Advances in Cognitive Systems X (20XX) 1-6 Submitted X/20XX; published X/20XX

Deep Learning of Raven's Matrices

Can Serif Mekik MEKIKC@RPI.EDU
Ron Sun RSUN@RPI.EDU
Department of Cognitive Science, Rensselaer Polytechnic Institute, Troy, NY 12180

David Yun Dai YDAI@ALBANY.EDU
Department of Educational and Counseling Psychology, State University of New York at Albany, Albany, NY 12222

Abstract

In this paper, we develop a model of performance on Raven's Matrices—a commonly used test of human intelligence. Raven's matrices are presented in a visual form but require high-level (e.g., rule-based) reasoning. Integrating high-level reasoning and visual processes is a challenge for models of Raven's matrices. To address this, we present a new approach to modeling Raven's matrices which integrates deep learning networks and rule-based reasoning. Matrix problems require subjects to complete a pattern of visual figures by selecting one figure, among a set of alternatives, which best completes the pattern. Our model solves matrices by progressively observing the layout of their visual features and detecting changes between pairs of visual figures using a deep neural network. These changes are used to infer the distribution of visual features in the matrix and to evaluate each alternative for a fit using a set of rules. This model constitutes only a first step towards eventually developing a psychologically realistic model of human performance on Raven's Matrices.

1. Introduction

The Raven's progressive matrices (RPM) tests are a family of intelligence tests which are composed of sequences of increasingly difficult items (Raven et al., 1998). Each item consists of two arrays of visual figures, the *matrix* and the *answer array*. The matrix is a square array of visual figures that is missing a part. The answer array contains several figures, one of which best completes the matrix. Subjects are instructed to pick the answer array figure which best completes the accompanying matrix; an item is complete when a figure is selected. RPM were designed for the measurement of *eductive ability*, which is the ability to make sense of complex novel stimuli (also known as *fluid intelligence*), but they have since been found to also provide reliable measurements of general intelligence. Several versions of the tests exist, each optimizing the format's discriminative power for specific assessment scenarios. The standard version of RPM, the *Standard Progressive Matrices* (SPM), are optimized for general assessment of children and adults. Two alternative versions of the test, the *Coloured Matrices* and the *Advanced Matrices*, are designed for increased discriminative power in the lower and higher ranges of the scoring spectrum respectively.

Computational cognitive models of intelligence tests are seen to be a promising avenue of research both for improving our understanding of the psychological demands of intelligence tests and for basic research in artificial intelligence (Hernández-Orallo et al., 2016). Since eductive ability and cognitive processes are both thought to be biologically determined, RPM has attracted attention from modelers seeking to understand the contributions of basic cognitive processes to general intelligence (see e.g., Carpenter et al., 1990). Modeling research has targeted the cognitive mechanisms enabling RPM performance as well as sources of variability in performance. In what concerns mechanisms, modelers have investigated working memory (Carpenter et al., 1990; Ragni & Neubert, 2014), rule induction (Carpenter et al., 1990; Rasmussen & Eliasmith, 2014), analogical reasoning (Lovett & Forbus, 2017) and iconic processing (Kunda et al., 2013; McGreggor et al., 2014). As for variability in performance, modelers have investigated differences between high versus low performers (Carpenter et al., 1990), item difficulty (Carpenter et al., 1990; Ragni & Neubert, 2014; Lovett & Forbus, 2017), cognitive decline (Rasmussen & Eliasmith, 2014), and differences between typically developing versus autistic subjects (Kunda et al., 2010). Some aspects of task performance that have seen little or no attention include learning, the role of embodiment, basic perceptual processes, and hybrid models combining sub-symbolic and symbolic processing. We are interested in extending the RPM modeling tradition in these new directions.

A significant challenge in RPM modeling is due to the task's visual format. This format has prompted two prominent approaches to RPM model design, each with its own limitations. The majority of implementations have foregone modeling basic perceptual processes involved in RPM and focused on symbolic or analogical reasoning over abstract representations of RPM stimuli (e.g., Ragni & Neubert, 2014). However, foregoing basic perceptual processes has not prevented modelers from incorporating higher-order perceptual processes. To date, modelers following the first approach have prepared abstract representations by annotating matrix items; that is to say, in many RPM models, human modelers/annotators have already performed much of the basic perceptual work required to complete the task. In contrast, a minority of modelers have opted to implement algorithms which solve RPM using transformations and similarity metrics on raw images (pixel arrays) and to forgo symbolic processing (e.g., McGreggor et al., 2014). While this latter approach has been successful from a computational point of view, these models appear to be incomplete from a psychological point of view. This is because these models omit explicit processing of abstract visual features, such as shapes, textures, and symmetry, which are believed to involve symbolic representations. It is generally accepted that subjects reason about abstract visual features using inferential or analogical processes while solving matrix problems (Carpenter et al., 1990; Lovett & Forbus, 2017; Rasmussen & Eliasmith, 2014). An important experimental study in this regard is that of Primi (2001), which shows that many difficult RPM problems require explicit control of perceptual organization—that is to say, explicit selection of the abstract features used to characterize matrix elements.

