Solving Raven's Matrices: Artificial Intelligence on Human Intelligence Scale

Zixuan Zhao

Master of Science
School of Informatics
University of Edinburgh
2019

Abstract

Intelligence is a board concept that describes the ability of agents being able to process its context and generate solutions to problems. The concept of artificial intelligence had emergence for years, however, it is still hard to tell how intelligent they are. In this project, we have trained convolutional neural networks (CNN), convolutional neural networks with feed-forward neural networks (CNN+FNN), recurrent neural network (RNN), convolutional neural networks with recurrent neural networks (CNN+RNN) to solve increasingly complicated Raven's Matrices problems. The results are promising and distributed. CNN with all contrast training method achieves the best accuracy in first six integrated problems (6 out of 6 are completely solved), CNN+FNN yield the best, stable result in two by two Raven's Matrices problems (7 out of 12), RNN provide best answers in 6-12 integrated problems (2 out of 6) and CNN+RNN gives the best performance in three by three cases (8 out of 36). The algorithms are generally incapable of solving the three by three Raven's Matrices. They are inefficient in representing the relationships between part and integrity. When the associations are embedded in strings prepossessed by the experimenter, 22 out of 36 cases can be efficiently solved. For all algorithms, if effective distinctions between the results of algorithms are possible, maximally 36 problems are solved by the algorithms, 41 including the cases solved by more efficient associative representations. The unsolved cases primarily contain cases which require the algorithms to identify the missing permutation of given patterns. These results are validated by experiments with test number matrices designed for algorithms. The attempts prove that deep neural networks can be efficient in abstracting non-facial overall relationships in novel situations with limited previous experiences or resources if they cooperate optimally. Building a hybrid high-level algorithm might be a possible further development for this project in the future.

Acknowledgements

I would like to take this opportunity to acknowledge a personal thank to Dr. Michael Herrmann for his brilliant advice and kind acceptance. It was all his broad scope of knowledge and familiarity with experimental frameworks that makes my originally unorganized ideas a doable project. Also thanks to Calum Imrie and all the lab members for the colorful meetings and board conversations that give me a lot of ideas in completing the thesis.

Table of Contents

Chapter 1

Introduction

Intelligence is a direct consequence of high-level cognitive abilities that make us survive. It is generated by highly conscious agents who have their need to perceive and take control of the environment they live in [18]. During evolution that started from single-cell creatures to more evolved ones such as birds, primates, or human, the beneficial reflections are selected and retained across the generations, shaping the brain and developing a layer of cortex where the instinctual behaviors can be manipulated deliberately with verbal thoughts and readiness which we call it consciousness [18] [38]. It is hard to weigh consciousness. Psychologists have been trying to map the consciousness since the creation of the discipline. They developed clever animal and infant experiments [27] [5] [36]. The new findings of the experiments continuously blur the boundary of what we concerned as consciousness and what is believed to rises from instincts. The emergence of artificial intelligence makes the distinguishment even harder [12] [25], as one might wonder if the underlying algorithms, under the same scrutiny, are analogous to animal intelligence. The comparison might provide directions that are beneficial for its current role of the problem solver or 'computer' in modern life.

1.1 Theories of Consciousness

Consciousness, as a global concept, is not directly studied in the discipline of artificial intelligence. Nevertheless, the effect of consciousness such as awareness [46] are well discussed and interpreted. According to **Integrated Information Theory (IIT)**, the consciousness of a system arises from the recurrent connections and deliberate structures that enable the agent to perceive the existed states and possibilities, store informa-

tion from the past and use them calculate the next move leading to a desirable future. According to this theory, consciousness can be calculated by measuring the communication efficiency and representation efficiency of the system [66]. The calculation method defined by IIT theory enables us to compare the circumstantial awareness of different algorithms which yield useful information about the efficiency of the temporal information propagation structures, providing a score that indicates the position of the algorithm in the hierarchical different levels of 'artificial consciousness'.

The idea that the level of consciousness is comparable between algorithms is supported by another important method (In the sense of responsiveness to conditioning stimuli [46]) that is popular in unsupervised learning and cognitive neuroscience: **Predictive coding** [39]. Predictive coding states that the comparison of the current outcomes and the previous ones enables the algorithm being sensitive to changes. With changes to be processed in successive layers, the method can model the structure of the problem and make predictions without direct contrast from the answers. The method is not stated to be directly related to consciousness, however, the process of abstracting changes implies a high-level interface which is identified to be similar to the top-down perception process in the brain. (This method enable the algorithm to focus on the divergence, the divergence can represent changes in the relationships that can be generalized to novel cases, this properties make it very useful in this project, see Chapter 2 and 8.)

According to IIT and predictive coding, the accumulation of information, the ability to integrate information of the current state, and make predictions are three crucial factors in our decision of whether the system is consciousness. The characteristics of structural consciousness are quantized in the measurement introduced by IIT, which makes the measurement efficient and delicate. Nevertheless, the measurement, by itself provides little insight into how ready the algorithms are for processing and integrating upcoming information streams. The readiness or responsiveness is what we usually consider to be consciousness, especially in experiments with human or animal agents [69].

