

RPM Project: Final Report

Karan Bhasin

karanbhasin@gatech.edu

1 RAVEN'S PROGRESSIVE MATRICES AI AGENT

A knowledge-based AI agent was developed to pass the Raven's Progressive Matrices (RPM) human intelligence test. The agent is configured to solve 2x2 matrix problems from Set B, and 3x3 matrix problems from Sets C, D, and E. Each of these four sets contains four types of problems: Basic, Test, Raven's, and Challenge. The agent solves the problems using nine main components: Preprocessing, Equality, Reflections, Rotations, Area Change, Shape Fill, Shape Quantity functions, Logic Operation functions, and Dark Pixel Pattern functions. The agent's solving process is illustrated in Figure 1. The agent processes the images before passing them through a list of 2x2 or 3x3 functions in the order they are listed till a suitable answer is returned. For example, if a 3x3 RPM problem has a transformation that involves the area of a shape changing, the agent would first pass the preprocessed input through Equality, then Reflections, before finding a suitable option through the Area Change function.

The main components of the agent are explained in the following sections. The functions' descriptions include a reference to the image pairs or groups in the grid analyzed to check for a transformation or pattern. If a transformation or pattern is detected in the image pair or group, it is applied to an image or pair of images adjacent to the missing image along the same axis to find a suitable answer from the provided options. The image pairs or groups and their respective adjacent images for 2x2 and 3x3 problems is illustrated in Figure 2.

1.1 Preprocessing

The agent first reads each of the given grid images and option images in grayscale, after which it applies inverse-binary thresholding to them. The color inverse ensures the agent can detect contours more accurately. The grid and option images are stored in separate dictionaries. In the dictionaries, the key is the image name, and the value is the image as a numpy array. The agent detects all the unique contours, contour areas, and number of contours in each of the given images. It also calculates the ratio of the number of black pixels to total number of

pixels in each grid image. If the number of grid images is 3, the agent passes the dictionaries through the 2x2 functions, if the number of given images is 8 then it passes the dictionaries through the 3x3 functions. The first function to be called in both cases is the Equality function.