The challenge presented by the visual format of the RPM is modeling the processes of visual perceptual abstraction required to obtain the kinds of abstract visual representations on which inferential or analogical processes recruited by the task depend. This challenge is a difficult image understanding problem which is addressed neither by models relying on annotated stimuli nor by models which operate on raw images: the former are spared from having to tackle the task thanks

to human annotators while the latter avoid it by design. A realistic model of the basic perceptual processes responsible for generation of the abstract visual representations humans use in solving RPM problems can place important constraints on the structure of these representations, and consequently on the nature of the cognitive processes involved in task performance, which may otherwise be left unaddressed. We are therefore interested in exploring a new direction in RPM modeling by developing a model which simulates the processes of perceptual abstraction from raw pixel inputs up to abstract visual features without any interventions by human annotators. We will discuss what such an approach may contribute to the RPM modeling literature in Section 4.

Deep learning is a promising paradigm for this new approach. Deep neural networks are notable for their success in image understanding tasks (classification, labeling, segmentation etc.; LeCun et al., 2015), and they stand out for their ability to achieve high performance in these tasks from pixel-level input. A striking example of the success of deep learning methods on tasks with pixel-level inputs is the work of Mnih et al. (2013), who trained deep neural networks to learn successful control policies on a range of Atari games from raw video input. Another relevant feature of deep neural networks is their ability to learn hierarchical abstract representations of data (LeCun et al., 2015). Deep convolutional neural networks are a common architecture for image processing applications; these networks have a feed-forward architecture consisting of a sequence of layers of artificial neurons, where each layer can be viewed as the repeated application at different locations in the image (convolution) of a set of local nonlinear filters. Layers in convolutional neural networks tend to learn progressively more complex visual features, thus early layers tend to learn simple edge detectors whereas units in later layers learn to detect features such as faces (see Zeiler & Fergus, 2014, for visualizations). Finally, it is also worth mentioning that convolutional neural network architectures are themselves inspired by the architecture of biological visual neural networks. The correspondence between biological visual neural networks and deep convolutional neural networks is, in practice, rather loose; nevertheless, a deep learning approach to RPM can approximate task-related low-level perceptual processes more closely than other existing approaches. Moreover, this approximation can be improved by incorporation of additional known biological constraints on the architecture of visual neural networks in order to serve as realistic neural models of visual processing (Kriegeskorte, 2015). Put together, the considerations above suggest that deep learning is a promising way to avoid manual annotation of matrix images, while, at the same time, enabling models to extract and process abstract visual features.

Here, we present a new model of the human RPM performance which integrates a convolutional neural network with rule-based reasoning. Our model, which is a first step towards ultimately developing a more psychologically realistic model, integrates both visual (subsymbolic) and symbolic abilities: its perceptual apparatus, a deep convolutional neural network, allows it to abstract matrix features which then inform its strategy, implemented in the form of a simple rule set. This model introduces three novelties to the RPM modeling literature. To our knowledge, our model is the first learning-based model of RPM, it is the first model of RPM that makes use of deep convolutional neural networks, and it is also the first model of RPM to combine sub-symbolic and symbolic processing. Finally, our model emphasizes the constraints of embodiment; we discuss this latter point in greater detail below.

Note also that, instead of the original Raven's matrices, we used matrices from Matzen et al. (2010). Matzen et al. developed matrix generation software based on an analysis of SPM item structure which was shown, in a norming study, to generate matrices that have psychometric properties comparable to the original SPM problems. Since the generation software was developed at Sandia National Laboratory, we call matrices generated with this software *Sandia matrices*. We chose to use Sandia matrices instead of original Raven's matrices for three major reasons. First, RPM tests offer only a limited number of test items (e.g., 60 in SPM). We thought that the number of available matrices was much too small for training our network, moreover we were concerned that using these items for training would compromise our ability to assess model performance on RPM. As Matzen et al. point out, one way to overcome these problems is to generate novel matrices, which is precisely what their software does. Second, the Matzen et al. norming study reports variation in item difficulty by item type, error patterns by item type, and other similar data which we found germane to the human modeling enterprise. Finally, we were further attracted to Sandia matrices as they have a well-defined structure suitable for a simple approach to training a convolutional neural network, a welcome feature given the exploratory nature of the present work.

2. Model

The visual system's limited capacity to process fine visual detail is an important constraint human embodiment places on RPM performance. RPM stimuli are large and intricate enough that subjects tend to fixate individual matrix or answer array figures as they work through a problem. Indeed, Carpenter et al. (1990) report that subjects tend to fixate back and forth between pairs of figures, suggesting that they are comparing these figures. Our present model is based on a simplifying assumption inspired by this observation. For the present purposes, we assume that all processing in RPM is based on pairwise comparisons of the kind suggested by Carpenter et al.'s observation. Under this assumption, subjects can inform their response for each matrix item by means of two actions: they can choose which pair of figures to inspect and they can compare the features that they observe, involving cognitive processes such as memory and executive control, in order to inform their response. In general, test takers should look for visual features and make comparisons that yield information about the correct answer. We can therefore define and evaluate a subject's *strategy* in terms of the observations and comparisons they are disposed to make (Kunda et al., 2013).

