Multi-Label Contrastive Learning for Abstract Visual Reasoning

Mikołaj Małkiński and Jacek Mańdziuk, Senior Member, IEEE

Abstract—For a long time, the ability to solve abstract reasoning tasks was considered one of the hallmarks of human intelligence. Recent advances in the application of deep learning (DL) methods led to surpassing human abstract reasoning performance, specifically in the most popular type of such problems—Raven’s progressive matrices (RPMs). While the efficacy of DL systems is indeed impressive, the way they approach the RPMs is very different from that of humans. State-of-the-art systems solving RPMs rely on massive pattern-based training and sometimes on exploiting biases in the dataset, whereas humans concentrate on the identification of the rules/concepts underlying the RPM to be solved. Motivated by this cognitive difference, this work aims at combining DL with the human way of solving RPMs. Specifically, we cast the problem of solving RPMs into a multilabel classification framework where each RPM is viewed as a multilabel data point, with labels determined by the set of abstract rules underlying the RPM. For efficient training of the system, we present a generalization of the noise contrastive estimation algorithm to the case of multilabel samples and a new sparse rule encoding scheme for RPMs. The proposed approach is evaluated on the two most popular benchmark datasets [I-RAVEN and procedurally generated matrices (PGM)] and on both of them demonstrate an advantage over the state-of-the-art results.

Index Terms—Abstract visual reasoning, contrastive learning, Raven’s progressive matrices (RPMs).

I. INTRODUCTION

Abstract visual reasoning tasks are considered a widely accepted way of measuring human intelligence (IQ). The most popular example of such a task is Raven’s progressive matrices (RPMs) [1], [2], where one is required to identify abstract relations between visually simple objects and their attributes (see Fig. 1). Solving an RPM requires an incremental strategy for inducing regularities in the matrix [3], which justifies the importance of RPMs in measuring IQ. Previous works identified a major gap between the performance of machine learning (ML) algorithms and humans [4], [5], which sparked interest in these problems within the deep learning (DL) community. In effect, human performance advantage has quickly vanished along with the development of advanced DL models [6].

However, the way ML algorithms solve RPMs leaves something to be desired. In order to solve these tasks, humans are required to come up with a strategy, which correctly identifies all the underlying rules and differentiates them from distracting features, while the goal of the vast majority of ML approaches is to focus on selecting a correct answer, without providing an explanation for their output. Such a direct optimization problem formulation encourages neural models to simplify the solution process and rely on biases instead of discovering the internal problem structure. Indeed, such pathology was identified for the RAVEN dataset [5], where the networks were able to arrive at the correct answer only by checking the set of possible choices [7]. Similar problems were also noted in other visual reasoning tasks [8], [9].

A large body of the cognitive literature identified contrastive mechanisms and the ability to create analogies as the key ingredients for adaptive problem solving [10]–[12]. Replication of the above-mentioned mechanisms in neural models, either in the form of an explicit contrastive module [13] or through the way the data is presented during training [14], resulted in notable improvements in solving RPMs. Moreover, it was demonstrated that when models were additionally

![Fig. 1. Solving an RPM consists in filling in the bottom-right empty panel “?” with the best-fitting answer panel (A–H). To this end, one must identify abstract relationships underlying an RPM (often hidden behind random visual distractors) and contrast possible answers. Left RPM, although has more visual details, is governed by only a single rule (rowwise AND applied to shape position), whereas perceptually simpler right RPM contains eight distinct relations applied to both outer and inner structures. The examples come from the PGM and I-RAVEN datasets, respectively, and in both cases, the correct answer is A.](image-url)
trained to predict a symbolic explanation for their answers (so-called auxiliary training), their generalization capabilities increased substantially [4]. Nonetheless, despite the notable positive impact of the auxiliary training, the topic remains underexplored.

### A. Motivation and Contribution

In order to build models that understand how to solve RPMs, we look for alternative training approaches. Encouraged by the effectiveness of both contrastive learning methods and symbolic explanations, we seek to develop a novel auxiliary training method for abstract visual reasoning tasks. We aim to exploit two recurring themes present in human approaches: the contrastive mechanism which differentiates between correct and wrong answers to the RPMs and the inherent ability to first identify all the abstract relations defining a given RPM and then use them to select the answer. The main contribution of this work is fourfold.

1) Approaching the problem of solving RPMs by casting it into a multilabel classification framework, where labels are determined by the underlying abstract rules.

2) Devising a new formulation of the noise contrastive estimation (NCE) learning algorithm for the case of multilabel samples, whose application imitates a human approach to solving RPMs.

3) Proposition of a new sparse rule encoding scheme for RPMs which provides a more explicit rule representation compared with the method used in prior works.

4) Integration of both contrastive and auxiliary training into a novel ML approach to solving abstract visual reasoning tasks (Fig. 2), which sets new state-of-the-art results on two major benchmarks—I-RA VEN [7] and procedurally generated matrices (PGM) [4].

### II. RELATED WORK

#### A. Raven's Progressive Matrices

Although RPMs are characterized by rather simple visual representation, solving them is often a challenging task, as it requires correctly identifying all abstract relations between the component RPM images. In order to measure the generalization ability of neural modules in relational reasoning problems, Barrett et al. [4] introduced the dataset of PGM, which contains RPM problems divided by the authors into train and test sets. An important part of the PGM dataset is meta-target annotations, which encode the relations between objects and their attributes in a given RPM. Models trained to additionally predict these meta-targets by means of an auxiliary training were shown to possess stronger generalization capabilities compared with those that did not employ such an auxiliary training.

The topic of meta-annotations was further extended in the RAVEN dataset [5], which contains additional structural annotations and RPMs with highly compositional structure. However, as reported in previous works, in the case of RAVEN data, the auxiliary training seems to have little to no positive impact [5], [13]. Our investigations of the I-RAVEN dataset show that a more explicit rule encoding scheme can mitigate this problem.

A recent paper [7] put in question the abstract visual reasoning capabilities of reported models, by highlighting a major flaw in the process of generating candidate answers in RAVEN. Due to an unintended bias in the generated set of possible choices, models trained purely on answers were still able to select a correct solution. To mitigate this problem, the dataset I-RAVEN was proposed, in which the answers are generated in an unbiased way [7].

Initial reports on machine RPM solving demonstrated mixed results, where some of the works reported superhuman performance on simpler datasets [15], [16], while others identified a notable gap between humans and ML algorithms in more demanding matrices [5]. The premise of reaching the level of human intelligence on gradually more and more challenging benchmarks stimulated active research in this area. A number of methods were designed for solving RPMs, which aim to identify relations between sets of objects [4], [17] using a relation network [18], reason with multilayer multiplex graph neural network [19] or discover compositional representations using a scattering transformation [6]. Other works investigate human-inspired approaches, by measuring feature differences [16] or exploiting the hierarchical structure of RPMs [7].

#### B. Contrastive Representation Learning

Previous studies have shown that the ability to make analogies and contrasting experiences is a key ingredient of human intelligence [10], [11], [20], [21]. Consequently, these concepts were also implemented in ML systems solving abstract reasoning problems. Hill et al. [14] discussed how to make analogies by contrasting abstract relational structure, while Zhang et al. [13] incorporated contrast in the model architecture. Motivated by these findings, we propose another
approach to abstract reasoning, by incorporating a contrastive mechanism directly into the objective function.