To generate a full and accurate view of the human sense of consciousness in algorithms, we start by considering the consequence of consciousness. Consciousness distinguishes the property of the instant frame of time and makes sense of it based on the persistence of the past saved as neuro-biological habits. This habits cause permanent changes in structure or unconscious reflections [14] which induce visible behavioral or neuro-electricity reactions that distinguish us from people in the coma or vegetative

state [24]. The integrated meaningful 'present' generated by consciousness enables cognitive functions [2] so that we can think, compare, predict and feel. These cognitive functions then operate our perceptions in an organized meaningful way so that we live.

We do not usually assume algorithms have full consciousness that assembles to human. The artificial algorithms are usually assumed to have limited capacities and generate only repeated or pre-designed action fragments [4]. The exact difference is inaccessible. The divergence in scales of calculation methods in human and artificial intelligence make them incomparable: We cannot assign a consciousness value for the structure of the brain (yet) [41] and we usually feed to much additional information of relationships and structures to neural networks and their potentials remain undiscovered. Base on the low-grade-conscious assumption, we may ask: if the algorithm do not fully aware of the causation and correlative progressions introduced by the outside world, to what degree can it perform in human-level? Is it theoretically impossible to build human-like independent artificial intelligent agent that take on responsibilities in society as a human [55]? What distinguishes us from non-conscious matter?

Focusing on the effect of consciousness is an accessible way to address the problem. From expectations and responsiveness to high-level integration such as analogy and intelligence, we constantly make sense of the world we live in with the gift of consciousness, and the algorithms need those abilities to operate and solve problems. The comparison of signs of intelligence in problem-solving ability between human and artificial systems would provide useful insight into identifying the difference between human and non-conscious matters. The differences might then help us to better understand the concept of consciousness and intelligence.

1.2 Intelligence and Intelligence Comparison

Intelligence, a symbol of cognitive abilities, is often used to approximate the potential of an individual in a non-physical way. It is well acknowledged that the intelligence level of an algorithm can be measured and compared to humans. The most typical experiment involves Truing test where a machine is considered as intelligent if it is capable of behaving in a human-like way to a degree that the difference is indistinguishable for human [16]. Put it more aggressively, according to Turing, the consciousness or intelligence that showed by the machines is only accepted and recognized to the degree that it resembles human [58]. This result-centering mean skips efficiently the

awkward asymmetry between the absents of functional output in the algorithms and the leak of structural interpretation of brain which makes it possible to compare the high-level functions in a visible parallel level.

However, the lack of constraints in the process gives rooms for carefully calculated mechanisms that produce pretentious programmed outputs which are mediated by human thinking. The 'black box' scoring process blurred of the degree of thinking involved and give rise to the possibility of reducing the difficulty of generating convergent conclusions to a much easier comprehensive case match as finding a word in the dictionary. Additional constraints are necessary for concrete conclusions.

Considering the condition mentioned above, it seems that a direct uniformed scale that map of the structure intelligence of algorithms to human reasoning and problem-solving intelligence level is very meaningful and it can provide useful information benefit our current understanding of the origin of intelligence. To build such an intelligence scale requires increasingly sophisticated algorithm structures and an approbatory intelligent test that requires a minimal amount of special knowledge such as language and easily understandable for the intelligent algorithm agents.

Among the popular psychological intelligence test, Raven's Progressive Matrices (Raven's Matrices) is an admissive potential choice, it contains figures that embeds relationships or expectations that are crucial for real-life problem solving and thriving inducing little cultural bias. It is believed to be able to provide precise measurement for human fluid intelligence [10] [53] [48]. Agents need to abstract shape or figures and perceive the progressive logic to solve the cases [13]. In this project, I will build increasingly complicated algorithms to solve Raven's progressive Matrices, attempting to 1) identify the minimal structure or training that enables human-like deduction that serves as an important component of creativity and 2) illustrate the advantages and disadvantages of different algorithms, trying to find the best possible combination of algorithms which can improve the performance and yield intelligent responses.

1.3 Limitations of Previous Experiments

As illustrated in the above sections. There are very limited attempts aiming to provide direct contrasts of the intelligence level of algorithms and human. There are previous projects intending to solve the Raven's Matrices. The projects had made interesting signs of progress in solving the problems and the solutions are provided in different representational levels. Despite the fact that the goal-directed algorithms tend to yield

highly accurate performance, from facticity point's of view, they tend to be overly simplified or specified in two ways: firstly, they do not use the original Raven's Matrices and use a well-constrained set with a refined subtype of shapes and relationships. This makes the problems easier and incomparable to human participants' performance [25] [7] [74] [49]. And secondly, the projects tend to prepossessed the data to a huge degree that the information they feed to the algorithm is hard or impossible for an algorithm to abstract by themselves. This advert the problem and thus the result does not directly represent the intelligence of the agent, but the intelligence of the coder himself [34] [51].

The formerly mentioned problems will be attempted to be addressed in this project: Algorithms of different structures complexity (Convolutional Neural Network (CNN), Convolutional Neural Network with Feed-forward Neural Network (CNN+FNN), Recurrent Neural Network (RNN), and Convolutional Neural Network with Recurrent Neural Network (CNN+RNN), 4 different structures complexity) and 'experience' (different training method: integrated training, crucial contrast training, and all contrast training and different representational method such as figure and string representation) will be built to solve increasingly hard Raven's Matrices. We hypothesize that algorithms with more complicated structures will be able to solve increasingly hard problems with efficient experiences and the achievement of the basic algorithms might be comparable to human children participants.

