Multi-Viewpoint and Multi-Evaluation With Felicitous Inductive Bias Boost Machine Abstract Reasoning Ability

Qinglai Wei, Senior Member, IEEE, Diancheng Chen, and Beiming Yuan

Abstract—Great efforts have been made to investigate AI’s ability in abstract reasoning, along with the proposal of various versions of RAVEN’s progressive matrices (RPM) as benchmarks. Previous studies suggest that, even after extensive training, neural networks may still struggle to make decisive decisions regarding RPM problems without sophisticated designs or additional semantic information in the form of meta-data. Through comprehensive experiments, we demonstrate that neural networks endowed with appropriate inductive biases, either intentionally designed or fortuitously matched, can efficiently solve RPM problems without the need for extra meta-data augmentation. Our work also reveals the importance of employing a multi-viewpoint with multi-evaluation approach as a key learning strategy for successful reasoning. Nevertheless, we acknowledge the unique role of metadata by demonstrating that a pre-training model supervised by meta-data leads to an RPM solver with improved performance. Codes are available in: https://github.com/QinglaiWeiCASIA/RavenSolver.

Index Terms—Abstract reasoning, Raven’s progressive matrices, inductive bias, pre-training model.

I. INTRODUCTION

FROM expert systems with elaborately designed rules to the renaissance of neural networks, AI practitioners never cease to work on machine intelligence, striving to make it equivalent to human intelligence. The remarkable success of machine learning in domains such as visual perception [1], [2], natural language processing [3], [4], [5], and generative models [6], [7], [8] has captivated researchers, prompting them to delve into the reasoning ability of AI. Noteworthy endeavors encompass, but not limited to, visual question answering [9], [10], versatile application of language models [11], [12], [13], and abstract reasoning problems [14], [15], [16]. In this context, we explore the RPM problem, originally developed as an IQ test [17], and recently serves as a benchmark for evaluating AI’s abstract reasoning prowess.

Fig. 1 illustrates two RPM problems. Without loss of generality, RPM problems are formalized in three steps. Firstly, rule samples are selected from a predefined rule set to determine the changing patterns of visual attributes. Common rules include, but not limited to, arithmetic operations, set operations, and logic operations. Secondly, based on the selected rules, proper values are assigned to all visual attributes. Some visual attributes may act as distractors, with their values randomly changing. Finally, images are rendered using the assigned visual attribute values. An instantiated RPM problem consists of a context and an answer pool. The context is assigned visual attribute values. An instantiated RPM problem is a 3 × 3 image matrix with the image in the lower right corner missing. The answer pool contains 8 images for selection, and test-takers are expected to choose the most suitable image from the answer pool to complete the matrix, ensuring compatibility with the internal rules.

Fig. 1. Demonstrations of RPM problems. These two RPM questions are snapshots from I-RAVEN and PGM dataset, respectively.
To achieve satisfactory reasoning accuracy in RPM problems, models are expected to extract relevant visual attributes for downstream tasks while inferring the underlying rules. Traditional perception neural networks, consisting only of perception modules, are inadequate for solving RPM problems [18], [19].

In this work, we propose end-to-end solutions for RPM problems. Several key points should be considered when developing a black-box RPM solver. These include distinct modularization to mimic complete perception and reasoning processes, encapsulation of two essential RPM characteristics, namely permutation invariance and transpose invariance, into the inductive bias design, and the implementation of a multi-viewpoint and multi-evaluation strategy. To achieve distinct modularization, a clear boundary and effective cooperation between the feature extraction module and the reasoning module are necessary. Each module should fulfill its role effectively, as adding a new module without proper integration would simply increase the depth of the neural network. To address this, we incorporate available inductive bias into the reasoning module, making it aware of the permutation invariance and transpose invariance characteristics of RPM problems. Furthermore, RPM problems involve various visual attributes and rules, resulting in numerous attribute-rule combinations. To tackle this, we equip the feature extraction module with a multi-viewpoint strategy and the reasoning module with a multi-evaluation strategy. This enables the model to tackle RPM problems from different perspectives. These aforementioned details are sufficient for building an RPM solver with high reasoning accuracy. Additionally, we train an auxiliary model to predict the natural language descriptions of the rules in RPM problems. By using this auxiliary model as a pre-training model, we achieve faster training, higher reasoning accuracy, and improved generalization ability for the RPM solver.

The results of our work are promising and intriguing in several ways. Firstly, we demonstrate that models with proper inductive biases, whether based on convolutional neural networks (CNN) or vision transformers (ViT [20]), achieve competitive reasoning accuracies without relying on any meta-data. Secondly, we show that the multi-viewpoint and multi-evaluation strategy is key to developing transformer-based RPM solver. Thirdly, through experimental evidence, we show that the rules captured by the neural network differ from the predefined rules. Finally, we discover that a model capable of predicting the rules in RPM problems can serve effectively as a pre-training model for the RPM solver, leading to faster training speed, higher reasoning accuracy and better generalization performance.

II. RELATED WORK

A. RPM Dataset

We examine three datasets in this study: RAVEN [18], I-RAVEN [31], and PGM [19]. While these datasets share a common construction guideline, they also exhibit subtle differences.

