, J. and Goel, A. K. (2001). Visual analogy in problem solving. In *IJCAI*, pages 377–384.

DeShon, R. P., Chan, D., and Weissbein, D. A. (1995). Verbal overshadowing effects on Raven's Advanced Progressive Matrices: Evidence for multidimensional performance determinants. *Intelligence*, 21(2):135–155.

Detterman, D. K. (2011). A challenge to watson.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dorolle, M. (1949). *Le Raisonnement Par Analogie*. PUF, Paris.

Embretson, S. E. (1995). The role of working memory capacity and general control processes in intelligence. *Intelligence*, 20(2):169–189.

Embretson, S. E. (1998). A cognitive design system approach to generating valid tests: Application to abstract reasoning. *Psychological methods*, 3(3):380.

Embretson, S. E. (2004). Measuring human intelligence with artificial intelligence: Adaptive item generation. In *Georgia Tech School of Psychology Faculty Publications*. Georgia Institute of Technology.

Eslami, S., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Hinton, G. E., et al. (2016). Attend, infer, repeat: Fast scene understanding with generative models. *Advances in Neural Information Processing Systems*, 29.

Evans, T. G. (1964). A program for the solution of a class of geometric-analogy intelligence-test questions. Technical report, AIR FORCE CAMBRIDGE RESEARCH LABS LG HANSCOM FIELD MASS.

Freund, P. and Holling, H. (2011a). How to get really smart: Modeling retest and training effects in ability testing using computer-generated figural matrix items. *Intelligence*, 39(4):233–243.

Freund, P. and Holling, H. (2011b). Retest effects in matrix test performance: Differential impact of predictors at different hierarchy levels in an educational setting. *Learning & ind. diff.*, 21(5):597–601.

Freund, P. and Holling, H. (2011c). Who wants to take an intelligence test? personality and achievement motivation in the context of ability testing. *Personality & Ind. Diff.*, 50(5):723–728.

Freund, P. A., Hofer, S., and Holling, H. (2008). Explaining and controlling for the psychometric properties of computer-generated figural matrix items. *Applied Psychological Measurement*, 32(3):195–210.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive science*, 7(2):155–170.

Gierl, M. J., Lai, H., and Turner, S. R. (2012). Using automatic item generation to create multiple-choice test items. *Medical education*, 46(8):757–765.

Hahne, L., Lüddecke, T., Wörgötter, F., and Kappel, D. (2019). Attention on abstract visual reasoning. *arXiv preprint arXiv:1911.05990*.

Hayes, T. L. and Kanan, C. (2021). Selective replay enhances learning in online continual analogical reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3502–3512.

He, W., Ren, J., and Bai, R. (2021a). One-shot visual reasoning on rpms with an application to video frame prediction. *arXiv preprint arXiv:2111.12301*.

He, W., Ren, J., and Bai, R. (2021b). Two-stage rule-induction visual reasoning on rpms with an application to video prediction. *arXiv preprint arXiv:2111.12301*.

Hernández-Orallo, J., Martínez-Plumed, F., Schmid, U., Siebers, M., and Dowe, D. L. (2016). Computer models solving intelligence test problems: Progress and implications. *Artificial Intelligence*, 230:74–107.

Hersche, M., Zeqiri, M., Benini, L., Sebastian, A., and Rahimi, A. (2022). A neuro-vector-symbolic architecture for solving raven's progressive matrices. *arXiv preprint arXiv:2203.04571*.

Hersche, M., Zeqiri, M., Benini, L., Sebastian, A., and Rahimi, A. (2023). A neuro-vector-symbolic architecture for solving raven's progressive matrices. *Nature Machine Intelligence*.

Hill, F., Santoro, A., Barrett, D. G., Morcos, A. S., and Lillicrap, T. (2019). Learning to make analogies by contrasting abstract relational structure. *arXiv preprint arXiv:1902.00120*.

Hofer, S. (2004). Matrixdeveloper [unpublished computer software]. Unpublished software.

Hofstadter, D. R. (1979). *Gödel, Escher, Bach : an eternal golden braid / Douglas R. Hofstadter.* Basic Books, New York.