1.4 Main Achievements

- Various aspects of intelligence can be represented by specific neural networks, namely Convolutional Neural Networks (CNN), Convolutional Neural Network with Feed-forward Neural Networks (CNN+FNN), Recurrent Neural Networks (RNN), Convolutional Neural Network with Recurrent Neural Networks (CNN+RNN, Chapter 3,4, and 5).

- The progression of the matrices requires more and more complex structures. From CNN to CNN+RNN, the algorithms show, respectively, the ability of visual identification, abstract and identify visual relationships, detection of change continuity, abstract time/step dependent relationships (Chapter 3,4, and 5).

- Comprehensive training methods with all possible contract and effective representation method embed the relationship between integrity and parts have posi-

tive impacts on the established ability (Chapter 3 and 6).

- The results are validated using an algorithm-oriented test (Calculation Salient matrices) designed in the experiment, the convergence affirms the conclusions (Chapter 7).

- Possible hybrid-algorithm and improvements are specified, open the possibility for an automatic problem-solving algorithm in novel environments (Chapter 8).

In the following chapters, general experimental methods (Chapter 2) and algorithm with different structure complexities will be introduced, respectively: Convolutional Neural Networks (CNN, Chapter 3), Feed-forward Neural Network (FNN, Chapter 4) and Recurrent Neural Network (RNN, Chapter 5). Problem-solving attempts, different training method, and other adjustments will be documented within the chapters. String representations (Chapter 6) will be included as a diagnose and potential evolutionary points of the current strategies. Validation of the methods and conclusions will be established with an algorithm oriented intelligence test (Chapter 7). Finally, comparisons between different algorithms, training or representational methods, and contrast against human children participants will be introduced in General Discussion (Chapter 8).

Chapter 2

Method

It is very common in researches to compare the performances of different algorithms [47] [8]. This comparison provides a global view of how the algorithm performance in the task. The comparison can often be made in the level of computational complexity, or efficiency [20] [47]. The diversities in advantages provide a potential match of their specialties to various tasks. Three types of basic neural networks and highly productive combinations of the algorithms: Convolutional neural networks (CNN), Convolutional neural network with Feed-forward neural networks (CNN+FNN), Recurrent neural networks (RNN), Convolutional neural network with Recurrent neural networks will be built in the following experiments. The algorithms used in the experiments are programmed with python on the platform of TensorFlow (see appendix C). The algorithms are expected to yield improving results or compensating each other in their capabilities.

2.1 Material

All training and testing materials are constructed directly from the standard intelligence test of Raven's Matrices [50]. Raven's Matrices contains 60 increasingly hard intelligence problems and it can be further divided into three parts according to the form of the problems (or more according to difficulties): The **integrated cases (problem 1-12)**, The **two by two cases (problem 13-24)** and The **three by three cases (problem 25-60)**. The relationships and properties embedded in the problems are summarized in figure A.1 in the appendix. The test is very mature and complete with all item difficulties and performs of different age groups well documented [54]. Given the standards, The result we obtained from this experiment will be compared with the documented

results in an objective and meaningful way. All images used in the experiments are extracted directly from the original file. All question and answer frames are cut without the frames and saved in numpy arrays.

2.2 Training

Just like in real life novel problem-solving instances, training is highly restricted by the available training materials in this experiment: Completely novel training are introduced to the algorithm at the beginning of each problem set, in other word, no overall pre-training is applied in any way, a new network is built at the start of each problem. This method is practical and similar to the human test procedure. It is proved to be very efficient in pre-experiment (appendix B): It is efficient and yield high accuracy. The pre-trained networks are usually in-adjustable and they are highly affected by noises. Additionally, the dimension of real-world problems are unlimited and unexpected, so comprehensive pre-training is not possible in solving real-world problems.

Since the problems are in the form of figures, Convolutional Neural Networks are used in a variety of occasions, directly or as a pre-processing phase. Kernel training is crucial for the visual identification to achieve good performance. To minimize the interference between recognition and relationship extraction and make the best use of the limited materials available in the training phases, two basic convolutional neural networks are implemented in the experiment. One for kernel training and the other use the kernels to make decisions to detect features (as part of the agent to be test). The agent, with the adapted fixed kernel, is expected to have the same training experience of the first convolutional neural network. The isolation of the training algorithm and the testing algorithm pre-excluded the possibility that the kernel weight is affected by posterior adjustments. In this way they can full-fill their role of 'feature detectors' parallel to the visual cortex [60]. After training, the kernels are abstracted from the convolutional layers and applied in the second convolutional neural network.

The parameters for the networks are primarily selected referencing the MNIST examples, taking the currently available training materials into account. A few trial runs are carried out before the experiment to make sure the algorithms are trained efficiently and achieve a good accuracy level. However, because of the existing divergence of training and testing methods, good training accuracy might not guarantee good judgments in error comparison phases (See general discussion). Parameter details are specified in each experiment.