The RAVEN dataset comprises 7 distinct configurations with varying levels of difficulty. The simplest configuration, referred to as ‘Center’, consists of problem matrices with only one entity per panel. In contrast, more challenging configurations like ‘3 × 3 Grid’ contain up to nine entities in each panel. Test-takers are tasked with extracting visual attributes, observing the row-wise changing patterns, summarizing the rules governing these changes, and subsequently selecting the appropriate choice to complete the problem matrix. The most demanding configuration, ‘O-IG’, as shown in Fig. 1(a), requires test-takers to categorize entities in each panel into two groups, each following a specific set of rules, and perform reasoning accordingly. Some studies have indicated that the answer generation process in RAVEN may lead neural network solvers to discover shortcuts instead of identifying underlying rules [24], [31]. To address this issue, additional datasets such as I-RAVEN and RAVEN-Fair have been proposed, which refine the answer generation strategies [24], [31].

Fig. 1(b) presents an example from the PGM dataset, where panels can contain entities in the foreground and lines in the background. To solve PGM, test-takers need to observe the changes of visual attributes both row-wise and column-wise simultaneously, summarize potential rules for the foreground and background separately, and then complete the reasoning task accordingly.

On average, RAVEN and I-RAVEN datasets have more rules per question compared to PGM (6.29 vs. 1.37 [18]). Additionally, RAVEN and I-RAVEN include two fixed visual attributes as distractors, while PGM allows any visual attribute to act as a distractor. Notably, the rules in RAVEN and I-RAVEN are encoded row-wise, whereas in PGM, test-takers must consider both row-wise and column-wise information to summarize the rules effectively.

B. RPM Solvers

The literature on RPM solvers has expanded rapidly in recent years, and we can broadly categorize them into two categories. The first category is end-to-end solvers, which constitute the majority of previous works. The second category leverages symbolic AI techniques to achieve results beyond reasoning accuracies, such as interpretability.

End-to-end models primarily focus on improving reasoning accuracy in RPM problems. Early works have shown that conventional visual models struggle to solve RPM problems, and the introduction of extra labels containing structural or rule information has shown some improvement [18], [19]. In LEN [21], researchers highlight the elimination of distracting information as the main challenge in solving RPM problems. CoPINet [22] and DCNet [23] employ contrastive learning techniques in reasoning. However, the aforementioned models either rely on short-cut solutions or demonstrate competitive performance exclusively within a single dataset. MRNet [24] demonstrates that retrieving features from different CNN blocks connected sequentially helps capture multiple visual attributes simultaneously. It is also the first work to report that extra meta-data can have detrimental effect on
network performance. SCL [25] employs tensor scattering to make each scattered part attend to specific visual attributes or rules. SAVIR-T [26] extracts intra-image information and inter-image relations to enhance reasoning ability. The last three state-of-the-art models coincidently tend to process raw images from multiple viewpoints.

Symbolic AI-powered methods have demonstrated higher reasoning accuracies and stronger model interpretability in RPM problem solving. In PrAE [27], a neural symbolic system employs probabilistic abduction and execution to generate an answer image. ALANS [28], which eliminates the prior knowledge requirement present in PrAE, achieves superior generalization ability and outperforms monolithic end-to-end models. NVSA [29] utilizes holographic vectorized representations to construct a neural-symbolic model. Compared to end-to-end models, these models generally demand more prior information.

In our work, we draw upon successful experiences from previous models. Specifically, we adopt the encoder architecture employed in MRNet for our CNN-based models. However, what sets our models apart is the active expressiveness of the inductive bias, multi-viewpoint with multi-evaluation strategies, and a multi-modal pre-training scheme, which contribute to improved overall performance compared to previous models.

C. CLIP

CLIP is a multi-modal pre-training neural network that simultaneously trains an image encoder and a natural language encoder. It achieves this by maximizing the similarity between visual representations and natural language embeddings of positive sample pairs, while minimizing the similarity in negative sample pairs. This training process enables CLIP to learn high-quality visual representations, allowing for zero-shot transfer to downstream tasks [30].

In our study, we discover that our model generates unaligned rule representations for RPM problem matrices governed by the same rule. To address this issue, we train a CLIP model to align the rule representation of each RPM problem matrix with the embedding of the corresponding rule’s natural language description, and introduce a mask scheme to filter useless information. We then utilize the visual end of the trained CLIP model as a pre-trained perception module for our RPM solver. The integration of CLIP-wise pre-training leads to a new model as a pre-trained perception module for our RPM solver.

We demonstrate that RS-CNN achieves accurate reasoning on RAVEN and I-RAVEN datasets through proper inductive design. On the other hand, the inductive bias of RS-TRAN naturally adapts to all RPM problems without the need for additional design. Furthermore, we emphasize the remarkable improvement in the reasoning ability of RS-TRAN, which is achieved through the adoption of a multi-viewpoint with multi-evaluation mechanism. Additionally, we discuss the potential issues with the original meta-data and introduce RS-TRAN-CLIP, which is a masked CLIP-based pre-training model specifically designed for RS-TRAN.