Hofstadter, D. R. (2001). Analogy as the core of cognition. In Gentner, D., Holyoak, K. J., and Kokinov, B. N., editors, *The Analogical Mind: Perspectives From Cognitive Science*, chapter 15, pages 499–538. The MIT Press, Cambridge, MA.

Hofstadter, D. R. and Sander, E. (2013). *Surfaces and essences: Analogy as the fuel and fire of thinking*. Basic books.

Hornke, L. F. and Habon, M. W. (1986). Rule-based item bank construction and evaluation within the linear logistic framework. *Applied psychological measurement*, 10(4):369–380.

Hoshen, D. and Werman, M. (2017). Iq of neural networks. *arXiv preprint arXiv:1710.01692*.

Hu, S., Ma, Y., Liu, X., Wei, Y., and Bai, S. (2021). Stratified rule-aware network for abstract visual reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(2):1567–1574.

Hua, T. and Kunda, M. (2019). Modeling gestalt visual reasoning on the raven's progressive matrices intelligence test using generative image inpainting techniques. *arXiv preprint arXiv:1911.07736*.

Hunt, E. (1974). Quote the raven? nevermore. In *Knowledge and cognition*. Lawrence Erlbaum.

Hunt, E. (2010). *Human intelligence*. Cambridge University Press.

Jahrens, M. and Martinetz, T. (2018). Multi-layer relation networks. *arXiv preprint arXiv:1811.01838*.

Jahrens, M. and Martinetz, T. (2019). Multi-layer relation networks for relational reasoning. In *Proceedings of the 2nd International Conference on Applications of Intelligent Systems*, pages 1–5.

Jahrens, M. and Martinetz, T. (2020). Solving raven's progressive matrices with multi-layer relation networks. In *International Joint Conference on Neural Networks*. IEEE.

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. (2020). Supervised contrastive learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673. Curran Associates, Inc.

Kiat, N. Q. W., Wang, D., and Jamnik, M. (2020). Pairwise relations discriminator for unsupervised raven's progressive matrices. *arXiv preprint arXiv:2011.01306*.

Kim, Y., Shin, J., Yang, E., and Hwang, S. J. (2020). Few-shot visual reasoning with meta-analogical contrastive learning. *Advances in Neural Information Processing Systems*, 33:16846–16856.

Kosslyn, S. M., Thompson, W. L., and Ganis, G. (2006). *The case for mental imagery*. Oxford University Press, New York, NY, 1st. edition.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA. Curran Associates Inc.

Kunda, M. (2013). *Visual problem solving in autism, psychometrics, and AI: The case of the Raven's progressive matrices intelligence test*. PhD thesis, Georgia Institute of Technology, Atlanta, GA.

Kunda, M. (2015). Computational mental imagery, and visual mechanisms for maintaining a goal-subgoal hierarchy. In *Proceedings of the Third Annual Conference on Advances in Cognitive Systems*, Atlanta, GA.

Kunda, M., McGreggor, K., and Goel, A. (2009). Addressing the raven's progressive matrices test of "general" intelligence. In *2009 AAAI Fall Symposium Series*.

Kunda, M., McGreggor, K., and Goel, A. (2010). Taking a look (literally!) at the raven's intelligence test: Two visual solution strategies. In *Proceedings of the Annual Meeting of the Cognitive Science Society*.

Kunda, M., McGreggor, K., and Goel, A. K. (2013). A computational model for solving problems from the raven's progressive matrices intelligence test using iconic visual representations. *Cognitive Systems Research*, 22:47–66.

Kunda, M., Soulières, I., Rozga, A., and Goel, A. K. (2016). Error patterns on the Raven's Standard Progressive Matrices test. *Intelligence*, 59:181–198.

Laird, J. E., Gluck, K., Anderson, J., Forbus, K. D., Jenkins, O. C., Lebiere, C., Salvucci, D., Scheutz, M., Thomaz, A., Trafton, G., et al. (2017). Interactive task learning. *IEEE Intelligent Systems*, 32(4):6–21.

Lovett, A. and Forbus, K. (2017). Modeling visual problem solving as analogical reasoning. *Psychological Review*, 124(1):60.