Among contrastive learning algorithms, the most related to our work is the NCE method [22], [23], which was successfully utilized in various domains, such as image recognition [24], natural language processing [25], or Atari games [26], [27]. NCE aims at building congruent representations for semantically related samples (positive pairs) and dissimilar representations for unrelated observations (negative pairs). It was shown that the concept of learning representations with NCE (as well as with other contrastive learning methods) is related to mutual information (MI) maximization [28]–[30].

Contrastive learning is particularly useful in semisupervised pretraining methods, where the availability of labeled data in downstream tasks is scarce [24], [31]–[33]. A recent work [34] has shown that contrastive learning can outperform classical approaches also in a fully supervised setting (with cross-entropy) and is characterized by greater robustness and stability to hyperparameter selection. This superior performance is a result of employing multiple positive pairs, which was further shown to increase MI lower bound by considering MI estimation as multilabel classification [35]. Despite this multilabel viewpoint, the framework proposed by Song and Ermon [35], as well as the vast majority of prior works (e.g., [30], [34], [36]), focus on multiclass classification and do not discuss the applicability of contrastive methods to multilabel problems. Our work aims at bridging this gap by defining a general contrastive learning framework suitable for multilabel samples.

III. MULTILABEL CONTRASTIVE LEARNING

Motivated by successful applications of contrastive learning methods in other domains, we aim to extend this technique for multilabel settings and investigate its efficacy in learning representations of abstract visual reasoning problems. We start by proposing a general multilabel learning framework and then discuss its applicability to solving RPMs.

A. Preliminaries

Our method is built on the foundations of Supervised Contrastive Learning proposed by Khosla et al. [34], which is extended in this article to support multilabel data. Given a randomly sampled batch \( \{x_i, y_i\}_{i=1}^N \in \{\mathcal{X} \times \mathcal{Y}\} \) of size \( N \), the base method consists of the following integral components.

1) A data augmentation (DA) module, which transforms image \( x_i \) into two randomly augmented views \( \tilde{x}_{2i} \) and \( \tilde{x}_{2i-1} \), leading to an extended batch \( \{x_i, \tilde{y}_i\}_{i=1}^{2N} \) such that \( \tilde{y}_{2i} = \tilde{y}_{2i-1} = y_i \). It should be noted that while this step is critical to supervised contrastive learning its usage in our framework is optional.

2) An encoder network \( f_\theta \) which forms latent representations of the augmented views, defined as \( h_i = f_\theta(\tilde{x}_i) \). The representations obtained from the encoder are \( \ell_2 \)-normalized, which encourages learning from hard negatives and hard positives [34] and simplifies the final linear classification task by aligning the features from positive pairs and uniformly distributing them on the hypersphere [37].

3) A projection network \( g_\phi \), which maps feature representation into a lower dimensional vector \( z_i = g_\phi(h_i) \) suitable for computation of the contrastive loss. The representations obtained from the projection network are \( \ell_2 \)-normalized, which allows for measuring the similarity between two vectors based on their dot product. The projection network is implemented as an MLP with a single hidden layer. The presence of this nonlinear component was shown to be of critical importance [24], [31], [32]. Both \( f_\theta \) and \( g_\phi \) are optimized jointly with respect to the supervised contrastive loss, defined as follows:

\[
\mathcal{L}^\text{sup} = \sum_{i=1}^{2N} \mathcal{L}^\text{sup}_i
\]

\[
\mathcal{L}^\text{sup}_i = \frac{1}{2N_{\tilde{y}_i}} \sum_{j=1}^{2N} \mathbb{I}_{i \neq j} \cdot \mathbb{I}_{\tilde{y}_i = \tilde{y}_j} \cdot \mathcal{L}^\text{sup}_{i,j}
\]

\[
\mathcal{L}^\text{sup}_{i,j} = -\log \frac{\exp(z_i; z_j/\tau)}{\sum_{k=1}^{2N} \exp(z_i; z_k/\tau)}
\]

where \( N_{\tilde{y}_i} \) is the number of samples in a given mini-batch with the same label as the anchor \( i \), \( \mathbb{I}_B \in \{0, 1\} \) is an indicator which evaluates to 1 iff \( B \) is true, and \( \tau > 0 \) is a constant temperature parameter.

After the pretraining stage, the weights of the encoder \( f_\theta \) are frozen and the projection network \( g_\phi \) is replaced with a randomly initialized linear classification head, which is trained with cross-entropy on the downstream task. This procedure, commonly referred to as the linear evaluation protocol [28], [38], provides a simple way to measure the quality of learned representations.

B. Multilabel Contrastive Loss

In the default setting, the supervised contrastive loss supports multiclass samples, i.e., \( y_i \in \mathcal{Y} \). In order to extend it to multilabel samples \( \{x_i, Y_i\} \) such that \( Y_i \subseteq \mathcal{Y} \), we propose a novel objective function, the multilabel contrastive loss, which is defined as follows:

\[
\mathcal{L}^\text{mlc} = \sum_{i=1}^{2N} \mathcal{L}^\text{mlc}_i
\]

\[
\mathcal{L}^\text{mlc}_i = \frac{1}{2N_{\tilde{y}_i}} \sum_{j=1}^{2N} \mathbb{I}_{i \neq j} \cdot \mathbb{I}_{\tilde{y}_i \neq \tilde{y}_j} \cdot \mathcal{L}^\text{mlc}_{i,j}
\]

where \( N_{\tilde{y}_i} = |\{\tilde{y}_j \mid \tilde{y}_i \neq \tilde{y}_j\}| \) is the number of samples in a given mini-batch which share at least one label with the anchor \( i \) and \( \mathcal{L}^\text{mlc}_{i,j} = \mathcal{L}^\text{sup}_{i,j} \).

Compared with (2), the difference lies in the definition of positive pairs for anchor \( i \). While the base formulation defines the set of positives as those samples with exactly the same label, (5) defines it as those samples which share at least one label. Formulation (5) preserves the key properties of the base objective, i.e.: 1) aggregates an arbitrary number of positive samples in the numerator and 2) increases contrastive strength...
observe that each RPM can be naturally viewed as a multilabel learning (MLCL) framework for solving RPMs, let us first examples, with full RPMs, are presented in the supplementary material. In order to preserve RPM abstract structure, neither roll nor shuffle transformations are used when a progression rule governs object positions. For clarity, we only depict different views of a single row for two RPMs belonging to configurations 2x2grid (left part) and 3x3grid (right part), respectively. Additional examples, with full RPMs, are presented in the supplementary material.

by using all negative samples in the denominator. At the same time, the modified definition of positive pairs allows handling samples with multiple labels.

C. Adaptation to RPMs

In order to utilize the above-proposed multilabel contrastive learning (MLCL) framework for solving RPMs, let us first observe that each RPM can be naturally viewed as a multilabel sample, where labels correspond to the rules governing the RPM. Let us consider a mini-batch \( \mathcal{M}_i^{N} \) of size \( N \), where \( \mathcal{M}_i = \{X_i, Y_i, k_i\} \) represents the RPM instance composed of a set of 16 images \( X_i \), i.e., eight context panels and eight answer panels (out of which only one correctly completes the matrix). \( Y_i \subset Y \) is a set of associated rules, such that \( 1 \leq |Y_i| \leq N_y \) and \( k_i \in \{1, \ldots, 8\} \) is an index of the correct answer. Here, \( Y \) is the set of all possible rules, and \( N_y \) is a dataset-dependent maximal number of rules for a single RPM.