A. RS-CNN

RS-CNN consists of a perception module and a reasoning module. The perception module captures various visual attributes simultaneously. We adopt the architecture of the multi-scale encoder from MRNet [24], where different convolutional blocks attend to different visual attributes, as shown in Fig. 2.

For the images in a problem matrix \( \{X^i_1\}_{i=1}^{8} \) and the corresponding answer candidates \( \{X^i_{ac}\}_{i=1}^{8} \), the perception module of RS-CNN generates representation triplets \( \{e_p^i, e_p^m, e_i^m\}_{i=1}^{8} \) and \( \{e_{ac}^i, e_{ac}^m, e_{ac}^h\}_{i=1}^{8} \), where \( h, m, l \) refer to the convolutional blocks \( E_H, E_M, E_L \) in Fig. 2, respectively.

Fig. 3 illustrates the reasoning module, which is part of the aforementioned downstream task \( T \). This module receives the outputs from each convolutional block of the perception module \( (E_H, E_M, E_L) \) as inputs. Let’s consider \( E_H \) as an example. The representations of images within each problem matrix are concatenated row-wise and passed through an information fusion module (IFM) consisting of convolutional blocks. This process allows us to obtain the row-wise aggregated representations:

\[
\text{row}1: IFM[Concatenate\{e^1_p, e^2_p, e^3_p\}] = e^1_h \\
\text{row}2: IFM[Concatenate\{e^4_p, e^5_p, e^6_p\}] = e^2_h \\
\text{row}3: IFM[Concatenate\{e^7_p, e^8_p, e^9_p\}] = e^3_h
\]

where \( e_{ac}^i = \{e_{ac}^i\}_{i=1}^{8} \).

Note that rules can only be determined after observing at least two rows of the problem matrix [26]. Take the aggregated representations of \( E_H \) as example, every two rows of aggregated information are paired up and fed into a rule extraction
module (REM) formed by bottleneck residual convolutional blocks to obtain the rule representation, in the perspective of $E_H$:

$$REM(e_1^r, e_2^r) = r_{12}$$  \(\text{(4)}\)

$$REM(e_3^r, e_3^r) = r_{33}$$  \(\text{(5)}\)

$$REM(e_1^r, e_3^r) = r_{13}$$  \(\text{(6)}\)

Finally, we concatenate the rules derived by each block of the perception module ($E_H$, $E_M$, and $E_L$) to represent the rule representations:

$$\text{Concatenate}(r_{12}, r_{12}^m, r_{12}^e) = r_{12}$$  \(\text{(7)}\)

$$\text{Concatenate}(r_{23}, r_{23}^m, r_{23}^e) = r_{23}$$  \(\text{(8)}\)

$$\text{Concatenate}(r_{13}, r_{13}^m, r_{13}^e) = r_{13}$$  \(\text{(9)}\)

The network architecture introduced so far encompasses the fundamental components required to construct RS-CNN. In addition to the perception module, the approach of aggregating information and simultaneously observing two rows for rule extraction bears resemblance to the triplets in the relation module of MRNet and the shared rule extraction mechanism in SAVIR-T. Therefore, RS-CNN can be seen as a combination of these two models. From a technical perspective, RS-CNN maintains the multi-scale encoder architecture and relation module from MRNet, incorporating information fusion from each encoder after reasoning, similar to MRNet. However, RS-CNN replaces the intricate pattern module in MRNet with bottleneck residual convolutional blocks, referred to as the REM module. Meanwhile, the similarity between RS-CNN and SAVIR-T lies in the fact that both models observe two rows simultaneously to derive a rule representation, which is essential to prevent ambiguity in the rule representation derivation process.

In the context of RS-CNN for RAVEN, we incorporate an inductive bias design that ensures permutation-invariance. Permutation-invariance means that rearranging the order of any two rows in the problem matrix should not affect the internal rules [33]. Previous studies have also focused on designing neural networks that adhere to the permutation-invariance criterion [22], [24], [26]. In line with the modularization approach, we imbue the reasoning module with the property of permutation-invariance. As illustrated in Figure 4(a), we explicitly model the original aggregated information and its corresponding permuted version, feeding both into the REM module. Here, $\tau_a$ or $\tau_c$ represents the third row of the problem matrix with its last panel filled by the correct answer (denote as $a$) or a randomly selected answer (denote as $c$) from the answer candidates, respectively. For example, $r_{13}$ represents the rule representation derived by processing the first row and third row of the problem matrix, with the corresponding correct answer from the answer candidate, while $r_{3a}$ represents the permuted version of $r_{13}$.