Lovett, A., Forbus, K., and Usher, J. (2007). Analogy with qualitative spatial representations can simulate solving raven's progressive matrices. In *Proceedings of the Annual Meeting of the Cognitive Science Society*.

Lovett, A., Tomai, E., Forbus, K., and Usher, J. (2009). Solving geometric analogy problems through two-stage analogical mapping. *Cognitive science*, 33(7):1192–1231.

Małkiński, M. and Mańdziuk, J. (2020). Multi-label contrastive learning for abstract visual reasoning. *arXiv preprint arXiv:2012.01944*.

Mańdziuk, J. and Żychowski, A. (2019). Deepiq: A human-inspired ai system for solving iq test problems. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Matzen, L. E., Benz, Z. O., Dixon, K. R., Posey, J., Kroger, J. K., and Speed, A. E. (2010). Recreating raven's: Software for systematically generating large numbers of raven-like matrix problems with normed properties. *Behavior research methods*, 42(2):525–541.

McGreggor, K., Kunda, M., and Goel, A. (2014). Fractals and ravens. *Artificial Intelligence*, 215:1–23.

Mekik, C. S., Sun, R., and Dai, D. Y. (2017). Deep learning of raven's matrices. In *Proceedings of the fifth Annual Conference on Advances in Cognitive Systems*.

Mekik, C. S., Sun, R., and Dai, D. Y. (2018). Similarity-based reasoning, raven's matrices, and general intelligence. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 1576–1582. AAAI Press.

Meo, M., Roberts, M. J., and Marucci, F. S. (2007). Element salience as a predictor of item difficulty for raven's progressive matrices. *Intelligence*, 35(4):359–368.

Michelson, J., Palmer, J. H., Dasari, A., and Kunda, M. (2019). Learning spatially structured image transformations using planar neural networks. *arXiv preprint arXiv:1912.01553*.

Mitchell, M. (1993). *Analogy-making as perception: A computer model*. Mit Press.

Mulholland, T. M., Pellegrino, J. W., and Glaser, R. (1980). Components of geometric analogy solution. *Cognitive psychology*, 12(2):252–284.

Naglieri, J. A., Das, J. P., and Goldstein, S. (2014). *Cognitive assessment system: Interpretive and technical manual*. Pro-ed.

Pekar, N., Benny, Y., and Wolf, L. (2020). Generating correct answers for progressive matrices intelligence tests. *Advances in Neural Information Processing Systems*, 33:7390–7400.

Prabhakaran, V., Smith, J. A., Desmond, J. E., Glover, G. H., and Gabrieli, J. D. (1997). Neural substrates of fluid reasoning: An fMRI study of neocortical activation during performance of the Raven's Progressive Matrices test. *Cognitive Psychology*, 33(1):43–63.

Prade, H. and Richard, G. (2009). Analogy, paralogy and reverse analogy: Postulates and inferences. In *Annual Conference on Artificial Intelligence*, pages 306–314. Springer.

Prade, H. and Richard, G. (2010). Reasoning with logical proportions. In *Twelfth International Conference on the Principles of Knowledge Representation and Reasoning*.

Prade, H. and Richard, G. (2013). From analogical proportion to logical proportions. *Logica Universalis*, 7(4):441–505.

Primi, R. (2001). Complexity of geometric inductive reasoning tasks: Contribution to the understanding of fluid intelligence. *Intelligence*, 30(1):41–70.

Ragni, M. and Neubert, S. (2012). Solving raven's iq-tests: an ai and cognitive modeling approach. In *ECAI 2012*, pages 666–671. IOS Press.

Ragni, M. and Neubert, S. (2014). Analyzing raven's intelligence test: Cognitive model, demand, and complexity. In *Computational Approaches to Analogical Reasoning: Current Trends*, pages 351–370. Springer.

Ragni, M., Schleipen, S., and Steffenhagen, F. (2007). Solving proportional analogies: A computational model. *Analogies: Integrating Multiple Cognitive Abilities*, 5:51.

Rahaman, N., Gondal, M. W., Joshi, S., Gehler, P., Bengio, Y., Locatello, F., and Schölkopf, B. (2021). Dynamic inference with neural interpreters. *Advances in Neural Information Processing Systems*, 34.