Matrices constrain viable strategies by virtue of their structure. We analyze the structure of a matrix in terms of its *visual lexicon* and its *layout*. The visual lexicon of a matrix is a finite set of visual features from which a matrix can be constructed. The term *lexicon* is meant to be suggestive: just as there are many ways to segment an acoustic signal, there are many ways to segment a raw matrix image (see Lovett & Forbus, 2017), but only some segmentations are allowed by the lexicon. The layout of a matrix is the spatial distribution of visual lexical items in the matrix. RPM matrix layouts consist of 1×1, 2×2, or 3×3 arrays of figures constructed out of basic elements in the visual lexicon; elements of the visual lexicon are generally distributed among matrix figures according to some pattern, we discuss several such patterns below. Next, answer arrays consist of several figures similar to those found in matrices. Every matrix has a blank area near or at its bottom right corner; this is the part that is considered missing. The blank must be filled with the answer array figure that

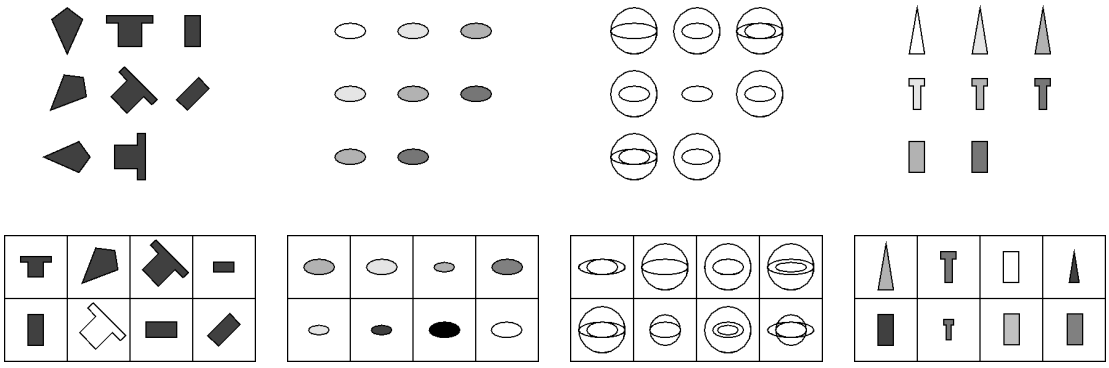


Figure 1. Sandia matrices and their answer arrays. The following types are depicted: Type II (a), Type III (b), Type IV (c) and composite Type II-III (d) .

fits the matrix best. According to the present analysis, this would be the answer array figure which agrees with the matrix visual lexicon and matrix layout the most. A sound strategy for an arbitrary matrix would therefore be one which disposes subjects to discover the matrix's visual lexicon, its layout, and, ultimately, the answer array figure which agrees with these the most. An important theoretical issue is whether a unique figure in the answer array best satisfies the constraints set by a matrix's visual lexicon and layout, as matrix problems must have exactly one correct answer in order to be well-formed (Wang & Su, 2015). Sandia matrices satisfy this constraint by virtue of their construction; however, it is unclear whether all original RPM problems satisfy this constraint.

The most influential analysis of RPM matrix structure is due to Carpenter et al. (1990); it consists of a collection of heuristics describing matrix layout patterns. Our work, however, closely follows Matzen et al.'s (2010) analysis of SPM, and this analysis does not map neatly onto that of Carpenter et al. (1990). For this reason, we present our own analysis of RPM matrix structure which closely reflects the Matzen et al. (2010) analysis and simplifies our presentation. Most analyses of Raven's matrices, including that of Carpenter et al. (1990), stipulate that matrix layouts preserve features along rows or columns. Matzen et al. (2010) make the notable observation that, in addition to these layout patterns, some matrix layouts also preserve features along diagonal axes.[1] This observation enables our model to cover a large variety of matrices despite a minimal design (see below). In some matrices, features (e.g., figure size) are progressively incremented as figures become increasingly distant from the top left, Matzen et al. call this layout pattern an *outward rule*. Finally, *logic rules* preserve logical relationships governing the presence of features between figures belonging to the same row (e.g., feature X is present in column 3 if it is present in column 1 and/or in column 2). Based on these observations by Matzen et al. (2010), we distinguish four basic strategies which solve four distinct matrix types. It is worth noting that some matrices require the

1. These matrices follow *distribution rules* in the Carpenter et al. (1990) terminology.

combined use of multiple basic strategies. See Figure 1 for samples of Sandia matrices solvable by three of the four basic strategies, as well as a matrix solvable by a combined strategy.

- Type I matrices have a 1×1 format and feature a texture-like figure, a part of which is missing. These matrices can be solved by comparing answer array items to the surroundings of the blank. The solution is the answer array item which matches its surroundings in the smoothest way. The Matzen et al. (2010) software does not generate Type I matrices (which was considered too simple).
- Type II matrices are matrices where each feature is constant along at least one axis (see below). These matrices can be solved by comparing matrix figures along linear scan paths. Matrices of this type are necessarily in the 2×2 or 3×3 formats. The solution is the answer which replicates all changes in figure attributes along each axis. Our model focuses on solving matrices of this type.
- Type III matrices are what Matzen et al. call outward matrices. These matrices can be solved by comparing matrix figures along the main diagonal. Matrices of this type are necessarily in the 3×3 format. The solution is the answer which differs from the center element in the same way as the center element differs from the top left.
- Type IV matrices have layouts which follow logic rules. They can be solved by comparing differences in pairs within each row to corresponding pairwise differences in other rows. These matrices are in the 3×3 format. The solution is the answer which differs from each pair in the third row in such a way that these differences are identical to those in the other rows.

2.1 Scope

Before describing our hybrid deep learning model of RPM in detail, some practical constraints on the scope of its present implementation should be mentioned. The foregoing discussion elaborates multiple strategies for tackling matrix problems, but, as our list suggests, each strategy solves a limited set of matrix problems. For example, the reader can verify that a strategy for solving Type II matrices, such as the one elaborated in the next subsection, may fail to solve Type III matrices such as the one depicted in Figure 1b. Thus, the elaboration and/or selection of a strategy is a key component of our model. However, for this paper we restricted our attention to implementing a solution strategy only for Type II matrices. Our deep-learning-based model of RPM can later be generalized to capture a wide range of human behavior on the task.