1) Data Augmentation: Previous works on solving RPMs do not report the usage of DA. On the other hand, augmentation was highlighted as a fundamental component of NCE-based learning in other domains, e.g., by Chen et al. [24]. In order to conduct a meaningful comparison with the prior literature, in the experiments both setups with and without augmentation are considered. In the former case, for each RPM in the mini-batch we apply two randomly selected augmentations, which gives a mini-batch of \( 2N \) RPMs \( \{\tilde{X}_i, \tilde{Y}_i, \tilde{k}_i\}^{2N} \), where \( \tilde{X}_2 \) and \( \tilde{X}_{2-1} \) are both obtained by augmenting \( X_i \). All augmentations preserve both the underlying rules and index of the correct answer, hence: \( \tilde{Y}_{2-1} = \tilde{Y}_2 = Y_i \) and \( \tilde{k}_{2-1} = \tilde{k}_2 = k_i \). Since RPMs consist of greyscale images, we cannot rely on augmentation methods popular in image recognition. Instead, simple transformations which rearrange or rotate images are used, as showcased in Fig. 3. If augmentation is not applied a base mini-batch \( \mathcal{M}_i^{N} \) is used.

2) Contrastive Pretraining: For each RPM, we construct eight different matrices by filling in the remaining empty panel with each of the answer panels. For each sample \( \mathcal{M}_i \), this gives us a single RPM, which satisfies all rules \( \tilde{Y}_i \) and seven RPMs, which satisfy fewer rules, because of the incorrectly chosen answer. We arrive at a batch divided into two components: one containing correct RPMs with their rules \( \{\tilde{x}_i, \tilde{Y}_i\}^{N} \) and the other one composed of incorrect RPMs \( \{\tilde{x}_i, \tilde{Y}_i\}^{N-1} \) (see Fig. 4). This division allows us to consider the incorrectly completed RPMs as additional negative samples, for which

\[
\mathcal{L}_{\text{mlc}} = - \log \frac{\exp(z_i \cdot z_j / \tau)}{\sum_{k=1}^{2N} \mathbb{1}_{i \neq k} \cdot \exp(z_i \cdot z_k / \tau) + \Sigma_{i,k}} \quad (6)
\]

\[
\Sigma_{i,k} = \sum_{l=1}^{7} \exp(z_i \cdot z_{i,l} / \tau). \quad (7)
\]

We optimize both \( \phi \) and \( \omega \) with respect to the multilabel contrastive loss defined in (4)–(7).

3) Auxiliary Training: We support the contrastive pretraining with an auxiliary loss \( \mathcal{L}_{\text{aux}} \) introduced in [4]. For this purpose, we employ a rule discovery network \( p_a \) (implemented as an MLP with a single hidden layer), which transforms outputs of the encoder \( f_\theta \) into a meta-target prediction \( \hat{Y}_i = p_a(h_i) \cup \{h_{i,l}\}_{l=1}^7 \in \mathbb{R}^d \), where \( d \) is the dimension of meta-target encoding. We compare two setups for the rule encoding.

1) Dense—multihot encoding scheme introduced in [4] and [5]. It encodes each rule as a binary string of fixed length (\( d = 12 \) for PGM and \( d = 9 \) for I-RAVEN) and performs logical OR operation on the whole set of rules.

2) Sparse—our proposed scheme, which encodes each rule as a one-hot vector of fixed length equal to the number of unique relations in the dataset (\( d = 50 \) for PGM and
\[ d = 38 \text{ for I-RAVEN). Similar to dense encoding it performs logical OR operation on the whole set of rules.} \]

The advantage of proposed sparse encoding over the dense encoding used in prior works stems from information lossless or operation, i.e., from sparse representation one can recover which rules were encoded, which is not always the case for dense encoding (e.g., when a single object is governed by multiple rules or the same relation is applied to different objects). The difference between the two encodings is further discussed in Appendix B. Since sparse encoding provides a more explicit training signal it is the default encoding method for MLCL. Rule predictions are activated using a sigmoid unit and the loss is calculated using binary cross-entropy. We define the multilabel contrastive loss for solving RPMs as

\[
\mathcal{L} = \gamma \mathcal{L}_\text{mc} + \beta \mathcal{L}_\text{aux} \tag{8}
\]

where \( \gamma \) and \( \beta \) are balancing factors. In the default setting \( \gamma = 1 \) and \( \beta = 10 \). The joint pretraining procedure which combines DA, the multilabel contrastive loss, and the auxiliary loss, is formalized in Algorithm 1.

### Algorithm 1 MLCL: Pretraining

**Input:** batch size \( N \), constant \( \tau \), networks \( f_\theta \), \( g_\phi \), \( p\omega \), data augmentation module \( \mathcal{A} \)

**Output:** trained encoder network \( f_\theta \)

1. for a sampled minibatch \( \{X_i, Y_i, k_i\}_{i=1}^N \) do
2. for \( i = 1 \) to \( N \) do
3. # First augmented view
4. \( \tilde{X}_{2i-1} \leftarrow A(X_i) \)
5. \( \{h_{2i-1} \cup \{h_{2i-1}\}_i\}_{i=1}^N \leftarrow f_\theta(\tilde{X}_{2i-1}) \)
6. \( z_{2i-1} \leftarrow g_\phi(h_{2i-1}) \)
7. \( \{z_{2i-1}_i\}_{i=1}^N \leftarrow \{g_\phi(h_{2i-1})\}_i^N \)
8. \( \hat{p}_{2i-1} \leftarrow p_\omega(h_{2i-1}) \cup \{h_{2i-1}_i\}_i^N \)
9. # Second augmented view
10. \( \tilde{X}_2i \leftarrow A(X_i) \)
11. \( \{h_{2i} \cup \{h_{2i}\}_i\}_{i=1}^N \leftarrow f_\theta(\tilde{X}_{2i}) \)
12. \( z_{2i} \leftarrow g_\phi(h_{2i}) \)
13. \( \{z_{2i}_i\}_{i=1}^N \leftarrow \{g_\phi(h_{2i})\}_i^N \)
14. \( \hat{p}_{2i} \leftarrow p_\omega(h_{2i}) \cup \{h_{2i}\}_i^N \)
15. end for

16. \( \mathcal{L} = \gamma \mathcal{L}_\text{mc}(\tau, \{Y_i^N\}_i, \{z_i\}_i^N, \{z_i^N\}_i) + \beta \mathcal{L}_\text{aux}(\{Y_i^N\}_i, \{\hat{p}_i^N\}_i) \)
17. update networks \( f_\theta \), \( g_\phi \) and \( p\omega \) to minimize \( \mathcal{L} \)
18. end for

4) **Linear Evaluation:** After the contrastive pretraining step, we discard the projection head \( g_\phi \) and the rule discovery network \( p\omega \), freeze the parameters of the encoder \( f_\theta \) and attach a simple linear scoring head \( s_\psi \) with a single-output neuron. For each RPM problem \( \mathcal{M}_1 \), we employ \( f_\theta \) to generate the matrix representations \( \{h_{1i}^f\}_{i=1}^8 \), calculate a score using the scoring head \( s_\psi \) and apply softmax to produce a probability distribution \( \hat{p} \) over the set of possible answers. Using the estimated probability, we optimize the scoring head with a standard cross-entropy loss and keep the weights of the encoder network frozen. The linear evaluation procedure is outlined in Algorithm 2. After training, the obtained models \( f_\theta \) and \( s_\psi \) can be applied to solve new RPMs, based solely on the matrix panels \( X \) (cf. Algorithm 3).