Rasmussen, D. and Eliasmith, C. (2011). A neural model of rule generation in inductive reasoning. *Topics in Cognitive Science*, 3(1):140–153.

Raudies, F. and Hasselmo, M. E. (2017). A model of symbolic processing in raven's progressive matrices. *Biologically Inspired Cognitive Architectures*, 21:47–58.

Raven, J. (2000). The raven's progressive matrices: change and stability over culture and time. *Cognitive psychology*, 41(1):1–48.

Raven, J. (2008). The raven progressive matrices tests: their theoretical basis and measurement model. *Uses and abuses of Intelligence. Studies advancing Spearman and Raven's quest for non-arbitrary metrics*, pages 17–68.

Raven, J., Raven, J. C., and Court, J. H. (1998). *Manual for Raven's Progressive Matrices and Vocabulary Scales*. Harcourt Assessment, San Antonio, TX.

Raven, J. C. (1936). Mental tests used in genetic studies: The performance of related individuals on tests mainly educative and mainly reproductive. *Unpublished master's thesis, University of London*.

Raven, J. C. (1941). Standardization of progressive matrices, 1938. *British Journal of Medical Psychology*, 19(1):137–150.

Roid, G. H. and Miller, L. J. (1997). *Leiter International Performance Scale-Revised: Examiners Manual*. Stoelting.

Sahu, P., Basioti, K., and Pavlovic, V. (2023). Savir-t: Spatially attentive visual reasoning with transformers. In Amini, M.-R., Canu, S., Fischer, A., Guns, T., Kralj Novak, P., and Tsoumakas, G., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 460–476, Cham. Springer Nature Switzerland.

Schmid, U. and Kitzelmann, E. (2011). Inductive rule learning on the knowledge level. *Cognitive Systems Research*, 12(3-4):237–248.

Schwering, A., Krumnack, U., Kuhnberger, K.-U., and Gust, H. (2007). Using gestalt principles to compute analogies of geometric figures. In *Proceedings of the Annual Meeting of the Cognitive Science Society*.

Sekh, A. A., Dogra, D. P., Kar, S., Roy, P. P., and Prasad, D. K. (2020). Can we automate diagrammatic reasoning? *Pattern Recognition*, 106:107412.

Shegheva, S. (2018). A computational model for solving raven's progressive matrices intelligence test. Master's thesis, Georgia Institute of Technology, Atlanta, GA.

Shekhar, S. and Taylor, G. W. (2021). Neural structure mapping for learning abstract visual analogies. In *SVRHM 2021 Workshop@ NeurIPS*.

Shi, F., Li, B., and Xue, X. (2021). Raven's progressive matrices completion with latent gaussian process priors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9612–9620.

Sinha, I., Webb, T. W., and Cohen, J. D. (2020). A memory-augmented neural network model of abstract rule learning. *arXiv preprint arXiv:2012.07172*.

Snow, R. E. (1980). Aptitude processes. *Aptitude, Learning, and Instruction: Cognitive Process Analyses of Aptitude; Snow, RE, Federico, P.-A., Montague, WE, Eds*, pages 27–63.

Snow, R. E., Kyllonen, P. C., Marshalek, B., et al. (1984). The topography of ability and learning correlations. *Advances in the psychology of human intelligence*, 2(S 47):103.

Spearman, C. (1923). *The nature of "intelligence" and the principles of cognition*. Macmillan, London, UK.

Spratley, S., Ehinger, K., and Miller, T. (2020). 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*, pages 601–616. Springer.

Stabinger, S., Peer, D., Piater, J., and Rodríguez-Sánchez, A. (2021). Evaluating the progress of deep learning for visual relational concepts. *Journal of Vision*, 21(11):8–8.

Steenbrugge, X., Leroux, S., Verbelen, T., and Dhoedt, B. (2018). Improving generalization for abstract reasoning tasks using disentangled feature representations. *arXiv preprint arXiv:1811.04784*.

Sternberg, R. J. (1977). *Intelligence, information processing, and analogical reasoning: The componential analysis of human abilities.* Lawrence Erlbaum.