Among the different types of matrices, we chose to focus on Type II matrices. The ability to solve these matrices gives our model a scope comparable to other models in the literature. Type II matrices form a plurality of items in the SPM: over a third of all problems in the test, not counting Type II-III composites, are Type II matrices (see Matzen et al., 2010, for an analysis of individual SPM matrices).[2] All rules identified in Carpenter et al.'s (1990) seminal paper can yield Type II matrices. Furthermore, all Carpenter et al. rules, except *figure addition or subtraction rules* which can yield Type IV matrices and *quantitative pairwise progression rules* which can yield Type

2. We counted SPM matrices B1 and B2 as Type II matrices since they have a completely uniform layout, satisfying our definition of a Type II matrix.

III matrices, necessarily yield Type II matrices. Our model is designed to solve Type II matrices generated by the Sandia matrix generation tool (Matzen et al., 2010). These matrices are all of size 3×3, however our procedure generalizes to 2×2 matrices as well.

There is one final constraint on the scope of our present work which we briefly mentioned in the introduction. Original RPM problems each have their own visual lexicon, though similar elements tend to be repeated between matrices. This means that test takers must, for each matrix, construct a working visual lexicon and amend it until they are able to solve the matrix problem. This issue arises in relation to the problem of correspondence finding in Carpenter et al. (1990) and is addressed by perceptual reorganization processes in Lovett & Forbus (2017). Determining the visual lexicon associated with a matrix can be a difficult task, especially in more advanced RPM items. Indeed, identifying the visual lexicon is closely connected with the role of perceptual organization in RPM performance. Identifying the visual lexicon requires making decisions about figure segmentation and other visual inferences, and many of these inferences depend, in humans, on processes of perceptual organization. In the present paper, we do not address these. This restriction was made possible by the fact that Sandia matrices all share a well-defined visual lexicon which includes five matrix figure features: figure shape, shading, orientation, size, and number.

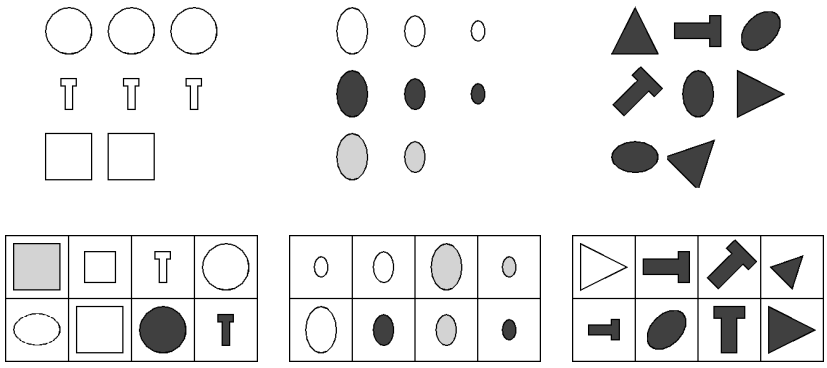
2.2 Solving Type II Matrices

Figure 2 shows a sampling of Type II matrices. The Matzen et al. (2010) software constructs Type II matrices by first selecting up to three features which will vary in the matrix layout. Each feature that varies is assigned an axis along which it will remain constant. These features are then assigned distinct values along each available path aligned with their assigned axis. For example, if the horizontal axis is selected as the axis along which a variable feature remains constant, as in Figure 2a, the feature is assigned the same value within each row in the matrix but different values between different rows. Note that this construction process determines the layout of the matrix; moreover, it completely specifies the answer to the matrix problem. Our solution strategy rests on the claim that one can solve Type II matrices by scanning and comparing figures along each axis. To get a sense of the intuition behind this strategy, consider again the problem depicted in Figure 2a. This problem can be solved by scanning the matrix along its horizontal and vertical axes. All cell figure features, other than shape, are constant throughout the matrix. All features, including shape, are constant along the horizontal axis. Shapes differ along the vertical axis. From these observations, one can conclude that the answer is the second cell from the left in the bottom row of the answer array. Our strategy for Type II matrices formalizes and systematizes this kind of reasoning for all Type II matrices.

Note, following Matzen et al.'s (2010) analysis, that subjects can scan matrices along four distinct axes: horizontal, vertical, and two diagonal axes. Scans along each axis have a linear form if matrix cell coordinates are interpreted to be in the integers modulo three ($\mathbb{Z}_3$). Interpreting matrix coordinates thus, we can define these linear scan paths parametrically as the set of points $\vec{x}$ such that

$$\vec{x} = \vec{c} \cdot t + \vec{b}$$

where t is a parameter in $\mathbb{Z}_3$, $\vec{c}$ is a coordinate vector specifying the axis with which the scan path is aligned, and $\vec{b}$ is the coordinate vector of a figure along the scan path. The vector $\vec{c}$ can be



(a) (b) (c)

Figure 2. Instances of easy (a), medium (b) and difficult (c) Type II matrices and their answer arrays.