Algorithm 2 MLCL: Linear Evaluation

**Input:** batch size \( N \), trained network \( f_\theta \), network \( s_\psi \)

**Output:** trained RPM classifier \( f_\theta \circ s_\psi \)

1. freeze the weights of \( f_\theta \)
2. for a sampled minibatch \( \{X_i, k_i\}_{i=1}^N \) do
3. for \( i = 1 \) to \( N \) do
4. \( \{h_{1i}\}_{i=1}^8 \leftarrow f_\theta(X_i) \)
5. \( \hat{p}_i \leftarrow \{s_\psi(h_{1i})\}_i \)
6. \( p_i \leftarrow \text{onehot}(k_i) \)
7. \( L_i \leftarrow \text{CE}(p_i, \hat{p}_i) \)
8. end for
9. \( L \leftarrow \frac{1}{N} \sum_i L_i \)
10. update network \( s_\psi \) to minimize \( L \)
11. end for

Algorithm 3 MLCL: Inference

**Input:** RPM images \( X \), trained networks \( f_\theta \) and \( s_\psi \)

**Output:** index of the correct answer \( \hat{k} \)

1. \( \{h_{1i}\}_{i=1}^8 \leftarrow f_\theta(X) \)
2. \( \hat{k} \leftarrow \arg \max_i \{s_\psi(h_i)\}_{i=1}^8 \)

5) **Fine-Tuning:** To further push the limits of the learned representations, the linear evaluation step can be followed by an optional fine-tuning (FT) stage. In this stage, we unfreeze the encoder network \( f_\theta \) and apply the same training protocol as in linear evaluation, but train the model end-to-end.

### IV. Experiments

We compare our method (MLCL) with a fully supervised framework for solving RPMs, used throughout the literature [4], [5], referred to as CE. CE utilizes both the encoder network \( f_\theta \) and the scoring head \( s_\psi \), which are optimized in an end-to-end manner with respect to the cross-entropy objective function. Since our contrastive training method is inherently based on the description of the RPM rules, we support the supervised training with the same auxiliary loss \( \mathcal{L}_\text{aux} \) and rule discovery network \( p\omega \), obtaining

\[
\mathcal{L} = \mathcal{L}_\text{ce} + \beta \mathcal{L}_\text{aux} \tag{9}
\]

where \( \beta \) is a balancing coefficient. This enhanced training is referred to as CE + AUX. Following the setup from [4], we set \( \beta = 10 \) in all experiments. In order to comprehensively compare our method with the baseline, in the experimental evaluation, we consider three different state-of-the-art models for abstract visual reasoning and use two abstract reasoning RPM benchmarks.

* a) **Datasets:** Our main findings are demonstrated on the newest RPM benchmark set—I-RAVEN [7], which contains visually diverse RPMs with highly compositional structure. I-RAVEN is a modification of the RAVEN dataset [5] which fixes the defect of biased answer panels. It contains novel relation types which are not present in PGM. Each RPM in I-RAVEN can be governed by up to eight different rules.
I-RAVEN is divided into seven distinct visual configurations. In the default setting each of them contains 10K problem instances.

Additionally, we support our studies with the experiments on large-scale PGM dataset [4], containing RPM problems with between one and four abstract rules. The main purpose of PGM is to test the generalization skills of ML models across various regimes. Each regime is characterized by explicitly defined differences between training and test data, e.g., interpolation or extrapolation or the ability to reason about relations not seen in the training set. The dataset contains 1.42M RPM problems per regime.

A. Results on I-RAVEN

We start the experimental evaluation by comparing the proposed MLCL method with CE training setups with or without DA, on three different, above-mentioned ML abstract reasoning models, on the I-RAVEN dataset (see Table I). To better showcase the quality of representations learned with MLCL, we report results obtained only with linear evaluation and analyze the effect of FT separately in the ablation study section.

1) Sparse vs Dense Encoding: Auxiliary training with sparse encoding (CE + AUX-sparse) always improves the performance over the baseline CE setup [with β = 0 in (9)]. At the same time, CE + AUX-dense results in worse than CE performance for CoPINet, which aligns with the outcomes reported in [13]. Similar conclusions can be drawn in terms of MLCL, for which sparse encoding yields better results for all three encoders. In a direct comparison on I-RAVEN, the more explicit, sparse rule encoding scheme is overall beneficial for both training methods (CE and MLCL), regardless of a particular ML model. Consequently, in the following experiments, we will focus on MLCL with sparse encoding and refer to it simply as MLCL.

Sparse encoding is essentially a one-hot encoding of the set of all unique rules. We would say that a model predicts the existence of the j'th rule if sigmoid(θ_j) > 0.5. In Table II, we compare the F1-scores of rule prediction of all encoders under auxiliary training with sparse and dense encoding—provides a more condensed representation, at the expense of not preserving crucial information. The differences between sparse and dense encodings, illustrated with an example, are further discussed in Appendix B.

It is important to note that all approaches used in prior works which rely on the dense (multihot) encoding proposed in [4] and [5] are not able to predict the exact rules in the considered RPM. This shortcoming arises from the nature of multihot encoding, which—when compared to sparse encoding—provides a more condensed representation, at the expense of not preserving crucial information. The differences between sparse and dense encodings, illustrated with an example, are further discussed in Appendix B.

2) (−DA): When augmentation is not applied, MLCL significantly outperforms the base setup (CE), which does not include the rule-related information in the training signal. This observation confirms that MLCL is able to absorb and efficiently utilize this additional structural information, which is not common property, as shown by the CoPINet’s performance deterioration in the case of auxiliary training with dense encoding. Generally speaking, for all three models, MLCL clearly outperforms CE with auxiliary training with dense encoding (CE + AUX-dense). Moreover, MLCL results are on-par with CE + AUX-sparse for SCL and CoPINet, and

| Method | DA | Mean test accuracy ± Std (%) | SCL | SRAN | CoPINet |
|--------|----|-----------------------------|-----|------|---------|
| CE     | ✓  | 82.8 ± 0.7                  | 56.4 ± 1.8 | 44.8 ± 0.8 |
|        | ✓  | 90.8 ± 1.3                  | 65.8 ± 0.7 | 48.3 ± 1.8 |
| CE + AUX | dense | ×  | 86.9 ± 1.5  | 59.4 ± 0.7 | 32.2 ± 1.9 |
|        | ✓  | 95.3 ± 0.3                  | 69.1 ± 1.8 | 35.2 ± 1.2 |
| CE + AUX | sparse | ×  | 95.6 ± 0.5  | 59.9 ± 1.6 | 50.3 ± 1.0 |
|        | ✓  | 95.9 ± 0.5                  | 72.2 ± 1.3 | 49.2 ± 1.2 |
| MLCL   | dense | ×  | 88.1 ± 0.9  | 57.6 ± 1.2 | 33.4 ± 1.7 |
|        | ✓  | 94.8 ± 0.4                  | 64.0 ± 1.3 | 48.8 ± 1.2 |
| MLCL   | sparse | ×  | 95.7 ± 0.5  | 66.1 ± 1.6 | 52.3 ± 1.1 |
|        | ✓  | 96.8 ± 0.4                  | 73.3 ± 1.0 | 57.1 ± 1.4 |
| Wu et al. [6] | ×  | 95.0          | -   | -   |         |
| Hu et al. [7] | ×  | -           | 60.8 | 46.1 |         |