2.3 Answer Selection

Special answer selection method is applied in this project to maintain the integrity of the algorithm and reduce artificial ad-hoc processing:

Most algorithms have or are attached to an output feed-forward soft-max layer to yield classification result. In this case, the direct method is proved to be inefficient. Firstly, there can always be more than one relationship to be abstract in one problem, integrating the binary output can be very difficult. And secondly, most relationships in the Raven's Matrices are not exact. Though the abstracted relationships and relationship to be decided share important crucial similarities, the similarity and not usually direct quantifiable (e.g. the change from a circle to a square is not the same as the change from a square to a triangle, but they might belong to one category in some problems). The output layer in the algorithms serves a better function in back-propagation than answer selection, and the **crucial layer** of the algorithms before the output layer is used to make decisions instead. To emphasis the divergent in the crucial layer, a method of error comparison which is commonly used in predictive coding is used: A **distinction** is computed for the output of the crucial layer, subtracting the output of training incidents (OT, with dimension d) from the candidate answers (OA, with dimension d); **The Correct Distinction Rate (CDR)** is computed with the sum of absolute distinction normalized by the sum absolute value of OT:

$$CDR = \sum_{d} \frac{|OA - OT|}{OT} \tag{2.1}$$

The answer with the smallest Correct Distinction Rate is selected as the answer to the given trial. If the correct answer is selected by the average Correct Distinction Rate across fifty trials, the problem is considered as solved. The main results will be displayed in graphs. Typically, Two graphs are produced for every experiment. One shows the main result: the number of trials that the correct answer is selected by the algorithm out of fifty trials; the other shows the CDR of the correct answer of the given question. The second graph is provided as a confident judgment. The lower the value is, the more confident the algorithm is selecting the answer. The validness of the error comparison method is further discussed in the discussion section.

More detailed usage and implementation are introduced in the following chapters along with the algorithms themselves. Training and answer selection methods are different in Chapter 6 and 7 where string or vector training data is used.

Chapter 3

Convolutional Neural Network (CNN) and Continuity

As suggested by the integrated information theory [66], the expectation of matching [63] and continuity [29], are important aspects of what humans and intelligent algorithms need to perceive the world and generate meanings. The requirement of expectation of the following state of visual stimuli in high-level functions is also implied in the first few Raven's Matrices. These matrices contain a contact image with a missing square region. Pattern coherence and shape continuity are needed for an algorithm to be able to solve them.

The expectation of continuity is identified to be one of the major visual clues that human use when processing the visual stimuli by the Gestalttheorie [68]. This visual organizational cue makes people see contact patterns emerging from its parts. In human, the expectation starts as early as children [63] and they present in early stages of vision [29]. For algorithms, these expectations are not systematically studied. Based on the acknowledgment of similarities in their inner hierarchically structure and identification ability between brains and 'intelligent' algorithms that use **Convolutional Neural Networks (CNN)** to realize visual identification [21], it is possible for an intelligent algorithm to share visual expectations. The remaining question is, however, how intuitive is the expectation in algorithms? Can the expectation be observed from basic algorithms such as Convolutional Neural Networks?

To test this idea, we build CNN algorithms and test their expectations by having them solve the integrated Raven's Matrices.

3.1 Method

As mentioned in Chapter 2, Two basic Convolutional Neural Networks are initialized and tuned in the experiment. In this experiment, three different contrast methods are applied to the kernel training Convolutional Neural Networks (2 layers, inputs: 60 widths * 50 height * 8 batch size):

Basic **integrated training** using parts of the original image (See red square in Figure A.2) against noise (80*80 random pixel picture); **Crucial training** which contrast against all different patterns in the problems; and **All contrast training** that contrast against all available images despite the overlap in crucial pattern (consider different densities to be different features). This distinction aims to provide information about the usage of different information in training and answer selection. We assured all the kernel training to be terminated at 1.0 accuracy (5-100 epochs of training) if the kernel training does not reach 1.0 accuracy at the 100th epoch, the attempt is excluded from the analysis.

After training, the kernels are abstracted from the convolutional layers and applied in the second CNN algorithms. Distinction is computed for the outputs of the CNN algorithms for part of the original image that used in integrated training and the output of the possible answers of the same size (See orange square in Figure A.2); The Correct Distinction Rate (CDR) is computed and the answer with the smallest sum of absolute normalized errors is selected as the answer. To make the point of texture change more understandable to the algorithm, the answers are constructed to include part of the problem picture (5 pixels at 4 edges) in crucial training and all contrast training cases. As in Figure A.3, the selections are constructed based on context and available answer.

The algorithm makes 50 independent choices (starting from the kernel training and process to error comparison) for each problem. The Number of correct trials, the Correct Distinction Rate, and the averaged error rate for all answers across 50 trials is recorded.