During the training phase, we utilize $r_{12}$, $r_{21}$, $r_{3a}$, and $r_{3b}$ as guiding representations. We compare each of them with two sets of rule representations: $\{r_{13}, r_{1c}\}$ and $\{r_{23}, r_{2c}\}$, respectively, in terms of cosine similarity, as depicted in Figure 4(b). The training objective is to minimize the discrepancy between the guiding representations and $r_{13}$, $r_{23}$ when the correct answer is selected (i.e., $r_{1c}$, $r_{2c}$ when $c = a$). Inspired by contrastive learning [34], we treat each guiding representation as an individual query and consider $\{r_{13}, r_{1c}\}$ and $\{r_{23}, r_{2c}\}$ as two dictionaries. The correct answer and wrong answers are treated as positive and negative samples, respectively. Each query and dictionary pair is used to compute an InfoNCE loss [35], [36]. These symmetric losses are aggregated to form the overall loss for RS-CNN:

$$L_{RS-CNN} = \sum_{r_q} - \log \frac{e^{\langle r_q, r_{13} \rangle / \tau}}{\sum_{c \neq a} e^{\langle r_q, r_{1c} \rangle / \tau}} \left(1 - \sum_{c \neq a} e^{\langle r_q, r_{1c} \rangle / \tau}ight) - \log \frac{e^{\langle r_q, r_{23} \rangle / \tau}}{\sum_{c \neq a} e^{\langle r_q, r_{2c} \rangle / \tau}} \left(1 - \sum_{c \neq a} e^{\langle r_q, r_{2c} \rangle / \tau}ight)$$  \(\text{(10)}\)

In practice, this loss is deemed as a Cross-Entropy loss augmented with a temperature hyperparameter $\tau$, which means the contribution of negative samples is incorporated into the softmax operation, eliminating the need for any additional terms. During the test phase, the guiding representations are
rule representation by feeding the concatenated vector to a two layer MLP representation with column rule representation, then obtain each row-column Fig. 5. Applying RS-CNN to PGM dataset. Concatenate each row rule problem matrix directly, computational resources are saved

tive biases in the reasoning module rather than shuffling the advantages and disadvantages. By implementing these induc-

tion invariance as an inductive bias in RS-CNN has its

data, resulting in a collection of row-column representations should be considered to live up with transpose invariance.

Reduced to $r_{12}$ and $r_{21}$ since the index of the correct answer is unknown. It is worth noting that SAVIR-T is also trained by comparing rule representations. However, the loss function in RS-CNN is more sophisticated and flexible, as it allows for more rule representations to be available for comparison and employs InfoNCE loss.

We propose necessary revisions to adapt RS-CNN to the PGM dataset. We assumed that both row-wise and column-wise rules always exist (even though it might not be true in practice). We follow the same logic as described earlier to obtain row rule representations. Then, we repeat the process in a column-wise manner to obtain column rule representations. As depicted in Figure 5, we pair up each row rule representation with the corresponding column rule representation by concatenating them, resulting in a collection of row-column rule representations. To facilitate contrastive learning, we designate guiding representations (queries) and dictionaries. The model is optimized to align each query with the correct answer members in each dictionary while diverging from the other members. Furthermore, it is important to note that transposing the problem matrix, which is equivalent to changing the concatenation order of the row-column representations, should not affect the answer.

However, in the case of PGM, the visual attributes of each panel in each row or column are allowed to change randomly, as long as at least one rule exists. Therefore, the aforementioned design is not suitable for PGM as it overlooks the phenomenon of “random change”. In practice, the row-column rule representations are further processed with a shallow MLP layer, which can only mitigate the negative influence of the “random change” phenomenon. That is, the inductive bias inherent in RS-CNN does not align perfectly with the characteristics of the PGM dataset, resulting in a certain degree of incompatibility.

To summarize, the explicit design of transpose and permutation invariance as an inductive bias in RS-CNN has its advantages and disadvantages. By implementing these inductive biases in the reasoning module rather than shuffling the problem matrix directly, computational resources are saved by avoiding repetitive computations in the perception module. However, while the idea behind RS-CNN is compatible with RAVEN and I-RAVEN, it contradicts the nature of PGM, and this contradiction can only be partially alleviated. The pursuit of solving both PGM and RAVEN comprehensively has led to the development of RS-TRAN.

B. RS-TRAN

Successful end-to-end RPM solvers such as MRNet, SAVIR-T, and SCL have demonstrated the importance of encoding image representations from different perspectives to achieve high reasoning accuracy. MRNet utilizes multi-scale encoders, SAVIR-T combines residual convolutional blocks with Transformer blocks to generate multiple visual tokens, and SCL scatters extracted visual representations into pieces. RS-TRAN follows the same spirit as these methods but with a more scalable implementation. It’s worth noting that the notations used in RS-CNN and RS-TRAN are independent of each other.

The perception module of RS-TRAN is built upon the ViT backbone and incorporates a multi-viewpoint mechanism. As shown in Fig. 6, each image in the problem matrix and the answer candidates is divided into 16 patches, which are then passed through a linear embedding layer with positional encoding. The outputs from this layer are fed into Transformer blocks. Instead of using global average pooling or a special classification token as in models like BERT and ViT [4], [20], RS-TRAN keeps all the outputs of the last Transformer block unprocessed, resulting in multiple outputs for each image. This approach maintains the diversity of output features, thereby realizing the multi-viewpoint mechanism. In RS-TRAN, the number of viewpoints corresponds to the number of patches per image, which is set to 16. Technically speaking, this is a standard transformer model for image processing.