obtained by taking the difference of the coordinate vectors of two figures on the chosen scan path. Consider, for example, the coordinate system for the matrix in Figure 2a where the top left figure is the origin, and, for each figure, the first coordinate indicates its row index from top to bottom, and the second coordinate indicates the column index from left to right. In this coordinate system, the center-right figure has coordinates $(1, 2)$ and the bottom left figure has coordinates $(2, 0)$. The vector $(2, 0) - (1, 2) = (1, 1)$ (coordinate vectors in $\mathbb{Z}_3 \times \mathbb{Z}_3$) defines the matrix axis with which this pair of figures is aligned. This axis is the top-left-to-bottom-right diagonal axis as can be seen from the fact that the vector $(1, 1)$ takes the origin figure to the center figure. Thus, one equation for the scan path which contains these example figures is given by

$$x = (1, 1)t + (2, 0)$$

For t ranging in 0, 1, 2, this equation yields the following coordinate vectors: $(2, 0)$, $(0, 1)$ and $(1, 2)$. The three figures located at these coordinates belong to the same scan path aligned with the top-left-to-bottom-right diagonal axis. Due to the symmetries of two dimensional vectors of coordinates in the integers modulo three, a total of 12 distinct linear scan paths can be constructed. These paths can further be spilt into groups of three, each group being aligned with one of 4 distinct matrix axes. These 4 axes correspond precisely to those identified by Matzen et al..

Two pairs of cells are aligned with the same matrix axis if the two vectors obtained by taking the difference of coordinate vectors within each pair are co-linear (in $\mathbb{Z}_3 \times \mathbb{Z}_3$). It follows from the construction of Type II matrices that for each image feature, there is always at least one axis along which it does not vary, and if a feature varies along some axis then each figure on a path along that axis exhibits a distinct value for that feature. Therefore, any pair of Type II matrix cells which are aligned with the same matrix axis will differ with respect to the same properties. We can summarize feature differences between a pair of matrix figures as a binary vector, with a value of 1 indicating a difference with respect to the corresponding property and a value of 0 indicating no difference.

We call such vectors *attribute difference vectors*. In this paper, we follow a convention whereby attribute difference vectors for Sandia matrices indicate differences in shape, shading, orientation, size and number in that order. For example, the pair consisting of the top left and middle figures in the matrix depicted in Figure 2a should be assigned an attribute difference vector of $(1, 0, 0, 0, 0)$ to indicate that figure shapes, but not other features, differ between the two cells.

It follows from the above that any Type II matrix will exhibit at most four distinct attribute difference vectors: one for each axis. It can be shown, by construction, that if attribute difference vectors are known for each axis, one has a complete specification of the answer figure. For each axis, one can pair a figure with the blank so that the pair is aligned with the chosen axis. We can construct a figure which agrees with the matrix figure in each such pair on the features which are constant along the chosen axis. Since every feature is constant along at least one axis, this construction determines a value for every feature in the visual lexicon. To see that such a figure is in complete agreement with the matrix layout, suppose that the blank were occupied by a figure constructed according to this procedure. By construction, attribute difference vectors from pairs including the constructed figure agree with the attribute difference vector of the corresponding axis for features that remain constant along that axis. This agreement holds also for features that change along the same axis. Consider such a changing feature. The constructed figure and its counterpart are, by assumption, not aligned with respect to any axis along which this feature is constant. Consequently, the value of the feature under question is inherited by the constructed figure from a scan path aligned with the axis along which this feature is constant that differs from the scan path determining the corresponding value for the constructed figure's counterpart. Since, for features that vary along some axis, figures on paths along that axis exhibit distinct values, the constructed figure must differ from the matrix figure with respect to the feature under question.

2.3 Architecture

The above discussion shows, by construction, that axial attribute difference vectors can be used to construct matrix answers; these vectors can equally be used to verify whether a given figure agrees with the matrix layout. Our strategy is thus to discover matrix axial attribute difference vectors and to then determine the answer array figure which best agrees with these vectors when inserted into the blank. This strategy is an example of a response elimination strategy (Vigneau et al., 2006). Our model simulates the processes of solving Type II matrices using this strategy. As explained at the start of the present section, it does so by executing a sequence of pairwise cell comparisons and using information from these comparisons to pick its answer. These comparisons can happen within the matrix, and between pairings of answer array and matrix cells. This process is captured by a rule set which collaborates with a deep convolutional neural network. The deep convolutional neural network represents the model's visual system. When the model chooses to analyze a pair of figures, these figures are passed to the network as inputs. The network then estimates an attribute difference vector for the input pair.

The model's deep convolutional neural network simulates the process of abstracting visual features from the raw image and that of detecting feature differences between pairs of images. See Table 1 for network architecture details. The abstraction process is captured by a stack of three convolutional layers and two pooling layers. The process of detecting feature differences is captured

Layer	Type	Filter Shape	Transfer Function
1	Convolution	$6 \times 2 \times 2$	$\tanh(x)$
2	Pool	3×3	$\max(x)$
3	Convolution	$10 \times 2 \times 2$	$\tanh(x)$
4	Pool	2×2	$\max(x)$
5	Convolution	$20 \times 2 \times 2$	$\tanh(x)$
6	Full	180×30	$2\,\mathrm{sech}(x) - 1$
7	Full	30×5	$(1 + e^{-x})^{-1}$

Table 1. Parameters for network layers. The Filter Shape column lists dimensions of the weight array for the corresponding layer if the layer is a convolutional (to be read $depth \times height \times width$) or fully connected layer. For pooling layers, the Filter Shape column designates the pool size ($height \times width$).