3.2 Result

The Correct Distinction Rate and number of correct trials are displayed in Figure 3.1 below and Figure A.4 in the appendix. Considering the averaged error rate, the number of problems solved by the algorithm (out of 7) using different training methods

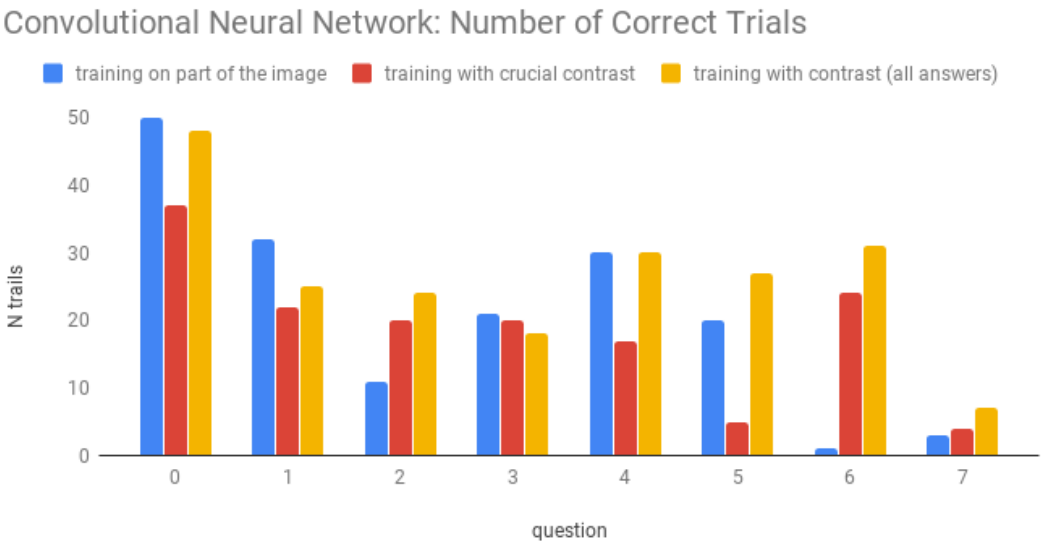

Figure 3.1: This picture shows the number of correct trials of the first seven problems. All methods (especially training with all contrast method) show constant, acceptable performance in the first six problems, which is clearly above chance level (8.33). All problems except the second problem are solved by all contrast training method (indicated by average Correct Distinction Rate across 50 trials), while the second problem contains too many blank images (only noise is available), it might be confusing for the training method. The second problem is solved by other training methods.

(integrated training; crucial training and all contrast training) are respectively, 3,3, and 6. The algorithms are very efficient as well. They take little time to train the kernel and chose the answer.

3.3 Discussion

The algorithms show interesting signs of progress in solving the first 12 Raven's Matrices problems. The algorithms processed to the 7th problem, prior to which point, texture match is crucial and basic. It can be hard to proceeded further when the pattern change continuity plays a major role in determining the answer. The 7th to 12th problems can not be efficiently presented to CNN algorithms in a solvable way. The experiment is carried out acknowledging the inefficiency as a baseline performance for other algorithms (See figure A.11 and 5.1). For even more complicated two by two or three by three cases where the relationships need to be abstracted explicitly, Convo-

lutional Neural Networks are only used to prepossess the images. Since the algorithm, by itself, is not expected to be able to understand relationships between pictures (See the following chapters).

This partial success indicates that the algorithms are capable of identifying texture or patterns. Especially with sufficient experiences (all contrast training). The result is expected given the fact that convolutional neural network is commonly used in patterns recognition [45]. It is more clear in this experiment, however, the pattern match expectation starts from within the Convolutional Neural Network itself, independent to further processes and answer selection. This process achieved by CNN algorithms might be parallel to the first stage of recognition in the brain where the features are identified and before feature integration and naming [67]. Both brain and CNN algorithms perceive patterns as features in this stage, the categorization of features, however, can be very different. For example, dense and shape might be considered as separate traits or properties of a picture for human, while shape with different dense are better considered as totally different patterns for an algorithm to fully distinguish against them (See the result of crucial training and all contrast training).

The different 'experiences' modulate the generation of pattern continuity expectation. Crucial training and all contrast training are proved to be beneficial providing additional contrasts. This is coherence with previous studies [25]. The conclusion regarding experience can not be generalized to human directly [56]. Bur from an evolutionary point of view, it is possible that the accumulation of experiences across and within generation have some effect on the 'instinctive' expectations such as continuity [17].

The algorithms cannot solve problems seven to twelve. These more difficult problems might require different representations that enhance temporal changes. This 'change continuity expectation' can not be generated by CNN. Many additional attempts were carried out: We tried to emphasize shape continuity by emphasizing the boundary error or induce random attention but the attempts were proved to be rather confusing for the algorithms (see appendix B.3).

The unpromising results further indicate the potential difference in representation of features between the algorithms and humans, we can infer that the algorithms do not have expectations of shape continuity or pattern change continuity as children [63]. However, given the established conclusion of the effect of experiences on learning, it is still possible that there exist some advanced, more controlled training methods which enable the algorithms to generate such expectations and solve the more complicated

problems. On the other hand, modifications or enhancements in structures might be necessary for the expectations to find its place to be generated and expressed, this idea will be further discussed in the next chapters.

In conclusion, CNN algorithms are excellent at visual identification, but they do not understand progression or dense.

Chapter 4

Feed-forward Neural Network (FNN) and Relationships

In the last chapter, we summarized the limited ability of CNN algorithms in solving intelligent problems. It turns out that the feature detectors or kernels, by itself, is capable of generating basic pattern matching expectation through error comparison. However, it is less effective in generating progressive expectations of pattern changes. More complicated structures might be needed for the more complicated relationships to carry the ability of the algorithms beyond the barrier of pattern recognition, convolutional neural networks with feed-forward neural networks (CNN+FNN) is used in this chapter.