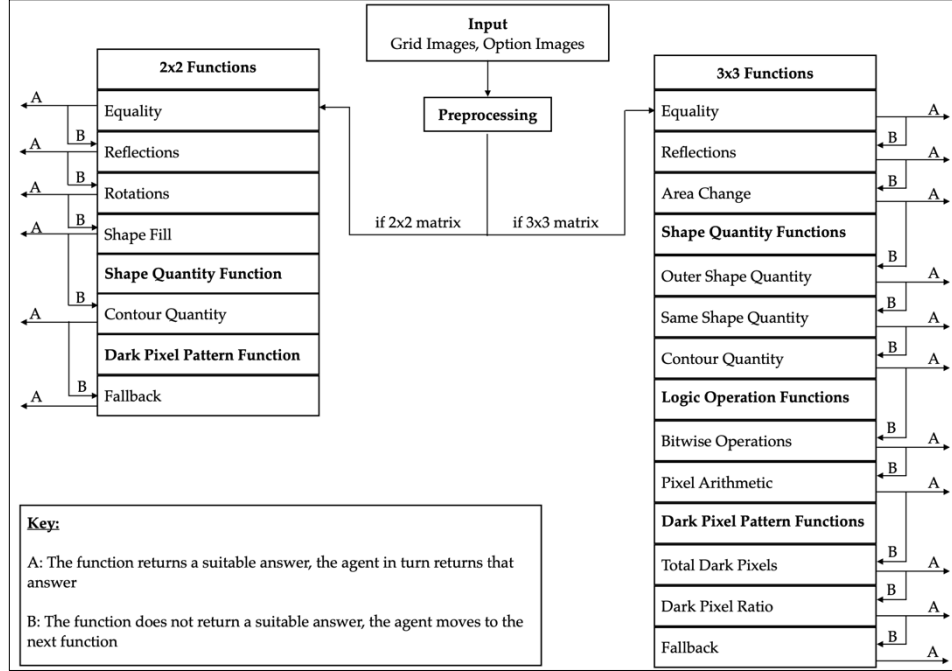


Figure 1 – The RPM AI agent's solving process

1.2 Equality

The agent checks for equality in the given images. For 2x2 problems, the agent checks for similarity in AB, and AC. If at least one of the image pairs shows similarity, the agent iterates through the option images to find an option similar to the adjacent image. For 3x3 problems, if AC or DF, BC or EF, AG or BH, DG or EH or AE show similarity and the number of contours, and number of black pixels in the images in the pairs mentioned are similar, then the agent iterates through the options to find an option similar to their respective adjacent image.

1.3 Reflections

The agent checks whether one image is the flipped version of another. For 2x2 problems, A is flipped around the y-axis and compared to B, and flipped around the x-axis and compared to C. If at least one of those cases has a similarity then the flipped version of the adjacent image is generated as a possible solution and tested by comparing with each of the options till a similarity is found. For 3x3

problems, the agent performs the same procedure with the image pairs AC and AG.

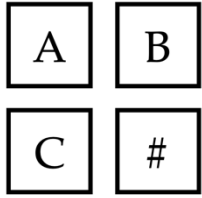
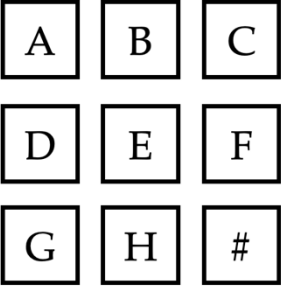
<p>A 2x2</p>  <table border="1" data-bbox="332 611 630 709"> <thead> <tr> <th>Image Pair</th><th>Axis</th><th>Adjacent Image</th></tr> </thead> <tbody> <tr> <td>AB</td><td>Horizontal</td><td>C</td></tr> <tr> <td>AC</td><td>Vertical</td><td>B</td></tr> </tbody> </table>	Image Pair	Axis	Adjacent Image	AB	Horizontal	C	AC	Vertical	B	<p>B 3x3</p> 	<table border="1"> <thead> <tr> <th>Image Pair/Group</th><th>Axis</th><th>Adjacent Image/Pairs</th></tr> </thead> <tbody> <tr><td>AC</td><td>Horizontal</td><td>G</td></tr> <tr><td>DF</td><td>Horizontal</td><td>G</td></tr> <tr><td>BC</td><td>Horizontal</td><td>H</td></tr> <tr><td>EF</td><td>Horizontal</td><td>H</td></tr> <tr><td>GH</td><td>Horizontal</td><td>H</td></tr> <tr><td>AG</td><td>Vertical</td><td>C</td></tr> <tr><td>BH</td><td>Vertical</td><td>C</td></tr> <tr><td>DG</td><td>Vertical</td><td>F</td></tr> <tr><td>EH</td><td>Vertical</td><td>F</td></tr> <tr><td>CF</td><td>Vertical</td><td>F</td></tr> <tr><td>AE</td><td>Diagonal</td><td>E</td></tr> <tr><td>DH</td><td>Diagonal</td><td>E</td></tr> <tr><td>ABC</td><td>Horizontal</td><td>GH</td></tr> <tr><td>DEF</td><td>Horizontal</td><td>GH</td></tr> <tr><td>ADG</td><td>Vertical</td><td>CF</td></tr> <tr><td>BEH</td><td>Vertical</td><td>CF</td></tr> </tbody> </table>	Image Pair/Group	Axis	Adjacent Image/Pairs	AC	Horizontal	G	DF	Horizontal	G	BC	Horizontal	H	EF	Horizontal	H	GH	Horizontal	H	AG	Vertical	C	BH	Vertical	C	DG	Vertical	F	EH	Vertical	F	CF	Vertical	F	AE	Diagonal	E	DH	Diagonal	E	ABC	Horizontal	GH	DEF	Horizontal	GH	ADG	Vertical	CF	BEH	Vertical	CF
Image Pair	Axis	Adjacent Image																																																												
AB	Horizontal	C																																																												
AC	Vertical	B																																																												
Image Pair/Group	Axis	Adjacent Image/Pairs																																																												
AC	Horizontal	G																																																												
DF	Horizontal	G																																																												
BC	Horizontal	H																																																												
EF	Horizontal	H																																																												
GH	Horizontal	H																																																												
AG	Vertical	C																																																												
BH	Vertical	C																																																												
DG	Vertical	F																																																												
EH	Vertical	F																																																												
CF	Vertical	F																																																												
AE	Diagonal	E																																																												
DH	Diagonal	E																																																												
ABC	Horizontal	GH																																																												
DEF	Horizontal	GH																																																												
ADG	Vertical	CF																																																												
BEH	Vertical	CF																																																												