by two fully connected layers which receive input from the convolutional stack. During a forward pass, each image is presented to the convolutional stack separately, the resulting outputs are then collectively fed to the fully connected layers. The signal from the two images merge at the first fully connected layer (Layer 6), which is designed such that its output is independent from the ordering of input cells. The layer is rendered input order agnostic by the choice of an even transfer function and a constraint on connectivity such that the input y_6 to the layer has the form

$$y_6 = Wx_{51} - Wx_{52} + b_6 \tag{1}$$

where W is a weight matrix, x_{51} is the Layer 5 output from the first cell in the pair, x_{52} is the Layer 5 output from the second cell in the pair, and b_6 is the layer bias. This arrangement has the additional benefit of bestowing Layer 6 outputs with a theoretically germane interpretation. In particular, each output can be interpreted as indicating the degree to which the two images differ with respect to some specific feature by an amount set in the bias term; for this reason, we refer to Layer 6 as the *difference layer*. Inputs to the network are two scaled (down by 255) and mean-cancelled 28×28 grayscale (1 channel) rasters each depicting one cell in a cell pair. Training examples for the network consist of pairs of figures taken from Sandia matrices or their answer arrays as inputs and corresponding attribute difference vectors as outputs. Once the convolutional neural network is trained, the model is ready to be run.

The model chooses its response by assessing the degree to which each answer cell violates the matrix layout. To do this it computes a *layout violation score* for each answer figure. The layout violation score is computed in three steps. First, attribute difference vectors are averaged by axis, yielding summaries of attribute differences along each axis. Then, similar axis-based averages are computed for each answer. The layout violation score of an answer array figure is then calculated as the sum, over each axis, of the manhattan distance between the matrix and answer axial attribute difference vector summaries. These operations are executed using all observed pairs and they are controlled by the rules shown in Table 2. While, in this specific implementation, the model analyzes all relevant pairs, this is not a necessity. The algorithm can, in principle, be run with omitted observations. Indeed, selective omissions may be used to simulate fixation patterns characteristic of human subjects performing the task, we discuss this possibility in greater detail in Section 4.

Rule	Condition	Action
`Analyze Pair`	There is a goal g to analyze some pair, p, of matrix cells.	Analyze pair p, record results, clear goal g.
`Select Pair`	There is a pair, p, of matrix cells that has not been analyzed	Set a goal, g, to analyze pair p.
`Select Answer`	All pairs have been analyzed.	Compute matrix layout violation scores for every answer cell, select the answer cell with minimal score as response.

Table 2. Model rule set. Rules are presented in order of precedence.

3. Computational Experiment

We tested our model on a subset of the stimuli used in the Matzen et al. (2010) norming study for Sandia matrices. Matzen et al. generated a total of 840 matrices for their norming study. These matrices were grouped into 20 test sets containing a complete and balanced assortment of matrix types. Each of these 20 sets were presented to 4 subjects as part of the norming study. We sourced our matrices from 6 of these 20 sets. We used all Type II matrices present in these 6 sets, a total of 108, for training and testing the model. The model was implemented in Python 2.7; the `theano` (Theano Development Team, 2016) package was used for implementing the convolutional neural network.

The model's perceptual neural network was trained on pairs of cells from within the 108 matrices in our data set. During testing, the model was presented both with within-matrix cell pairs and with pairs of cells between the matrix and answer array, which it had not been trained on. Glorot initialization (Glorot & Bengio, 2010) was used to determine initial weights and biases. Training was performed using simple backpropagation augmented with early stopping (20% of training items were randomly selected and set aside for validation) and a variable learning rate. Each epoch, the learning rate was multiplicatively incremented if weight updates were found to improve training error. Otherwise, updates were rolled back and the learning rate was multiplicatively decremented. L2-regularized cross-entropy error was used as the cost function. See Table 3 for training algorithm parameters.

We trained the network for 3000 epochs, testing the model at 500 epoch intervals and on the weights selected by the early stopping algorithm. We found that the model performed maximally at 2000 and 2500 epochs. In both cases, it solved 78.7% of the 108 matrices correctly. Note that this result was obtained with only a minimum amount of tweaking; with more parameter optimization, even better results can be expected. In addition to investigating the overall performance of our model, we were also able to compare it to human data from Matzen et al. (2010). Average human accuracy on the 108 experimental matrices was 81.25%, close to our model's performance.

Parameter	Value	Interpretation
η	0.015	Learning rate
α	1.01	Learning rate increment
β	0.99	Learning rate decrement
λ	0.003	L2 regularization parameter

Table 3. List of training parameters, their values, and their interpretation.

4. Discussion

In this paper, we presented a model of RPM which integrates a deep convolutional neural network with rule-based reasoning. This model solves one class of RPM problems using abstract visual features extracted from raw images with no intervention from human annotators. We found that our model performed at a level comparable to that of human participants on a representative set of matrix problems taken from Matzen et al. (2010). We view this result as a promising first step towards the development of a model of human RPM performance which goes beyond the existing literature in its comprehensiveness.

Our model addresses a set of constraints on RPM task performance that other models in the literature do not completely address. As discussed in the introduction, the visual format of matrix problems has prompted two prominent approaches to RPM modeling, and one important difference between these two approaches has to do with how models represent RPM stimuli. In one approach, the *rule-based* approach, stimuli are represented by means of hierarchical symbolic representations whereas in the other, *similarity-based*, approach they are represented as pixel arrays.[3] Representational differences between models following these two approaches are accompanied by differences in processing. Models following the rule-based approach, among which we include heuristic rule induction (Carpenter et al., 1990; Ragni & Neubert, 2014; Rasmussen & Eliasmith, 2014) and analogical (Lovett & Forbus, 2017) models, identify patterns governing the distribution of abstract visual features within matrices and select answers which best comply with these patterns. On the other hand, models following the similarity-based approach, among which we include affine (Kunda et al., 2013) and fractal (McGreggor et al., 2014) models, identify image transformations which relate matrix figures and select answers which best agree with the patterns of observed transformations.