In the reasoning module of RS-TRAN, feature representations from each viewpoint are processed individually. When dealing with the RAVEN or I-RAVEN dataset, the problem matrix with the right answer from the answer candidates is set as the positive sample; while the problem matrix with wrong answers is set as the negative samples. As depicted in Fig. 7, for each viewpoint, the representations are concatenated row-wise and passed through a bottleneck module to generate three aggregated information representations. These aggregated representations are then fed into Transformer blocks after positional encoding.

Similar to the perception module, the outputs of the last Transformer block in the reasoning module are left
unprocessed. Each of these representations is sent to a scoring module, which consists of bottleneck layers, to obtain a score. This process represents multi-evaluation, where each problem matrix with an answer from the answer candidates generates several scores, and the total number of scores equals to the product of the number of viewpoints in the perception module and the number of scores in the reasoning module. All the scores are averaged to obtain a final score.

By using the Cross-Entropy loss as the loss function, the network is optimized to ensure that the score of the problem matrix with the correct answer is higher than that of other candidates. For the problem matrix \( \{X^j_p\}_{j=1}^8 \) and the corresponding answer candidates \( \{X^j_w\}_{j=1}^8 \), the score is calculated as follows:

\[
s_j = \frac{1}{3 \times 16} \sum_{m=1}^{3} \sum_{n=1}^{3} s_{m,n}^{j}, \tag{11}
\]

Here, \( m \) denotes the index of the viewpoint, \( n \) denotes the index of the score within each viewpoint, and \( j \) denotes the index of the answer in the answer candidates. Therefore \( s_{m,n}^{j} \) represents the \( n \)th score in the \( m \)th viewpoint for the problem matrix with the \( j \)th answer, while \( s_j \) denotes the final score for the \( j \)th answer.

The loss function for RS-TRAN is defined as follows:

\[
L_{RS-TRAN} = -\log \frac{e^{s_{a}}} {\sum_{i=1}^{8} e^{s_{i}}} - \sum_{c=1}^{8} \log \left( 1 - \frac{e^{s_{c}}}{\sum_{i=1}^{8} e^{s_{i}}} \right), \tag{12}
\]

Note that in practice, negative sample terms are incorporated into the softmax, without explicit extra calculation.

The necessary modifications to adapt RS-TRAN to the PGM dataset are relatively straightforward. As shown in Fig. 8, both aggregated row and column information are involved. All the aggregated information is passed through the Transformer blocks, while the remaining operations remain the same as when applying RS-TRAN to solve the RAVEN or I-RAVEN dataset.

Explicit inductive bias design to express permutation invariance or transpose invariance in RS-TRAN is not necessary.

Changing the order of the aggregated information is equivalent to the capability of the permutation or transpose operator. In RS-TRAN, the order of the aggregated information only affects the order of the output scores (ignoring the impact of positional encoding), which becomes irrelevant after the averaging operation. In this scenario, it is preferable to use learnable positional encoding [37].

RS-TRAN shares similarities with RS-CNN as both architectures comprise a perception module and a reasoning module. However, what sets RS-TRAN apart is its implicit implementation of inductive bias, resulting in a network architecture that is refreshing and uncomplicated. Meanwhile, RS-TRAN does not make any assumptions about the presence of row rule or column rule in a problem matrix. Most importantly, the multi-viewpoint with multi-evaluation scheme enables RS-TRAN to perform parallel reasoning on different visual attributes with distinct rules.

C. RS-TRAN-CLIP

The original RAVEN, I-RAVEN, and PGM datasets are accompanied by meta-data that contains information about the rules of the problem matrix [18], [19]. For instance, we denote the visual attributes as \( V_1, V_2, \ldots, V_6 \), and the rules as \( R_1, R_2, \ldots, R_5 \). Figure 9 provides a simple demonstration of the meta-data.

In previous end-to-end models, the meta-data is typically utilized by incorporating an auxiliary neural branch to predict it. The results reported in these studies demonstrate that the auxiliary task of predicting meta-data improves reasoning.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
The trained RS-TRAN-CLIP is utilized as a pre-training model for RS-TRAN. More specifically, the parameters of the perception module from the trained RS-TRAN-CLIP are imported into RS-TRAN. The RS-TRAN is then trained with this new initialization. It's important to note that the visual end to generate two rule representations. As shown in Figure 11, the ‘CLS’ vectors of all viewpoints are evenly divided into two groups, with each group adheres to one set of rules. To simplify the implementation process, specifically in models with 16 viewpoints, we designate the mean rule representation originating from viewpoints 1 through 8 to align with one set of rules, while the mean rule representation from viewpoints 9 through 16 aligns with the remaining set of rules.

III. When using RS-TRAN-CLIP for PGM, adjustments are necessary in the reasoning module. Firstly, both row and column information are required to handle PGM. The detailed adjustments for obtaining row and column information simultaneously are depicted in Figure 8. Secondly, the ‘line’ and ‘shape’ in PGM are independent of each other. Consequently, the reasoning module also produces two rule representations: one for ‘line’ and the other for ‘shape’. In some cases where ‘line’ or ‘shape’ changes randomly, the corresponding rule is set as ‘NA’.