Feed-forward Neural Network (FNN) is a typical, high-performance kind of additive process for Convolutional Neural Networks so that a decision can be reached [35] [70]. Some literature documented that the FNN originated inspired by the massive saturating feed-forward connections in the brain [40]. It was considered a huge evolution of the computational methods comparing to human-defined function propagation [64]. This type of algorithms is excellent in abstracting underlying high-level structures automatically, and it has been widely applied in a broad range of delicate tasks, successfully eliminating tedious manual filing or identifying tasks [64] [9].

The core mechanism of how or what principal components are identified is studied broadly in neural computational studies contrasting to Independent Component Analysis (ICA) or human. Similarities, differences, and simplifications have been noted [1]. Regardless, given its ability to abstract relationships well acknowledged, it is very interesting then, whether the feed-forward neural network is adequate in abstracting the implicit principles behind Raven's Matrices.

In this chapter, we will build a Feed-forward Neural Network to process the result generated by the Convolutional Neural Network. As in the first experiment, the Correct Distinction Rate produced by different depths of the layers will be produced and contrasted for available answers.

4.1 Integrated Cases

Beside the convolutional neural network used in Chapter 3, a one-layer feed-forward network is appended (CNN inputs: 60 widths * 50 height * 8 batch size, max pool kernel size 5; FNN inputs: 36 inputs * 8 batch size, 1024 nodes per layer). The Correct Distinction Rate is computed for the feed-forward layer using the same manner. One additional change is made, namely, random attention is applied to the new networks. It makes more sense to have a comprehensive comparison with different (random) part of the image and it should provide more information when the patterns are not constant. The feed-forward layer does not improve the performance of the algorithm to a huge degree. The result is shown in Figure A.5 and A.6 in the appendix.

4.2 Two by Two Cases

Just as the integrated cases, a 3-layer feed-forward neural network is attached to the convolutional neural network. **As training materials**: all no-overlapping single patterns in problem (questions and answers) are passed to the kernel training Convolutional Neural Network. Training is limit to 200 epochs for the and the cases where the classification did not reach 1.0 accuracy will be replaced by a new round of training (inputs: 50 widths * 40 height * 8 batch size, max pool kernel size 3). A limit of 50 epochs of training and 0.8 accuracy termination is applied to the feed-forward neural network (input size: 180 inputs * 8 batch size, 1024 nodes per layer). Low accuracy trials are not typical but their final decisions are also included. **Training for Feed-forward Neural Network** focuses on the contact relationships in the problems that are specified by the distinctions of figures. Additional comparison is made to make the relationship instead of the pattern itself more salient for the algorithms: the distinction in output of the Convolutional Neural Network for two pictures in the contact (first) row (**row relationship**) and contact (first) column (**column relationship**) are passed to the feed-forward neural network. The feed-forward neural network is expected to distinguish against the row relationships and column relationships producing a Correct

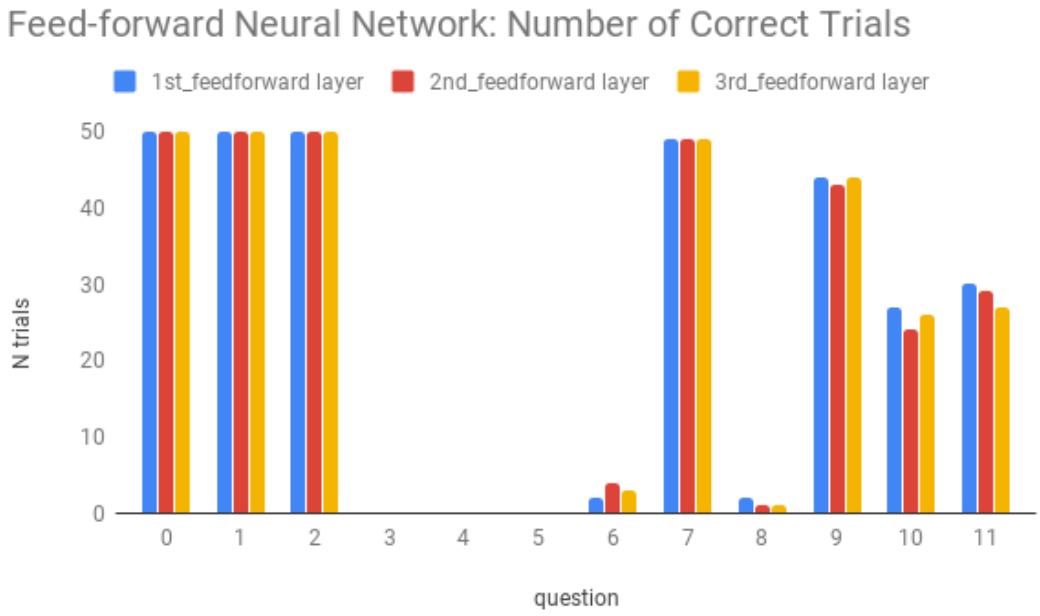

Figure 4.1: This picture shows the number of correct trials of the two by two Raven's Matrices problems (problem 12 to 24), contrasting the three layers of the feed-forward neural network. As can be seen in the picture, the performance of the algorithm is very consistent across trials, five problems are solved more than 80%. Totally seven problems are solved by the CNN+FNN algorithms. The problems that are not solved involves relationship progression and pattern computation (integration).