Figure 2— Image pairs/groups and their respective adjacent image/pairs for 2x2 and 3x3 problems

1.4 Rotations

The agent checks for a transformation involving rotation. A is rotated 90-degree clockwise and compared to B and C. If at least one of the two pairs is similar, then the adjacent image is rotated 90-degree clockwise to generate a possible solution and an option that is similar is returned by the agent. If the agent doesn't find an acceptable option, then transformations involving a 180-degree rotation and a 270-degree rotation are similarly checked.

1.5 Shape Fill

The agent checks if any of the same shapes in A is filled by a block of color in B or C. It identifies each contour and its area in the given images. If the number of shapes in AB or AC is equal, then each of A's shapes' area is divided by the number of black pixels in B and C respectively. If any of the ratios is similar, it can be deduced that that shape's area and number of black pixels are similar, and it has been filled in the transformation. Then, each of the adjacent image's contours' area is divided by the number of black pixels in each option image till a similar ratio is found. The option image that results in a similar ratio as the one calculated earlier is returned as an acceptable solution. Vice versa, the agent also

checks if a shape that is filled in A is “unfilled” in B or C and uses a similar approach in finding a possible optimum solution.

1.6 Area Change

The agent checks if there is a change in area of a shape from one image to the next. The agent checks for this transformation in EF, EG, and AE. It first checks if the number of contours/shapes are the same in both images and ignores any shapes that are similar in both figures. For the differing contours, it calculates the magnitude by which the area changes by dividing the area of the differing shape in one figure by the area of the differing shape in the adjacent figure. The agent then performs the same operation on the adjacent image, and each of the options. If the adjacent image and one of the options have a shape with a similar magnitude of change in area, then that option is returned as a favorable option.

1.7 Shape Quantity functions

1.7.1 *Outer Shape Quantity*

The agent checks for change in number of a shape surrounding a primary shape in the middle. The agent analyzes the image pairs AE, BC, and DG. If there is a change in number of contours between the images in a pair and the image with lesser contours has all its black pixels intersecting with the black pixels of the other image that means the number of surrounding shapes has increased from one image to the next. The agent returns the option that shows a similar relationship with the respective adjacent images.

1.7.2 *Same Shape Quantity*

The agent checks if the quantity of the same shape is changing by a magnitude or a uniform number from one image to the next. To check for a quantity changing by a magnitude, the agent analyzes the image pairs AC and AG. The agent finds the ratio of the number of shapes in one image to the number of shapes in the other. It then compares that with the ratio of the number of black pixels in one image to the number of black pixels in the other image. If the ratios are similar then the same ratios are calculated for the respective adjacent image and each option. If the adjacent image and an option have a ratio similar to the one of the image pair, then it means there is an increase in the quantity of shape with a similar magnitude and that option is returned as the answer.

For a uniform increase in the number of shapes, the agent compares the difference in the number of black pixels between image pairs AC and DF, and AG and BH. If the two differences are similar for either pair of image pairs, it calculates the difference in number of black pixels between the respective adjacent image and each option. If the difference between the adjacent image and an option is similar to that between the image pair, it returns that option as a suitable answer.