Finally, let’s discuss a special practice employed in RS-TRAN-CLIP. As mentioned earlier, the PGM dataset includes the ‘NA’ rule, while the RAVEN and I-RAVEN datasets already contain the ‘NA’ rule in their original package. The ‘NA’ rule indicates the random changes of certain visual attributes, and predicting chaotic rule representations as definite ‘NA’ does not contribute to improving the model’s reasoning capability. Therefore, we mask all occurrences of the ‘NA’ term in the training data. This means that RS-TRAN-CLIP only needs to correctly predict ‘non-NA’ rules, as illustrated in Figure 12. By focusing on accurately predicting non-NA rules, the model can better enhance its reasoning capabilities.

The trained RS-TRAN-CLIP is utilized as a pre-training model for RS-TRAN. More specifically, the parameters of the perception module from the trained RS-TRAN-CLIP are imported into RS-TRAN. The RS-TRAN is then trained with this new initialization. It’s important to note that the performance when the baseline reasoning accuracy is relatively low [18], [19], [27]. However, the same auxiliary task has a negative effect once the baseline reasoning accuracy reaches a certain level [24]. Upon closer examination of the structure of the meta-data, we observe that it was originally designed to enhance model performance. However, its binary representation contains less information compared to its natural language form. As depicted in Figure 9, when applying the “or” operation to all binarized [visual attribute: rule] pairs, we lose the specific pairing information between the visual attributes and the rules. This can lead to unnecessary optimization challenges for the model. Considering this limitation, we propose a new approach called RS-TRAN-CLIP, where we abandon the binarized meta-data and instead predict the meta-data in its natural language form. By doing so, we aim to provide the model with more detailed and precise information for improving reasoning capabilities.

The core structure of RS-TRAN-CLIP is inspired by CLIP [30]. RS-TRAN-CLIP refers to three analogous architectures designed for RAVEN and I-RAVEN with one set of rule (e.g., ‘3 × 3 Grid’), RAVEN and I-RAVEN with two sets of rule (e.g., ‘O-IG’), and PGM, respectively. Let’s introduce these architectures one by one.

I. RS-TRAN-CLIP for RAVEN and I-RAVEN with one set of rule. RS-TRAN-CLIP comprises a visual end and a natural language end. The perception part of the visual end, which inherits from RS-TRAN, takes the completed problem matrix with the correct answer as input, and generates image representations from multiple viewpoints. The reasoning part of the visual end and the natural language end is shown in Figure 10. In the reasoning part, instead of employing the multi-evaluation mechanism for image representations from each viewpoint, a ‘CLS’ vector is produced. Then, the average of all ‘CLS’ vectors from different viewpoints is considered as the rule representation for the problem matrix. On the other hand, the natural language end takes all the rules in the dataset as inputs and generates a representation for each rule using a standard Transformer with a ‘CLS’ vector. The model is optimized to align the rule representation from the visual end with the corresponding rule representation from the natural language end, measured by cosine similarity. The InfoNCE loss function is employed once again.

II. RS-TRAN-CLIP for RAVEN and I-RAVEN with two sets of rules. Adjustments are made in this version to enable...
parameters in the perception module are not frozen when training RS-TRAN.

IV. EXPERIMENTS

In this section, we present the results of experiments, evaluating the reasoning accuracy of the proposed models in solving RPM problems, and compare these results with previous state-of-the-art models. Additionally, we conduct ablation studies to assess the impact of dataset size, inductive bias, and multi-viewpoint with a multi-evaluation scheme. Furthermore, we perform a series of intuitive tests to study the intrinsic behaviors of our models and assess generalization ability.

A. Reasoning Accuracy

Both RS-CNN and RS-TRAN models are implemented using the PyTorch framework [38] and optimized using the ADAM optimizer [39]. Models are trained on 4 A-100 GPUs. The training process is terminated when the reasoning accuracy on the validation set no longer improves, and we report the final reasoning accuracy on the test set.

For each configuration in RAVEN and I-RAVEN datasets, we divide the data into three sets: training set, validation set, and test set. The RS-CNN model is trained on a dataset consisting of 80,000 samples for training, 20,000 samples for validation, and 40,000 samples for testing. On the other hand, the RS-TRAN model is trained on a larger dataset, with 155,000 samples for training, 5,000 samples for validation, and 40,000 samples for testing.

Table I presents the reasoning accuracies of RS-CNN, RS-TRAN, RS-TRAN with RS-TRAN-CLIP pre-training, and previous state-of-the-art models. However, it is important to note that direct comparisons between these models may be unfair due to the difference in training set size. The training set size for RS-CNN and RS-TRAN is significantly larger than that of MRNet and SCL. The reason for employing more data is due to the evident performance bottleneck observed in previous models, particularly evident in sub-datasets such as ‘D9’ and ‘O-IG’. We hypothesize that this limitation may be attributed to the restricted amount of training data available, which shall be discussed later. The results in Table I shows that RS-CNN, RS-TRAN and RS-TRAN with RS-TRAN-CLIP pre-training achieves the SOTA results.