Distinction Rate for each candidate answer. The Correct Distinction Rate is calculated using the distinction in the feed-forward layer for filled-in row and column and the original ones. The answer with the lowest CDR is selected as the answer. A separate comparison is made for each layer in the Feed-forward Neural Network. The experiment is run for fifty trials and the number of correct trials and mean CDR for the 12 problems is shown in figure 4.1 below and A.7 in the appendix. The result is concrete and consistent. The second and third feed-forward layers do not provide further improvement or difference (see discussion for details).

4.3 Three by Three Cases

As Chapter 4.2 pre-selected features that are mutually exclusive is passed to the kernel extraction Convolutional Neural Network (28 widths *28 height * 8 batch size, max

pool kernel size 2). The training limit is extended to 300 epochs accommodating the increase in the number of crucial features. For Feed-forward Neural Network, 4 contact **column relationship** and 4 contacts **row relationship** in the contact rows and columns (two relationships in one row/column: the relationship of the first and second question picture; the relationship of the second and third question picture). The Feed-forward Neural Network is trained to distinguish against the 4 columns or row relationships (294 inputs * 8 batch size, 1024 nodes). It then identifies the answer that yields the most similar output in the feed-forward layer for the relationships in the row/column to be filled. The algorithms show very limited ability in solving the cases (see figure A.8 in the appendix). The additional axis did not provide a meaningful improvement in the results (see figure A.9 and figure A.10 in the appendix).

4.4 Discussion

The feed-forward network is proved to be efficient in solving some of the problems especially those with simple relationships such as match, same or same divergence (see figure A.1 in the appendix). The relationships based clustering of solvable cases proves the efficiency of the algorithms in abstracting relationships [64] [9]. The efficiency of the algorithms reduced dramatically when the difficulty increases. The algorithms are very inefficient in dealing with the representation of part and integrity. This deficiency is understandable. Neural networks are delicate devices that are designed to be able to sustain the relatively stable attractors against the ever-changing environment. It is not their innate job to identify the most powerful representation of a problem than emphasizes the similarity across or generate expectations within limited samples (See chapter 6 for more details).

It is quite promising, however, how some of the problems being robustly solved by the feed-forward neural network, regardless of the misleading information such as the noise level, the slightly distorted positions of different images, and the variety of unspecified novel relationships. With limited examples and training, CNN+FNN algorithms can generate and express their expectations of relationships in the first feed-forward layer and generalize the expectations into unseen novel cases with unforeseen patterns, even if the problems are hard and no intuitive. The additional feed-forward layer did not tend to yield any additional improvement or difference in the result. An unsuccessful attempt trying to make use of the deeper layers applying escape connection failed to achieve its goal (See appendix). The attempt reflects, however, The

solved cases are coherent across experiments. It can be inferred that the relationships of match, same or same divergence might appears in a very early stage of computation. Unless specific different information is provided from elsewhere, no higher-level representation is available or essential. The formation of shallow relationship abstractions processes might also have something to do with the fact that the kernels weights and the feed-forward weights are trained separately. And thus no additional information (such as attention) can be co-trained by the different neural networks. And thus the relationships of all features are being processed at the same level. If more training examples are available, it can be meaningful to find a way to efficiently co-train the networks or define a hierarchically structured model to be able to abstract more complicated relationships [59].

In conclusion, CNN+FNN algorithms are efficient in abstracting relationships, but the ability is limited and the representation of relationships tend to be shallow.

Chapter 5

Recurrent Neural Network (RNN) and Change Continuity

Recurrent neural network (RNN) have been used extensively in demanding tasks such as speech recognition [43], visual reasoning [25], picture captions [73], and attention [72]. It had been proved to be efficient and accurate.

Recurrent neural networks are distinguished by recurrent connections embedded in the structure. Recurrent connections in neural networks enable them to combine activation from earlier states with current inputs, so that information from a sequence of states can affect the output [73]. This advanced characteristic parallels to the hippocampus, the organ that is believed to be responsible for long-term memory [23]. With the inherent structural related premise, Recurrent neural network can automatically abstract temporal or step-wise relations. This ability might enable this type of neural network to have continuity expectations in the temporal domain and have a distinctive performance in step-wise relationship abstraction. Some studies conclude that recurrent neural network achieves a better or equal level of performance in relevant tasks [57]. It is interesting then if the recurrent neural network can abstract more complicated relationships in Raven's Matrices problems and achieve better performance.

5.1 Integrated Cases

Integrated problems (especially problem seven to twelve) requires the understanding (or expectation) of continuity of pattern or pattern changes. It is proved (in the first experiment) to be extremely hard for the convolutional neural networks to generate expectations of pattern changes, especially in hard cases. The temporal remains might

provide useful insight into solving problems of this kind. In this experiment, a recurrent neural network (alone) is used to solve the integrated cases of Raven's Matrices.