1.7.3 Contour Quantity

For 2x2 problems, the agent checks if any shapes have been added to or deleted in the transformation in AB or AC. For each image in the pair, the agent calculates the number of shapes and each shape's area. If AB, or AC consist of similar shapes, based on their areas, but one of them has a higher number of shapes, then the areas of the additional shapes is calculated. Then iteratively, each option's shapes and areas are calculated, and if the adjacent image and the option have similar shapes and the additional shape in one of the figures is the same in area as the one previously calculated, then that option is returned as the solution. For 3x3 problems, the agent analyzes the change in number of black pixels and contours between the image groups, ABC, ADG, and image pairs, AE and BF. If the number of contours stay the same across the row, column, or diagonal but the number of black pixels changes, then that indicates the same number of different shapes on that axis. The option that has the same number of contours but different number of black pixels compared to each image in the adjacent image pair is returned as a solution.

1.8 Logic Operation functions

1.8.1 Bitwise Operations

The agent checks for different logic transformations that might be present in the grid. The agent first applies binary thresholding instead of inverse-binary thresholding to the given and option images for accurate processing using the OpenCV bitwise logic operators. The agent checks for AND, OR, and XOR operations in the rows and columns of the grid. The agent checks for the presence of one of these three operations on the image groups ABC, DEF, ADG, and BEH. If one of the logic operations is found in the image groups, then the same operation is performed on the adjacent image pair and the option matching the resulting image is returned by the agent.

1.8.2 Pixel Arithmetic

The agent checks if the sum or difference of the number of black pixels in two images in an image group is equal to the number of black pixels of the third image. The agent checks for this trend in ABC, DEF, ADG, and BEH. If the trend is observed in one of the image groups, the agent returns the option that has the number of black pixels equal to the sum or difference, depending on the trend, of the adjacent pair.

1.9 Dark Pixel Pattern functions

1.9.1 Total Dark Pixels

The agent checks if the total number of black pixels in the three images in each of the two rows, ABC and DEF or the two columns, ADG and BEH are equal. If so, then the agent adds the number of black pixels in the adjacent pair to each option till the sum of black pixels is similar to that of the respective image group. If multiple options fulfill this condition, then the option not present in the grid images is returned as the answer.

1.9.2 Dark Pixel Ratio

The agent calculates the difference in black pixel ratios of AC, AG, AE, CF, and GH. The black pixel ratio is the total number of black pixels in a figure divided by the total number of pixels in a figure. The agent subtracts the difference in the black pixel ratios, or adds the difference if an increase in black pixel is detected, from the black pixel ratio of the adjacent image to calculate the black pixel ratio of a possible solution. Whichever option has a similar black pixel ratio is returned as an answer.

1.9.3 Fallback

If still without an answer, the agent calculates difference in black pixel ratios of AB and AC for 2x2 problems, and AC, AG, AE, CF, and GH for 3x3 problems. It then subtracts/adds this difference with the respective adjacent image's black pixel ratio. This intended black pixel ratio is compared with the black pixel ratio of each option. The option with the least difference is returned as a fallback answer.

2 AGENT APPROACH

The agent uses different variations of generate and test across the functions. In the Equality, Reflections, Rotations and Pixel Logic methods, the agent first generates an answer and if a match is found with the given options, then that option is returned. For Area Change, Shape Quantity functions and Pixel Pattern functions, if the agent detects a transformation in the grid, then the agent tests that transformation with the transformation between the adjacent image and each of the options on the same axis. If there is a similarity in the transformations, that option is returned. In the Fallback function, the agent uses nearest neighbor method by returning the option that has the dark pixel ratio nearest to the ratio expected in the answer.

Most of the functions are structured as production systems. The agent first checks if certain conditions indicating a specific transformation or pattern along a certain axis are met. If so, then the agent checks if one of the options meets the conditions necessary to be returned as an answer. If the conditions for a transformation/pattern are not met, or none of the options qualify as an answer, then the agent moves on to the next set of possible conditions along the other axes. If all the possible conditions of a function are exhausted, the agent then moves on to the next function. It repeats this process till it finds a satisfactory solution.