Strannegård, C., Cirillo, S., and Ström, V. (2013). An anthropomorphic method for progressive matrix problems. *Cognitive Systems Research*, 22:35–46.

Yang, Yuan, Sanyal, D., Michelson, J., Ainooson, J., and Kunda, M. (2022). An end-to-end imagery-based modeling of solving geometric analogy problems. In *Proceedings of the Annual Meeting of the Cognitive Science Society*.

Thagard, P. (1996). *Mind: Introduction to cognitive science*, volume 4. MIT press Cambridge, MA.

Tomai, E., , Lovett, Andrew Forbus, K. D., and Usher, J. (2005). A structure mapping model for solving geometric analogy problems. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 27, pages 2190—2195.

van Steenkiste, S., Locatello, F., Schmidhuber, J., and Bachem, O. (2019). Are disentangled representations helpful for abstract visual reasoning? *arXiv preprint arXiv:1905.12506*.

Wang, D., Jamnik, M., and Lio, P. (2020). Abstract diagrammatic reasoning with multiplex graph networks. *arXiv preprint arXiv:2006.11197*.

Wang, K. and Su, Z. (2015). Automatic generation of raven's progressive matrices. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Webb, T., Dulberg, Z., Frankland, S., Petrov, A., O'Reilly, R., and Cohen, J. (2020a). Learning representations that support extrapolation. In *International conference on machine learning*, pages 10136–10146. PMLR.

Webb, T. W., Sinha, I., and Cohen, J. (2020b). Emergent symbols through binding in external memory. In *International Conference on Learning Representations*.

Wechsler, D., Coalson, D. L., and Faiford, S. E. (2008). *Wechsler Adult Intelligence Scale—Fourth Edition Technical and Interpretive Manual*. Pearson.

Wei, Q., Chen, D., and Yuan, B. (2023). Multi-viewpoint and multi-evaluation with felicitous inductive bias boost machine abstract reasoning ability.

Wu, Y., Dong, H., Grosse, R., and Ba, J. (2021). The scattering compositional learner: Discovering objects, attributes, relationships in analogical reasoning. *arXiv preprint arXiv:2007.04212*.

Xu, J., Vaidya, T., Wu, Y., Chandra, S., Lai, Z., and Chong, K. F. E. (2023). Abstract visual reasoning: An algebraic approach for solving raven's progressive matrices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6715–6724.

Yang, Y., McGreggor, K., and Kunda, M. (2020). 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)*.

Yu, S., Mo, S., Ahn, S., and Shin, J. (2021). Abstract reasoning via logic-guided generation. *arXiv preprint arXiv:2107.10493*.

Zhang, C., Gao, F., Jia, B., Zhu, Y., and Zhu, S.-C. (2019a). Raven: A dataset for relational and analogical visual reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5317–5327.

Zhang, C., Jia, B., Gao, F., Zhu, Y., Lu, H., and Zhu, S.-C. (2019b). Learning perceptual inference by contrasting. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Zhang, C., Jia, B., Zhu, S.-C., and Zhu, Y. (2021). Abstract spatial-temporal reasoning via probabilistic abduction and execution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9736–9746.

Zhang, C., Xie, S., Jia, B., Wu, Y. N., Zhu, S.-C., and Zhu, Y. (2022). Learning algebraic representation for systematic generalization in abstract reasoning. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Zhang, C., Xie, S., Jia, B., Zhu, Y., Wu, Y. N., and Zhu, S.-C. (2020). Learning algebraic representation for abstract spatial-temporal reasoning. Rejected by ICLR 2021.

Zheng, K., Zha, Z.-J., and Wei, W. (2019). Abstract reasoning with distracting features. *Advances in Neural Information Processing Systems*, 32.

Zhuo, T., Huang, Q., and Kankanhalli, M. (2021). Unsupervised abstract reasoning for raven's problem matrices. *IEEE Transactions on Image Processing*, 30:8332–8341.

Zhuo, T. and Kankanhalli, M. (2020). Solving raven's progressive matrices with neural networks. *arXiv preprint arXiv:2002.01646*.

Zhuo, T. and Kankanhalli, M. (2021). Effective abstract reasoning with dual-contrast network. In *International Conference on Learning Representations*.