TABLE I TEST ACCURACY ON THE I-RAVEN DATASET AVERAGED ACROSS FOUR RANDOM SEEDS. RESULTS ARE REPORTED FOR THREE DIFFERENT ENCODER NETWORKS AND FIVE DIFFERENT TRAINING SETUPS, IN EACH CASE WITHOUT AND WITH DA, RESPECTIVELY. THE LAST TWO ROWS PRESENT BEST RESULTS REPORTED IN THE LITERATURE

| Method    | DA | Mean test rule prediction F1 ± Std (%) | SCL | SRAN | CoPINet |
|-----------|----|----------------------------------------|-----|------|---------|
| CE+ AUX   | ✓  | 99.7 ± 0.0                             | 87.2 ± 1.2 | 94.0 ± 0.4 |
| sparse    | ✓  | 99.8 ± 0.0                             | 93.7 ± 1.3 | 94.0 ± 0.8 |
| MLCL      | ✓  | 99.7 ± 0.1                             | 90.6 ± 0.2 | 93.0 ± 0.9 |
| sparse    | ✓  | 99.9 ± 0.0                             | 93.8 ± 0.8 | 94.9 ± 0.4 |

TABLE II TEST RULE PREDICTION F1-SCORE ON THE I-RAVEN DATASET AVERAGED ACROSS FOUR RANDOM SEEDS. RESULTS ARE REPORTED FOR THREE DIFFERENT ENCODER NETWORKS TRAINED WITH CE + AUX-SPARSE AND MLCL-SPARSE, WITHOUT AND WITH DA, RESPECTIVELY
visibly better for SRAN. We conclude that MLCL exploits the supplementary training signal more efficiently than the current state-of-the-art auxiliary training method (CE + AUX-dense).

3) (+DA): When DA is included in the training process, MLCL is the preferred method for all three ML models, which suggests that the augmentation is exploited most efficiently in the proposed contrastive method. This aligns with observations reported throughout the contrastive representation learning literature, where data transformation modules were identified as crucial components [24], [34], [36]. Most notably, by employing MLCL with DA we improve the originally reported accuracy of SRAN to 73.3%, CoPINet to 57.1%, and SCL to 96.8%. The last of the above-mentioned outcomes sets a new state-of-the-art result on the I-RAVEN dataset.

Moreover, Table II shows that DA is also beneficial for learning to discover the abstract rules, resulting in 6.5 p.p. and 3.2 p.p. improvements for SRAN trained with CE + AUX and MLCL, respectively, and in 1.9 p.p. improvement for CoPINet trained with MLCL.

B. Results on PGM
Table III presents results on the PGM dataset. Due to a huge number of RPM instances, we compare the results of MLCL only with the best-performing encoder network, i.e., SCL. We evaluate MLCL in two setups, with and without FT. In the most demanding regime (H.O. shape-color) all training methods achieve close-to-random results. MLCL achieves superior results in total in four regimes, and in the remaining cases, the best outcomes are accomplished by SCL supervised training supported with an auxiliary loss with either dense (two regimes) or sparse (one regime) encoding. MLCL outperforms the base setup (CE) in six regimes.

The experiments conducted on PGM show that by applying the optional FT stage at the end of MLCL training, SCL’s performance increases. Specifically, after FT the results improved in five PGM regimes, most notably by 17.5% in the neutral regime. Encouraged by this improvement, the effects of extending MLCL with FT are further explored in the ablation study on I-RAVEN in Section IV-C.

It is interesting to observe that while FT improves the performance of SCL on Val. split in each regime, this improvement does not necessarily imply better accuracy on the Test. split (see H.O. triples and extrapolation regimes). While in the neutral regime the performance boost on the Val. split is correlated with an improvement on the Test. split (because both splits have the same data distribution), this no longer holds for the remaining regimes that focus on out-of-distribution generalization. This attracts attention to an important characteristic of the PGM dataset—although current methods are able to achieve high performance on the Val. splits, more work has to be done to effectively transfer this knowledge to out-of-distribution samples that are present in the Test. splits.

Though MLCL has shown to be a well-rounded method by achieving the best results in half of PGM regimes, it is clear that PGM poses a unique challenge, which is currently far from being solved by a single method. Moreover, the current approaches especially struggle in certain more demanding regimes, where their performance might be even indistinguishable from random guessing. These issues are exacerbated by the huge size of the PGM dataset, which makes the assessment of all the proposed methods, in terms of their unique generalization capabilities, even more difficult.

Our results show that as many regimes as possible should be considered when evaluating RPM solvers on PGM, as the performance of each of the tested methods varies significantly across individual regimes.

C. Ablation Study
In the ablation study, we further validate the role of the MLCL framework as an auxiliary training method and analyze its contrastive properties on the I-RAVEN dataset.

1) Joint Optimization: MLCL combines both contrastive and auxiliary losses, as shown in (8). We have verified that using either one of its individual components alone is not sufficient for building strong representations. When using contrastive loss only, i.e., with $\beta = 0$, we observed a serious performance downgrade for all models (see $L_{mlc}$ in Table IV). We hypothesize that the lack of auxiliary training information makes it difficult to extract abstract rules and encourages focusing too much on the visual similarity of RPMs, which is unprofitable for the final downstream task. Similarly, setting
coefficient settings depending on: 1) balancing factors in the definition of multilabel contrastive loss without and with DA and 2) the batch size used for the contrastive pretraining stage of MLCL without and with DA. (a) Coefficients (-DA), (b) Coefficients (+DA), (c) Batch size (-DA), (d) Batch size (+DA).

Fig. 5. MLCL ablation study on the I-RAVEN dataset. The figures present differences in the final classification performance averaged across four random seeds depending on: 1) balancing factors in the definition of multilabel contrastive loss without and with DA and 2) the batch size used for the contrastive pretraining stage of MLCL without and with DA. (a) Coefficients (-DA), (b) Coefficients (+DA), (c) Batch size (-DA), (d) Batch size (+DA).

Analogously, the methods which support multiple positive samples tend to benefit from a higher number of positive samples while calculating the loss [24]. In fact, one can see a certain drop in the final performance for larger batch sizes, which some works tried to tackle by storing additional negative samples in a memory bank [41] or using a dynamically updated queue with a moving-average encoder [32], [33]. Contrary to the above-mentioned findings, empirical evaluation of MLCL [Fig. 5(c) and (d)] shows that the method does not require large batch sizes and surpasses the performance of traditional supervised training under the same experimental protocol. In fact, one can see a certain drop in the final performance for larger batches across all models, which suggests their negative influence on the auxiliary training.

In the final linear classification stage, the main goal is to select a correct answer, which requires discriminating between correctly and incorrectly completed RPMs. We analyze the importance of using embeddings of RPMs with incorrect answers as additional negative samples, by removing the term \( \Sigma_{i,j} \) from the denominator of (6). Unavailability of these RPMs results in a notable drop of performance on the downstream task (see w/o \( L_{\text{mlc}}^{i,j} \) in Table IV). In fact, when this additional signal is ignored, the encoder network only learns how to differentiate between correctly completed RPMs governed by different sets of rules, whereas the final classification task requires differentiating between a correct RPM and a set of incorrectly completed RPMs. We conclude that this additional training signal obtained from RPMs with incorrect answers is mandatory for achieving high downstream task performance.

3) FT and DA on I-RAVEN: We have shown that models trained with MLCL with linear evaluation are at least competitive with alternative training approaches. In order to further evaluate the potential of representations learned with MLCL, we explore the impact of an optional FT phase, where the encoder \( f_0 \) is unfrozen in the final training stage. These experiments additionally test model performance under different DA settings, by removing selected transformations from the DA module during the pretraining stage. The results are summarized in Table V.