3 AGENT PERFORMANCE

The agent's performance is evaluated based on its ability to solve the Basic, Test, Challenge, and Raven's type problems of Sets B, C, D, and E. Since the objective of the project is for the agent to mimic human cognition and pass a human intelligence, it can further be assessed by converting its performance on the RPM test to a percentile and IQ in human terms. Table 1 summarizes the agent's performance across the four different sets and four types of problems. The agent performs satisfactorily on the Basic and Test problems. It can solve all 48 of the Basic problems and 36 out of 48 of the Test problems for a combined 84 out of 96 problems, giving it an 87.5% success rate on these sets. The agent's high performance on these sets can be attributed to the fact that Basic problems were solely used to develop the agent and the Test problems are directly analogous to the Basic problems. Though the performance is high, the agent might be vulnerable to overfit and could have a bias towards the transformations and trends found in these

types of problems. This is evident as the agent’s performance drops significantly when facing problems from the Raven’s and Challenge sets, solving 60.42% of the Raven’s problems and only 35.42% of the Challenge problems.

Table 1 – The RPM AI agent’s performance on the different types and sets of problems

Sets	Basic	Test	Raven's	Challenge	Basic and Test	Basic, Test, Raven's, and Challenge
Set B	12/12	11/12	7/12	4/12	23/24	34/48
Set C	12/12	8/12	9/12	8/12	20/24	37/48
Set D	12/12	10/12	4/12	1/12	22/24	27/48
Set E	12/12	7/12	9/12	4/12	19/24	32/48
Total	48/48	36/48	29/48	17/48	84/96	130/192
Success Rate	100%	75%	60.42%	35.42%	87.50%	67.71%

The agent’s performance can be further analyzed by converting its results to a percentile rank and IQ score in human terms. Since the raw scores used for the conversions are out of 60 and not 48, the success rate is applied to derive a raw score for conversion. The tables used for the conversion of raw scores into percentile ranks and IQ score were developed by Peck (1970). The agent can be compared to a 30-year-old human, the same age as me, its creator. The Basic and Test problems are inspired by the real RPM test, so we can use its results for an assessment of the agent. The agent’s 87.50% success rate gives a raw score of 52/60, giving the agent an 85-percentile rank for a 30-year-old and an IQ of 115. Strictly for only the Raven’s problems, the 60.42% success rate translates to a raw score of 36/60 giving the agent a 32-percentile rank for a 30-year-old and an IQ of 93.

3.1 Agent struggles

The two major vulnerabilities the agent has that causes it to struggle with certain problems is overfitting and “trapping”. The agent was developed only using the transformations and trends in the Basic problems; hence, it is vulnerable to overfitting to that set. Since the Test set is analogous to the Basic set, the agent performs well on that set too. The agent however vastly underperforms in the Raven’s and Challenge type problems as can be seen in Table 1. This is even more apparent in Set D where the agent gets 22/24 Basic and Test correct but is only able to get 1 problem in the Challenge set correctly. An example of this can be

seen in Figure 3A, the agent has no function to account for such a transformation and on running our agent for this problem, the Total Dark Pixels function seems to return a suitable albeit incorrect answer.

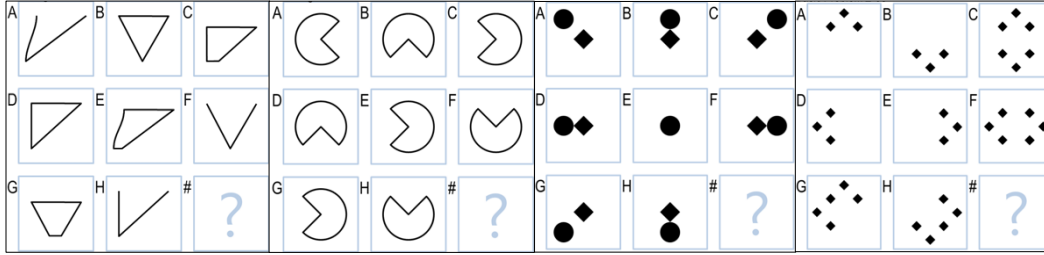


Figure 3 – Left to right - 3A: Example of overfitting. 3B: Example of trapping. 3c: Example of Reflection. 3D: Example of Logic