The performance of rule-based models depends on the number and kinds of abstract visual features they can represent as well as the complexity of available feature distribution rules. On the other hand, similarity-based model performance depends on the metric properties of RPM stimuli (two dimensional images in euclidian space) and the complexity of available image transformations. Unlike existing models of RPM, our model's performance is simultaneously and explicitly subject to both sets of constraints. Limitations of knowledge and/or assumptions represented in the rule

3. The definition of these two approaches was inspired by the difference between the Analytic and Gestalt RPM algorithms introduced by Hunt (1974). A rule-based model is an instance of the analytic algorithm, whereas a similarity-based model is an instance of the Gestalt algorithm.

set, including knowledge about which image features are subject to variation (visual lexicon) and assumptions about matrix layout, place constraints on our model's performance of the same kind as that to which rule-based models are subject. To see how constraints to which similarity-based models are subject affect our model, we need to take a closer look at the model's perceptual apparatus. Our model's perceptual apparatus estimates attribute difference vectors by summarizing activations at the difference layer. As can be inferred from Equation 1, input to the difference layer is a linear function of the vector difference between representations obtained from the convolutional stack of the figures in a pair of interest. Therefore, given the final convolutional representation of one figure in a pair and the input vector for the difference layer, the convolutional representation of the second figure in the pair can always be obtained. In other words, our model identifies, as an intermediate step in estimating the attribute difference vector for a pair of matrix figures, a transformation relating each figure in the pair. The complexity of these transformations depends on the complexity of the convolutional stack and on the size of the difference layer. Metric properties of RPM stimuli can also affect our model's performance. For instance, if relevant matrix features are too small or the model's receptive fields are too large, the model may be unable to resolve these features and use them to inform its response.

The human visual system has a limited ability to resolve spatial patterns and therefore to discriminate visual features (as evidenced by e.g., contrast sensitivity functions; see Palmer, 1999, p. 163–165). At the same time, humans can process limited numbers of visual features at a time due to the limits of visual short-term memory (see Marois & Ivanoff, 2005, for discussion) and they identify some clusters of visual features more readily than others due to the effects of perceptual organization (see Palmer, 1999, Chapter 6 for a review). The former constraints are best captured by similarity-based models whereas the latter constraints are best captured by rule-based models. For this reason, our model, which integrates features of both approaches, presents a platform which can model the visual aspects of human RPM performance in a more comprehensive fashion than is afforded by either approach on its own. That said, our model also has a number of limitations which should not be left unaddressed. These limitations concern both model architecture and model task knowledge, but we believe they can be addressed by further development.

The primary limitation of our model's task knowledge is its specificity. All current model knowledge is specific to Type II Sandia matrices. In particular, the model's knowledge of the visual world is limited to Sandia matrices, and its knowledge of matrix layouts is sufficient to solve arbitrary Type II matrices, provided attribute difference vectors are correctly estimated (see Section 2.2). An important prerequisite for the validity of test results is that subjects are unfamiliar with the test (Raven et al., 1998). Thus, under normal circumstances human subjects bring only generic knowledge about the visual world, in addition to general reasoning and observation skills, to bear on the task. In other words, our model has too little general knowledge about the visual world and too much knowledge about a specific kind of matrix problem in comparison to ordinary human subjects taking the RPM tests.

A more psychologically realistic version of our model will need to have more general visual knowledge and have much less a priori knowledge about the structure of matrix problems. One way these requirements can be met is through use of different training data and network parameters, and through modification of rules used. For instance, the model may have to produce more complex

attribute difference vectors (e.g., by a higher number of dimensions, as determined by the number of nodes available in the output layer of the perceptual neural network) in order to enable its rule system to explicitly process a greater variety of visual knowledge. Another direction for improving the psychological realism of our model has to do with learning. Currently, learning processes equip our model with knowledge of the visual world by tuning its perceptual apparatus to relevant stimuli. Human subjects acquire this kind of knowledge over the course of their development; that is, on time scales ranging from months to decades. However, human subjects also learn within the duration of a single RPM test. RPM items get progressively more difficult and build on concepts from previous items, thus subjects can learn to solve more difficult matrices based on their experience with easier ones (Raven et al., 1998). It may be possible to capture this kind of online learning by developing a general procedure for inducing matrix layout patterns. Such a procedure would govern visual exploration of matrix layouts and processes for inferring regularities based on such exploration.