In the majority of model/DA configurations, FT leads to better final classification performance. For SRAN and CoPINet, this additional training stage further improves their test accuracy by around 1 pp, to 74.1% and 58.0%, respectively.

\( \gamma = 0 \), decreased the accuracy of all tested models even more (see \( L_{\text{mlc}}^{i,j} \) in Table IV). This suggests that with purely auxiliary training, models are unable to relate the same abstract relationships to different visual figure configurations. Moreover, the absence of contrastive loss significantly hinders the ability to discriminate between correctly and incorrectly completed RPMs. These observations stress the importance of using joint loss, i.e., setting both \( \beta > 0 \) and \( \gamma > 0 \).

In Fig. 5(a) and (b), we further validate the choice of balancing coefficients by fixing \( \gamma = 1 \) and varying \( \beta \). It is revealed that a correct choice is critical to the final performance. In the experiments without DA, too small, as well as too big values of \( \beta \) have a negative impact on the accuracy of all models and the best results are achieved with \( \beta = 10 \). When DA is applied, small values of \( \beta \) once again hinder the performance, however, the results for bigger \( \beta \) values are less conclusive, and depending on the model the best outcomes are accomplished with either \( \beta = 10 \) or \( \beta = 15 \).

2) Contrast Strength: The quality of representations learned with the contrastive mechanism benefits from applying bigger contrast to positive samples, which is realized by including additional negative pairs while calculating the loss [24]. Analogously, the methods which support multiple positive samples tend to benefit from a higher number of positive pairs in a given batch [34]. These two properties of contrastive approaches result in the reliance on large batch sizes, which some works tried to tackle by storing additional negative samples in a memory bank [41] or using a dynamically updated queue with a moving-average encoder [32], [33]. Contrary to the above-mentioned findings, empirical evaluation of MLCL [Fig. 5(c) and (d)] shows that the method does not require large batch sizes and surpasses the performance of traditional supervised training under the same experimental protocol.
Table V: Test accuracy on the I-RAVEN dataset averaged across four random seeds. Results are reported for three different encoder networks trained with MLCL with or without FT and five DA setups, where base denotes default DA setup used in the main experiments presented in this article.

| DA                      | FT | Mean test accuracy ± Std (%) | SCL | SRAN | CoPInet |
|-------------------------|----|-----------------------------|-----|------|---------|
| base                    | ☒  | 96.8 ± 0.4                  | 73.3 ± 1.0 | 57.1 ± 1.4 |
|                         | ☑  | 96.6 ± 1.2                  | 74.1 ± 1.3 | 58.0 ± 1.1 |
| w/o roll                | ☒  | 95.6 ± 1.6                  | 71.5 ± 1.9 | 55.6 ± 2.0 |
|                         | ☑  | 96.2 ± 1.6                  | 71.6 ± 2.2 | 57.2 ± 1.2 |
| w/o rotation            | ☒  | 96.3 ± 1.0                  | 73.1 ± 2.2 | 57.2 ± 2.5 |
|                         | ☑  | 96.2 ± 0.7                  | 72.9 ± 2.0 | 57.1 ± 1.8 |
| w/o shuffle             | ☒  | 94.8 ± 1.4                  | 68.5 ± 1.3 | 50.4 ± 1.8 |
|                         | ☑  | 96.1 ± 1.5                  | 73.0 ± 1.7 | 56.1 ± 2.1 |
| w/o roll, rotation, shuffle | ☒  | 95.4 ± 1.0                  | 64.0 ± 2.7 | 49.3 ± 2.2 |
|                         | ☑  | 96.1 ± 1.1                  | 70.1 ± 1.7 | 51.8 ± 0.5 |

Interestingly, the experiments indicate that FT is able to compensate to some degree for the lack of diverse augmentation methods. While for all three models the removal of certain data transformations during the pretraining stage decreases MLCL performance when tested with linear evaluation, FT is able to push their performance close to the results obtained with their best training setups.

In summary, the experiments show that on I-RAVEN the models trained with MLCL, default DA, and tested under the linear evaluation protocol are on-par (within a standard deviation of the mean) with their FT versions.

4) Learning Dynamics: To better showcase the intricacies of MLCL training dynamics, Fig. 6 compares the validation accuracy of SCL trained for 100 epochs on I-RAVEN with CE + AUX and MLCL, both using sparse encoding and DA. While CE + AUX keeps improving the accuracy right from the beginning, MLCL starts with the pretraining stage where the accuracy is essentially random and then immediately increases close to the model’s highest capacity in the linear evaluation stage. This shows that the pretraining stage, which uses the multilabel contrastive loss, produces high-quality embeddings, which are well-suited for the final classification task.

Due to the method design, it is natural that MLCL requires more time for each training step. While traditional training setup only performs one forward pass for a batch of RPMs, MLCL performs two forward passes for a given batch—one pass per each augmented view. In practice, this is implemented as one forward pass with a batch twice as big, but DA has to be performed twice to obtain two random views of each RPM from the batch. This results in increased computation time. Training the model from Fig. 6 for 100 epochs with CE + AUX took 142 min in total, whereas MLCL required 253 min on the same hardware. During pretraining (65 epochs), each MLCL epoch took 186 s on average, which was reduced to 88 s on average for the remaining part of the training (35 epochs). In comparison, each epoch of CE + AUX required 84 s on average.

In order to check whether a longer training time for CE + AUX could reduce the gap to MLCL, we have removed the early stopping mechanism and trained all three models with CE + AUX-sparse for 300 epochs (this took on average 425 min for each run of SCL). However, none of the models was able to achieve improved performance given the additional time. We observed that, in general, the performance of all three models stagnates before the 80th epoch, which suggests that the better results achieved with MLCL come not from additional training time, but rather from better exploitation of the training signals.

In summary, MLCL utilizes the same training signals like those used in prior works. While the previous state-of-the-art models were generally unsuccessful in utilizing the dense rule encoding to improve performance, MLCL with sparse encoding and DA, introduced in this article, efficiently exploits the training signals provided by the rule annotations, which leads to better performance. Also, the increase in computation time happens only during training. The inference time of a model trained with MLCL is the same as of a model trained with CE or CE + AUX.

MLCL belongs to a family of pretraining methods which follow a similar training scheme: first, a general representation model is trained, which is then fine-tuned on chosen target tasks. Notable examples can be found in the domains of natural language processing [42], computer vision [24], or game playing [27]. Despite the wide adoption of pretraining approaches, especially those based on contrastive learning, so far no such method has been proposed for the AVR domain. MLCL provides a general framework for how such multistage training could be performed for AVR tasks. We believe that building such universal representation models for the AVR domain is a promising direction for future research.

V. Conclusion

In this article, we propose a novel NCE algorithm suitable for multilabel samples and integrate it with auxiliary training to devise a new ML approach (MLCL) to abstract
visual reasoning tasks. The efficacy of MLCL is tested on a challenging task of solving RPMs which is formulated in this article as a multilabel classification problem with the 1–1 correspondence between labels and abstract rules underlying a given RPM. The proposed approach establishes new state-of-the-art results on the I-RAVEN dataset and demonstrates superior performance in four regimes from PGM. The MLCL framework is additionally supported by a sparse rule encoding scheme of the RPMs proposed in this article, which on the I-RAVEN dataset consistently outperforms the encoding used in prior works.