“Trapping” happens when the agent unwillingly and unexpectedly returns an answer through a function other than the desired function that can provide a more suitable answer. This can happen either because a function precedes the desired function and the incorrect option satisfies the conditions set, or the conditions set for the desired function don’t account for a certain problem and a latter function returns an incorrect option. Figure 3B shows a problem where the Reflect function should have returned the correct answer, but it failed to and later Total Dark Pixel’s conditions were triggered, and an incorrect answer was returned. This can be avoided by employing a scoring system where each function returning a suitable option is recorded and the option that is the closest matching or satisfies an additional condition is returned as the answer.

3.2 Agent successes

The agent is successful in solving problems with lower complexity transformations such as Equality, Reflection and Rotations. These functions alone solve 14/48 of the Basic problems. The agent examines all pertinent relationships in the grid, and through the development of the agent these functions have been generalized and have effectively solved problems in each set. Figure 3C shows an example of a problem where the agent can quickly return an answer using Reflection without analyzing other transformations in the grid. These functions are vulnerable to trapping as discussed above but are still quite simple and effective. The agent also performs well on logic transformations. This is again due to the simplicity of the Logic Operations functions and the ability to easily generalize them. Since the agent is looking for a specific pattern across the rows and

columns, it is easy to identify whether a problem is a logic problem. If it is a logic operation problem, then the agent can easily identify the correct option by performing the same operation on the adjacent pair. Figure 3D shows an example of such a problem. These functions were able to solve 11/12 Basic Set E problems.

Finally, the agent's Dark Pixel Pattern functions are successful in generalizing a large array and types of problems by analyzing the sum or ratio of their black pixels. They successfully solve 7/48 of Basic problems across all sets. They are especially helpful in avoiding overfitting. If all the preceding, more biased functions fail, the Total Dark Pixels, Dark Pixel Ratio, and Fallback functions are successful in solving many problems successfully.

4 AGENT - HUMAN COMPARISON

Before developing the RPM AI agent, I wrote down every unique approach I used, or any common pattern I observed as I solved each of the Basic problems. These approaches and patterns became the basis of my agent's functions. The agent, in essence, thinks like I do when approaching a problem. It employs each of my strategies on the problem till it finds a solution. It looks for patterns, and familiar transformations and if confirmed, checks if any of the options satisfy the pattern. If not, it looks for other patterns. This is reflective of how a human would approach these problems. Since, the agent and a human are working with limited information, they both use generate and test in search for an optimum answer. If a simple transformation is observed, a potential answer is generated and tested against the options. For more complex transformations, the agent, like a human, compares the relationship between the adjacent image and each option with the patterns on the rest of the grid to check if an option might be the answer.

There are however some differences between the agent and a human. Since the agent was developed using a limited scope of problems, it is biased towards certain transformations and patterns, and hence prone to overfitting. Humans do not have such a problem as they have a more robust cognition to identify new patterns. The agent can however perform better than a human in other aspects. It can use the minutiae of data to be a more effective problem solver, such as by using pixel ratios. Once properly configured, it is also much faster and more efficient in solving the problems. Overall, the RPM AI agent is a small, yet capable part of my own cognition in the form of code.

5 REFERENCES

1. Peck, D. F. (1970). The conversion of Progressive Matrices and Mill Hill Vocabulary raw scores into deviation IQ's. *Journal of Clinical Psychology*, 26(1), 67-70. <https://onlinelibrary.wiley.com/doi/abs/10.1002/1097-4679%28197001%2926%3A1%3C67%3A%3AAID-JCLP2270260117%3E3.o.CO%3B2-T>