Regarding the PGM dataset, we adopt the same training set size, validation set size, and test set size as other models, namely 1,200,000 samples for training, 20,000 samples for validation, and 400,000 samples for testing. As shown in Table II, RS-TRAN, either with or without RS-TRAN-CLIP, surpasses the former state-of-the-art results, while RS-CNN only achieves a reasoning accuracy of 82.8%. This outcome can be expected since the inductive bias of RS-CNN is not compatible with the characteristics of the PGM dataset.

Although RS-CNN and RS-TRAN have already demonstrated satisfactory results, RS-TRAN, when combined with a pre-training perception module powered by RS-TRAN-CLIP, achieves even higher levels of reasoning accuracy, especially in challenging RPM tasks. Figure 13 illustrates the rapid and seamless convergence of RS-TRAN with pre-training towards its upper performance limit.

B. Effect of Dataset Size

In this section, we investigate the impact of dataset size on the performance of neural networks. It is well-known that the performance of neural networks heavily relies on the size of the dataset [40], and this holds true for Transformer-based networks as well [20], [41]. To simplify our analysis, we focus on three specific scenarios: ‘Center,’ ‘3 × 3 Grid,’ and ‘O-IG’ in both the RAVEN and I-RAVEN datasets.

We observed that the performance of RS-CNN in the RAVEN dataset is mediocre compared to its results in the I-RAVEN dataset. To address this issue, we increase the size of the training set in RAVEN to 155,000/configurations. Furthermore, it should be noted that RS-TRAN requires a large amount of training data. Therefore, we decrease the training set size of I-RAVEN to 80,000/configurations.

The general conclusion is not surprising, as demonstrated in Table III and Table IV: a larger dataset size leads to better performance. Specifically, we observed a significant

![Fig. 13. Training trajectories of RS-TRAN for PGM, with & without pre-training.](image-url)
improvement in RS-CNN when the training set size increased from 80K to 155K. On the other hand, RS-TRAN already exhibited good performance with a dataset size of 80K, and further improvements were achieved by expanding the training dataset size. We also recognize that RS-CNN is more sensitive to dataset size compared to RS-TRAN, indicating that its stability is not as high as that of RS-TRAN. This phenomenon partly accounts for the performance bottleneck encountered in previous models, primarily because the majority of them rely on CNN architectures.

C. Ablation Studies

In this section, we delve into the architecture of RS-CNN and RS-TRAN, and explore the importance of incorporating inductive biases into RS-CNN. We also examine the plausibility of RS-TRAN in terms of its “natural expressiveness” of these inductive biases. Additionally, we investigate the effectiveness of multi-viewpoint and multi-evaluation approaches in RS-TRAN. All the ablation studies are conducted on the PGM dataset and specific configurations of the I-RAVEN dataset.

In RS-CNN, the removal of explicitly designed inductive biases implies the exclusion of permutation and transpose operators. On the other hand, in RS-TRAN, the consideration of permutation and transpose invariance involves the explicit traversal of input ordering within the reasoning module. The results presented in Table V convey a strong message: explicit inductive bias design plays a crucial role in RS-CNN, whereas the architecture of RS-TRAN inherently possesses the property of permutation and transpose invariance. However, incorporating explicit inductive bias design does not harm the performance of RS-TRAN.

D. More on Multi-Viewpoint and Multi-Evaluation

Both the reasoning accuracy and the ablation study demonstrate the necessity of incorporating the multi-viewpoint with multi-evaluation mechanism in RS-TRAN. However, the specific effects of this mechanism are still not fully understood. To gain a clearer understanding, we conducted mask experiments.

In RS-TRAN, the multi-viewpoint mechanism is implemented by preserving all the outputs of the perception module. Each image in the problem matrix is divided into 16 patches, resulting in 16 viewpoints (outputs) in the perception module. By masking out specific viewpoints (outputs) and evaluating the reasoning accuracy of RPM problems with specific rules, we can roughly assess the functionality of each output. The same procedure is applicable to the multi-evaluation mechanism as well.

To simplify the study, we focused on the multi-viewpoint mechanism using single-rule problems from the PGM dataset. We masked out half of the outputs in the perception module of RS-TRAN. As illustrated in Figure 14, viewpoints $[0 - 7]$ and $[8 - 15]$ were found to focus on different rules. By employing the masking technique, the model acquires a certain level of post-hoc interpretability [42], [43]. Based on these findings, we concluded that the multi-viewpoint with multi-evaluation mechanism empowers the model to solve problems from multiple perspectives simultaneously.

E. On Generalization

In PGM, besides the ‘neutral’ dataset where the data from the training set and test set are independent and identically distributed (I.I.D.), there are other datasets where the distribution of training sets and test sets differs (Out of Distribution, O.O.D.). We assess the generalization ability of the RS-TRAN model, without and with RS-TRAN-CLIP pre-training, by training them on these datasets. The experimental setup remains consistent with the ‘neutral’ case.

The results presented in Table VII reveal the vulnerability of RS-TRAN when it comes to generalization challenges. While RS-TRAN with the TRAN-CLIP pre-training, demonstrates
Fig. 14. Mask experiments on single-rule data of PGM. Only problems with one rule will be considered to avoid semantic ambiguity.