APPENDIX A
EXTENDED PROBLEM DESCRIPTION

Due to the space limits, in the main text, we have introduced the problem of solving Raven’s progressive matrices (RPMs) only briefly. Here, we provide additional details and in particular discuss the differences between rule encoding schemes used in the two considered datasets (I-RAVEN and PGM). In both of them, the goal is to complete a 3 × 3 matrix with a missing panel in the bottom-right corner. The answer has to be chosen from a set of eight candidates, such that the chosen image satisfies all abstract rules defining a given RPM. In each problem instance, there is only one answer which correctly completes the matrix, while the remaining answers usually satisfy some (but not all) of the underlying rules.

The sets of possible rules differ between the datasets. In I-RAVEN a set of rules associated with a given RPM (called an abstract structure) is defined as a set of pairs \( S = \{[r, a] \mid r \in \mathcal{R}, a \in \mathcal{A}\} \), where \( \mathcal{R} = \{\text{Constant, Progression, Arithmetic, Distribute, Three}\} \) is the set of relation types and \( \mathcal{A} = \{\text{Number, Position, Type, Size, Color}\} \) is the set of attributes types.

The abstract structure of RPMs from the PGM dataset is defined as a set of triples \( S = \{[r, o, a] \mid r \in \mathcal{R}, o \in \mathcal{O}, a \in \mathcal{A}\} \), where \( \mathcal{R} = \{\text{progression, XOR, OR, AND, consistent union}\} \) is the set of relation types, \( \mathcal{O} = \{\text{shape, line}\} \) is the set of object types and \( \mathcal{A} = \{\text{size, type, color, position, number}\} \) is the set of attribute types.

In order to better present the problem, let us consider the first example of an RPM from the main text (Fig. 7). Searching for common patterns among the images leads us to the following observation: in both the top row and the middle row images, the octagon shapes in the right column images appear only in positions where there are octagons in both the left and the middle column images. Consequently, the respective rule \((s = \{\text{shape, position, AND}\})\) is potentially a part of the underlying abstract structure associated with this RPM, i.e., one may suppose that \( s \in S \). After applying this rule to the bottom row, one would expect the missing image to have only a single octagon, located in the top-right corner.

In fact, only a single image (with label A) from the set of potential candidates matches this description. Furthermore, the remaining shape attributes (size, color, and number), as well as various combinations of lines in this RPM do not satisfy any of the allowed rules from \( S \) (actually their role is to distract the solver). The above-mentioned observations lead us to a conclusion that for the considered RPM \( S = \{\text{shape, position, AND}\} \) and that the correct answer is A.

Observe that in the described example, the rule was applied rowwise, however, in general, the RPMs from the PGM dataset may have rules applied either rowwise or columnwise, whereas rules from the I-RAVEN dataset are only applied rowwise.

Further examples of RPMs together with explanation of their solutions are presented in Figs. 8–10. We refer the readers interested in a more detailed description of both datasets to the original works of Barrett et al. [4] and Zhang et al. [5].

APPENDIX B
SPARSE VERSUS DENSE RULE ENCODING

Let us now revisit the two rule encoding schemes discussed in the article. Barrett et al. [4] proposed to encode the abstract rules as binary strings of length 12 (called meta-targets) according to the following syntax: \((\text{shape, line, color, number, position, size, type, progression, XOR, OR, AND, consistent union})\). In order to support RPMs with multiple rules, the meta-target for a given RPM is obtained by performing an OR operation on the set of encoded individual rules.

For example, for an RPM instance with rules \( S = \{\{\text{OR, shape, type}, \text{AND, line, color}\}\} \), the method, referred to as dense encoding in this article, yields the following meta-target: \([\text{OR}(1000001000100), 011000000010] = [1110000100110] \). Based on the resultant string, one is able to conclude that the underlying structure consists of OR and AND relations, shape and line objects, and type and
Fig. 8. Augmented RPM from the I-RAVEN dataset with configuration Center. In each of two top rows, there is a single shape in each image ([Constant, Number]), located in the same position ([Constant, Position]), the shapes are of the same type [Constant, Type], and size [Constant, Size]. In each row there are shapes in three different colors ([Distribute_Three, Color]). This leads to an underlying abstract structure $S = \{[\text{Constant}, \text{Number}], [\text{Constant}, \text{Position}], [\text{Constant}, \text{Type}], [\text{Constant}, \text{Size}], [\text{Distribute_Three, Color}] \}$, which is realized by completing the matrix with answer D. (a) Base. (b) Horizontal flip. (c) Horizontal roll. (d) Shuffle $2 \times 2$.

Fig. 9. Augmented RPM from the I-RAVEN dataset with configuration O-IC. In this configuration, both inner and outer structures can be governed by different sets of rules. Looking at the outer structures, one can see that each panel is composed of a single object ([Constant, Number]) at the central position ([Constant, Position]) of the same color ([Constant, Color]). In each row, the outer structures have one out of three possible sizes and types—hexagon, pentagon, or a square ([Distribute_Three, Size] and [Distribute_Three, Type]). This gives an outer structure defined as $S_{\text{outer}} = \{[\text{Constant}, \text{Number}], [\text{Constant}, \text{Position}], [\text{Distribute_Three, Type}], [\text{Distribute_Three, Size}], [\text{Constant, Color}] \}$. Similarly, the inner structure of each image consists of single objects ([Constant, Number]) located in the center ([Constant, Position]). The objects differ in type, with circles in the left column, pentagons in the middle one and triangles in the right column ([Progression, Type]). The objects in inner structures of each row have three different sizes and colors ([Distribute_Three, Size] and [Distribute_Three, Color]), which leads to a final outer structure $S_{\text{inner}} = \{[\text{Constant, Number}], [\text{Constant, Position}], [\text{Progression, Type}], [\text{Distribute_Three, Size}], [\text{Distribute_Three, Color}] \}$. The only answer which matches all rules for both inner and outer structures is D. (a) Base. (b) Vertical flip. (c) Vertical roll. (d) Shuffle $3 \times 3$.

color attributes. However, recovering the exact relations governing the considered RPM is not possible.

The new rule encoding scheme proposed in this article referred to as sparse encoding, is more explicit and allows to unambiguously retrieve the encoded relations, with the aim of providing a more accurate training signal. Since the set of all possible abstract structures in PGM is composed of 50 elements, i.e., there are $|\mathcal{R}| \times |\mathcal{O}| \times |\mathcal{A}| = 2 \times 5 \times 5 = 50$ unique rules, our method encodes them as a one-hot vector of length 50. Similar to the dense encoding, we perform an $\text{OR}$ operation on the set of all rules for a given RPM. However, due to the nature of one-hot encoding, all information about the underlying abstract structure is preserved.

Analogously to PGM, the rule encoding method for RPMs from the I-RAVEN dataset, proposed by Zhang et al. [5], represents each rule as a multihot vector of length $|\mathcal{R}| + |\mathcal{A}| = 9$ and combines the set of individual rule encodings by means of an $\text{OR}$ operation. In effect, similar to
the case of PGM, recovering individual rules constituting the
final encoded representation is not possible. This problem is
further exacerbated by the generally higher numbers of rules
in I-RA VEN instances. RPMs in the I-RA VEN dataset contain
6.29 rules on average, compared with 1.37 rules (on average)
in RPMs from the PGM dataset [5]. Consequently, for
RPMs from the I-RA VEN dataset, the resultant representation
obtained with dense encoding is highly ambiguous.