Other modifications to the rule set may allow the model to better reflect individual differences in task performance by more closely approximating human eye movement patterns. In this paper, we defined a subject's strategy in terms of the pairwise comparisons they are disposed to make and in terms of how they make use of the resultant information to inform their response. This approach is consistent with observations from modelers (e.g., Carpenter et al., 1990) and experimentalists (e.g., Vigneau et al., 2006) that high and low performers on RPM tests exhibit different eye movement patterns which are thought to reveal differences in strategy. While this is another interesting direction for improving our model, some architectural refinements are required for meaningful capture of human strategies. An important discrepancy between the architecture of our model and that of the human visual system is that our model can simultaneously compare two images whereas humans must sequentially fixate two different locations in order to carry out similar comparisons. The discrepancy can be remedied by a visual memory store that replaces the second input to the model's perceptual array. But this remedy only serves to highlight a second important architectural limitation. In solving only a single matrix, our model currently makes 98 pairwise comparisons. Individually storing an attribute difference vector for each of these comparisons in memory is at odds with known limitations of human working memory. Working memory is an important contributor to RPM performance. Differences in working memory capacity are thought to be responsible for individual differences in RPM performance and the working memory load induced by a matrix problem is thought to be an important contributor to item difficulty (Vigneau et al., 2006). Thus another direction for refining the architecture of our model is to more closely model the limitations of human working memory. The final architectural refinement we consider here has to do with the addition of perceptual organization processes into our model. As discussed earlier in this paper, there is evidence that difficult matrix problems require explicit control of perceptual organization. A prominent way to model perceptual organization in neural networks is to make use of lateral inhibition (Grossberg et al., 1997). Our model could be extended to simulate explicit control of perceptual organization processes by the addition of lateral connections to its perceptual array and through a channel by which the rule set can exert top-down influence on the perceptual array.

5. Conclusion

In contrast to previous work on modeling RPM, our approach has focused on learning and integrating perceptual (similarity-based) processes and rule-based processes. This was done through the integration of a deep convolutional neural network with a simple rule system. This work represents an approach to modeling Raven's Progressive Matrices (RPM) which extends the existing literature in three respects: it is the first learning-based model of RPM, it is the first model of RPM that makes use of deep learning, and it is also the first model of RPM to combine sub-symbolic and symbolic processing. Our model learns and carries out processes of perceptual abstraction that underly the human ability to use inferential or analogical processes in order to solve RPM problems. We have argued that our approach is conducive to capturing a set of cognitive constraints that existing approaches cannot completely capture and we have suggested several ways in which our model can be refined towards this goal. We aim to refine our model in these directions in the hopes of improving our understanding of the processes that underly human intelligence.

Acknowledgements

We acknowledge support from the Army Research Institute through a research grant. We are grateful to Dr. Laura E. Matzen for providing us with the Sandia matrix-generation tool, matrices used in the Matzen et al. (2010) norming study and human data from this norming study.

References

Carpenter, P. A., Just, M. A., & Shell, P. (1990). What one intelligence test measures: A theoretical account of the processing in the Raven Progressive Matrices test. *Psychological Review*, *97*, 404–431.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS) 2010* (pp. 249–256). PMLR.

Grossberg, S., Mingolla, E., & Ross, W. D. (1997). Visual brain and visual perception: How does the cortex do perceptual grouping? *Trends in Neurosciences*, *20*, 106–111.

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

Kriegeskorte, N. (2015). Deep neural networks: A new framework for modeling biological vision and brain information processing. *Annual Review of Vision Science*, *1*, 417–446.

Kunda, M., McGreggor, K., & Goel, A. (2010). Taking a look (literally!) at the Raven's Intelligence Test: Two visual solution strategies. *Proceedings of the 32nd Annual Conference of the Cognitive Science Society*. Austin TX: Cognitive Science Society.

Kunda, M., McGreggor, K., & Goel, A. K. (2013). A computational model for solving problems from the Raven's Progressive Matrices intelligence test using iconic visual representations. *Cog-*

nitive Systems Research, *22–23*, 47–66.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*, 436–444.

Lovett, A., & Forbus, K. (2017). Modeling visual problem solving as analogical reasoning. *Psychological Review*, *124*, 60–90.

Marois, R., & Ivanoff, J. (2005). Capacity limits of information processing in the brain. *TRENDS in Cognitive Sciences*, *9*, 296–305.

Matzen, L. E., Benz, Z. O., Dixon, K. R., Posey, J., Kroger, J. K., & Speed, A. E. (2010). Recreating Raven's: Software for systematically generating large numbers of raven-like matrix problems with normed properties. *Behavior Research Methods*, *42*, 525–541.

McGreggor, K., Kunda, M., & Goel, A. (2014). Fractals and ravens. *Artificial Intelligence*, *215*, 1–23.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. ArXiv Preprint.

Palmer, S. E. (1999). *Vision science: Photons to phenomenology*. Cambridge, MA: The MIT Press.

Primi, R. (2001). Complexity of geometric inductive reasoning tasks: Contribution to the understanding of fluid intelligence. *Intelligence*, *30*, 41–70.

Ragni, M., & Neubert, S. (2014). Analyzing Raven's intelligence test: Cognitive model, demand and complexity. In H. Prade & R. Gilles (Eds.), *Computational approaches to analogical reasoning: Current trends*, volume 548 of *Studies in Computational Intelligence*. Berlin, Heidelberg: Springer-Verlag.

Rasmussen, D., & Eliasmith, C. (2014). A spiking neural model applied to the study of human performance and cognitive decline on Raven's Advanced Progressive Matrices. *Intelligence*, *42*, 53–82.

Raven, J., Raven, J. C., & Court, J. H. (1998). *Manual for Raven's Progressive Matrices and Vocabulary Scales. Section 3, the Standard Progressive Matrices*. San Antonio, TX: The Psychological Corporation.

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*. From `http://arxiv.org/abs/1605.02688`.

Vigneau, F., Caissie, A. F., & Bors, D. A. (2006). Eye-movement analysis demonstrates strategic infulence on intelligence. *Intelligence*, *34*, 261–272.

Wang, K., & Su, Z. (2015). Automatic generation of Raven's Progressive Matrices. *Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI2015)* (pp. 903–909).

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *EECV 2014*. Springer Internationl Publishing Switzerland.