TABLE VII
GENERALIZATION RESULTS OF RS-TRAN (WITHOUT / WITH RS-TRAN-CLIP) IN PGM

| Dataset                  | Accuracy(%) |
|--------------------------|-------------|
| Interpolation            | 77.2/94.1   |
| Extrapolation            | 19.2/17.4   |
| Held-out Attribute      | 12.9/12.6   |
| Held-out Attribute line-type | 24.7/21.4 |
| Held-out Tripples        | 22.2/27.7   |
| Held-out Pairs of Tripples | 43.6/96.0   |
| Held-out Attribute Pairs | 28.4/96.8   |

F. Does the Model Learn the Pre-Defined Rules?

Let’s consider the evaluation of cosine similarity between rule representations in RS-CNN. Based on this approach, it is assumed that the rule representations for the same rule should be identical or at least exhibit extremely close resemblance throughout the entire dataset. However, this assumption does not hold true in RS-CNN. Taking the configuration of ‘Center’ and ‘3 × 3 Grid’ in RAVEN as an example, 20 problem matrices with identical rules are generated for each configuration. As depicted in Figure 15, although the rules for these 20 problem matrices are identical, perfect matching of rule representations can only be observed within each problem matrix. In other words, RS-CNN does not summarize rules in the way they are predefined in the dataset. On the other hand, RS-TRAN with pre-training aligns the rule representations of problem matrices that share the same rule, resulting in improved overall performance.

G. Computation Efficiency

We take the most complicated case, namely training RS-TRAN with RS-TRAN-CLIP on PGM dataset as example. Since PGM is a large dataset, in each epoch, we only train a randomly selected minibatch of the training set. The whole training set is divided into 6 parts, and each part contains 200,000 samples. The batch size within each epoch is set as 500. Thanks to the GPU hardware acceleration technology and optimized deep learning libraries, training one epoch of RS-TRAN-CLIP takes approximately 3 minutes and 30 seconds, and it achieves a higher validation accuracy around the 37th epoch. Training one epoch of RS-TRAN takes around 6 minutes and 46 seconds, and it reaches a higher validation accuracy at about the 40th epoch.

V. CONCLUSION

In this paper, we have presented two RPM solvers based on CNN and ViT architectures, demonstrating through
experimental analysis that well-designed inductive biases are crucial for achieving successful end-to-end RPM neural solvers. Our work has also revealed that meta-data containing non-pixel level information (such as rule information and structure information) is not essential in abstract reasoning when sufficient pixel-level data with comprehensive coverage of rules is available. However, properly incorporated meta-data can expedite the training process and significantly enhance the model’s performance.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” Commun. ACM, vol. 60, no. 6, pp. 84–90, 2017.

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778.

[3] A. Vaswani, “Attention is all you need,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 5998–6008.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” 2018, arXiv:1810.04805.

[5] T. Brown, “Language models are few-shot learners,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 1877–1901.

[6] I. Goodfellow, “Generative adversarial networks,” Commun. ACM, vol. 63, no. 11, pp. 139–144, 2020.

[7] D. P. Kingma and M. Welling, “Auto-encoding variational Bayes,” 2013, arXiv:1312.6114.

[8] J. Ho, A. Jain, and F. Abbeel, “Denoising diffusion probabilistic models,” in Proc. Adv. Neural Inf. Process. Syst., vol. 33, 2020, pp. 6840–6851.

[9] S. Antol et al., “VQA: Visual question answering,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 2425–2433.

[10] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick, “CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 1988–1997.

[11] X. Wang et al., “Self-consistency improves chain of thought reasoning in language models,” 2022, arXiv:2203.11171.

[12] I. Dori et al., “A neural network solves, explains, and generates university math problems by program synthesis and few-shot learning at human level,” Proc. Nat. Acad. Sci. USA, vol. 119, no. 32, Aug. 2022, Art. no. e213433119.

[13] M. Chen et al., “Evaluating large language models trained on code,” 2021, arXiv:2107.03379.

[14] S. Devweger, C. A. Rothkopf, and F. Jükel, “Solving bongard problems with a visual language and pragmatic reasoning,” 2018, arXiv:1804.04452.

[15] W. Nie, Z. Yu, L. Mao, A. B. Patel, Y. Zhu, and A. Anandkumar, “Bongard-LOGO: A new benchmark for human-level concept learning and reasoning,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 16468–16480.

[16] D. Teney, P. Wang, and J. Cao, “V-prom: A benchmark for visual reasoning using visual progressive matrices,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 12071–12078.

[17] J. C. Raven, Raven’s Progressive Matrices. Torrance, CA, USA: Western Psychological Services, 1938.

[18] C. Zhang, F. Gao, B. Jia, Y. Zhu, and S.-C. Zhu, “RAVEN: A dataset for relational and analogical visual REmoNing,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 5312–5322.

[19] D. Barrett, F. Hill, A. Santoro, A. Morcos, and T. Lillicrap, “Measuring abstract reasoning in neural networks,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 511–520.