Sparse rule representations of I-RA VEN RPMs are obtained
analogously to the PGM ones. In this dataset, there are
|\mathcal{R}| \times |\mathcal{A}| = 4 \times 5 = 20 unique configurations of rules and attributes. However, Zhang et al. [5] pointed out that applying Arithmetic to Type is counterintuitive. Additionally, in some configurations, the rules can be applied to both component structures. Namely, each of the configurations L-R, U-D, O-IC, and O-IG consists of two distinct substructures: left/right, up/down, outer/inner, and outer/inner, respectively. This gives a total of 38 unique rules, which is the length of our one-hot vector representation. Again, due to the nature of one-hot encoding, all information related to the abstract structure is preserved after applying the OR operation.

In summary, although the advantage of sparse encoding for
PGM data is demonstrated only in certain regimes, it consist-
tently outperforms the previous (dense) encoding method on
I-RA VEN, due to the significantly higher average number of
rules per RPM instance in this dataset.

APPENDIX C
DATA AUGMENTATION

To the best of our knowledge, this work is the first to
investigate the use of data augmentation (DA) strategies in
solving RPMs. The augmentation process is described in
the main body of this article, in Section III-C. and Fig. 3.
Additional examples of augmented RPMs are presented in
Figs. 8–10. Each figure, from left to right, presents: 1) the
base RPM from either I-RA VEN or PGM; 2) the RPM after
applying the horizontal/vertical flip augmentation; 3) the RPM
after roll operation, which rolls an image along the vertical
axis, horizontal axis, or both axes simultaneously; and 4) the
RPM after grid shuffle, which splits an image into a 2 \times 2
or 3 \times 3 grid and randomly shuffles its components. The biggest
structural changes are introduced to the images after applying
the roll operation (the third column in each figure) and the
grid shuffle transformation (the last column in each figure).
Even though the images become more difficult to decipher
for human solvers, they greatly diversify the training set and
increase the overall number of training instances.

APPENDIX D
IMPLEMENTATION DETAILS

All hyperparameters reported in this article were chosen
without any extensive search, based on manual convergence
analysis of a limited number of runs. In all experiments on
both I-RA VEN and procedurally generated matrices (PGM)
datasets, we have trained models on images rescaled to the
size 80 \times 80, following the setup from [4]. This allowed us
to overcome hardware memory limitations, train with larger
batches, and conduct extensive comparisons. For the ease
of reproducibility, the experiments are structured as PyTorch
lightning modules [43] and made accessible under the follow-
ing URL: https://github.com/mikomel/mlcl.

REFERENCES

[1] J. C. Raven, “Mental tests used in genetic studies: The performance of
related individuals on tests mainly educative and mainly reproductive,”
M.S. thesis, Univ. London, London, U.K., 1936.
[2] J. C. Raven and J. H. Court, Raven’s Progressive Matrices and Vocab-
ulary Scales. Oxford, U.K.: Oxford Psychologists Press, 1998.
[3] P. A. Carpenter, M. A. Just, and P. Sholl, “What one intelligence
test measures: A theoretical account of the processing in the raven
progressive matrices test,” Psychol. Rev., vol. 97, no. 3, p. 404, 1990.
L. Smith and D. Gentner, “The role of difference-detection in learning,” in Proc. 35th Int. Conf. Mach. Learn., vol. 80, 2018, pp. 511–520.

C. Zhang, F. Gao, B. Jia, Y. Zhu, and S.-C. Zhu, “RAVEN: A dataset for relational and analogical visual reasoning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 5317–5327.

Y. Wu, H. Dong, R. Grosse, and J. Ba, “The scattering compositional learner: Discovering objects, attributes, relationships in analogical reasoning,” 2020, arXiv:2007.04122.

S. Hu, Y. Ma, X. Liu, Y. Wei, and S. Bai, “Stratified rule-aware network for abstract visual reasoning,” in Proc. AAAI Conf. Artif. Intell., vol. 35, no. 2, 2021, pp. 1567–1574.

P. Zhang, Y. Goyal, D. Summers-Stay, D. Batra, and D. Parikh, “You and yang: Balancing and answering binary visual questions,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 5014–5022.

J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zimick, 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. 2901–2910.

D. Gentner, “Structure-mapping: A theoretical framework for analogies,” Cognit. Sci., vol. 7, no. 2, pp. 155–170, 1983.

D. R. Hofstadter, Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought. New York, NY, USA: Basic Books, 1995.

L. Smith and D. Gentner, “The role of difference-detection in learning contrastive categories,” in Proc. Annu. Meeting Cognit. Sci. Soc., vol. 36, 2014. [Online]. Available: https://escholarship.org/uc/item/7i0ts5k0j

C. Zhang, B. Jia, F. Gao, Y. Zhu, H. Lu, and S.-C. Zhu, “Learning perceptual inference by contrasting,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 1075–1087.

F. Hill, A. Santoro, D. Barrett, A. Morcos, and T. Lillicrap, “Learning to make analogies by contrasting abstract relational structure,” in Proc. Int. Conf. Learn. Represent., 2019. [Online]. Available: https://openreview.net/forum?id=SylYYrCefn

D. Hoshen and M. Werman, “IQ of neural networks,” 2017, arXiv:1710.01692.

J. Mandziuk and A. Zychowski, “DeepIQ: A human-inspired AI system for solving IQ test problems,” in Proc. Int. Joint Conf. Neural Netw. (IJCNN), Jul. 2019, pp. 1–8.

K. Zheng, Z.-J. Zha, and W. Wei, “Abstract reasoning with distracting features,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 5842–5853.

A. Santoro et al., “A simple neural network module for relational reasoning,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 4967–4976.

D. Wang, M. Jannik, and P. Lio, “Abstract diagrammatic reasoning with multiplex graph networks,” in Proc. Int. Conf. Learn. Represent., 2020. [Online]. Available: https://openreview.net/forum?id=ByxQBIrKwH

J. E. Hummel and K. J. Holyoak, “Distributed representations of structure: A theory of analogical access and mapping,” Psychol. Rev., vol. 104, no. 3, p. 427, 1997.

A. Lovett and K. Forbus, “Modeling visual problem solving as analogical reasoning,” Psychol. Rev., vol. 124, no. 1, pp. 60–90, 2017.

M. Gutmann and A. Hyvärinen, “Noise-contrastive estimation: A new estimation principle for unnormalized statistical models,” in Proc. 15th Int. Conf. Artif. Intell. Statist., 2010, pp. 297–304.

A. Mnih and K. Kavukcuoglu, “Learning word embeddings efficiently with noise-contrastive estimation,” in Proc. Adv. Neural Inf. Process. Syst., 2013, pp. 2265–2273.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in Proc. 37th Int. Conf. Mach. Learn. (PMLR), vol. 119, A. Singh, Ed. Jul. 2020, pp. 1597–1607.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in Proc. Adv. Neural Inf. Process. Syst., 2013, pp. 3111–3119.

T. Kipf, E. van der Pol, and M. Welling, “Contrastive learning of textual representations,” in Proc. Int. Conf. Learn. Represent., 2020. [Online]. Available: https://openreview.net/forum?id=H1gaXTvID

M. Laskin, A. Srivivas, and P. Abbeel, “CURL: Contrastive unsupervised representations for reinforcement learning,” in Proc. 37th Int. Conf. Mach. Learn., vol. 119, 2020, pp. 5639–5650.

A. van den Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” 2018, arXiv:1807.03748.

R. Hjelm et al., “Learning deep representations by mutual information estimation and maximization,” in Proc. Int. Conf. Learn. Represent., 2019. [Online]. Available: https://openreview.net/forum?id=Bklr3j0cKX