The complete part of the image on the left-hand side of the blank is used as **training range**. Training images (130 widths of the picture * 20 time-steps * 128 batch size, 128 LSTM nodes) are constituted within the range (see figure A.2. Multiple random starting points are selected within the training range so that the algorithm can be less biased by the starting position. The algorithm is trained against patterns in other problems (partial replacement, the size of answers) in the attempt to avoid direct confirmation of the answers (an additional experiment with similar training and parameter as experiments in Chapter 3 is presented in the appendix). The algorithms are trained up to 200 epochs (batch size 128) and allow terminations at an accuracy level 0.9 to avoid over-fitting. It is very fast to train the algorithms, it reaches a high accuracy level with in the first 10 epochs at most time. The answer frame is constructed by stacking the 'above the blank area' of the question picture with the answers (The answers have length forty, they are cut in the middle and the two half-answers are used to form two separate selections with length 20 like the training set.) Two selections are constructed for each given answer for each problem. The output of the recurrent layer (128 units) is compared between answers and random correct training images (constructed use part of the question picture). Distinction rate is computed, and the answer with the lowest Correct Distinction Rate (for the two selections of an answer) is selected by the algorithm. 50 trials are run for each problem. The Correct Distinction Rate and the number of correct trials are reported in Figure 5.1 below and A.11 in the appendix. As can be seen in the figure, although the algorithm did not perform as well in the simplest cases (1-6) it does show improvement in the harder cases (7-12).

5.2 Two by Two Cases

Given the fact that basic pattern recognition is needed before step-wise relationship extraction, the recurrent neural network is attached to the Convolutional Neural Network just like Chapter 4.2. The algorithm (with two timesteps) is designed to distinguish between column relationships and role relationships just like the Feed-forward Neural Network (RNN inputs: 180 inputs * 2 timesteps * 8 batch size, 128 LSTM nodes). The Distinction rate is calculated for the answer column and answer row. The result is shown in Figure A.12 and A.13 (see appendix). Unfortunately, CNN+RNN algorithms did not provide a better result comparing to CNN+FNN algorithms as expected.

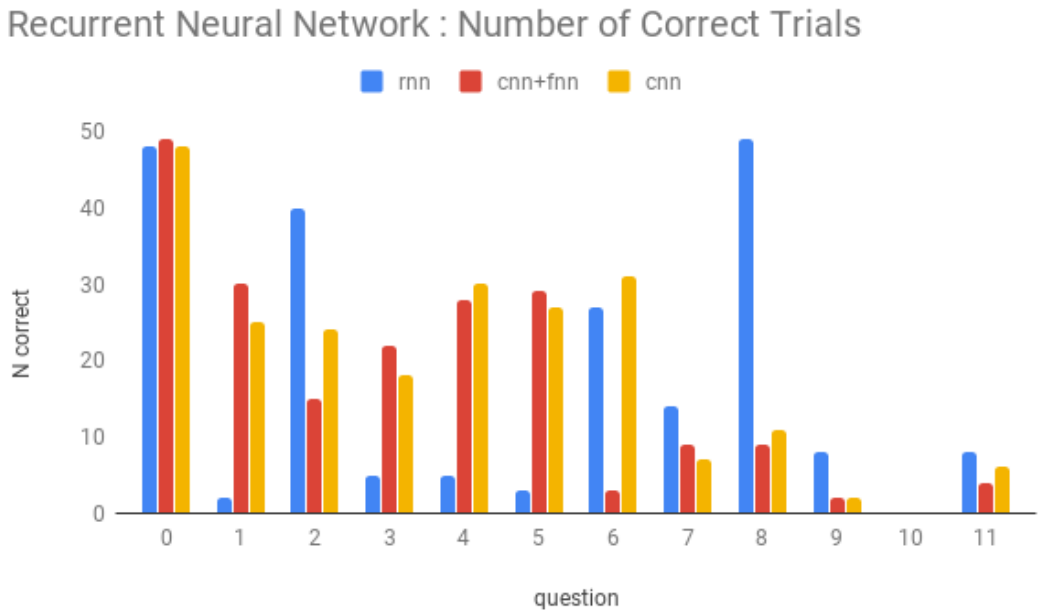

Figure 5.1: The number of correct trials of the integrated case problems, contrasting the convolutional neural network, convolutional neural network, and feed-forward neural network and recurrent neural network is shown in the figure. The recurrent neural network did not yield good performance in problem two to five. This might result from the stripe-wise representation makes the network more prone to noise. The network does yield better performance in more complete cases, for instance, problem seven, eight, nine, or eleven.

5.3 Three by Three Cases

Just like two by two cases, a Recurrent Neural Network (3 time-steps) is attached to the Convolutional Neural Network and it is trained to distinguish against two rows' relationships and two columns' relationships (The recurrent time steps gives the possibility of representing the relationship between three pictures in a row/column directly, so only one relationship is needed for one row/column. RNN inputs: 36 inputs * 3 timesteps * 8 batch size, 128 LSTM nodes). The CDR is calculated for the third row for each candidate answer. The result is shown in Figure 5.2. As can be seen in the picture (comparing to Figure A.8), the recurrent neural network can solve slightly more and harder cases than the feed-forward neural network, eight problems are solved by the average CDR across fifty trials, but the result remains unpromising.

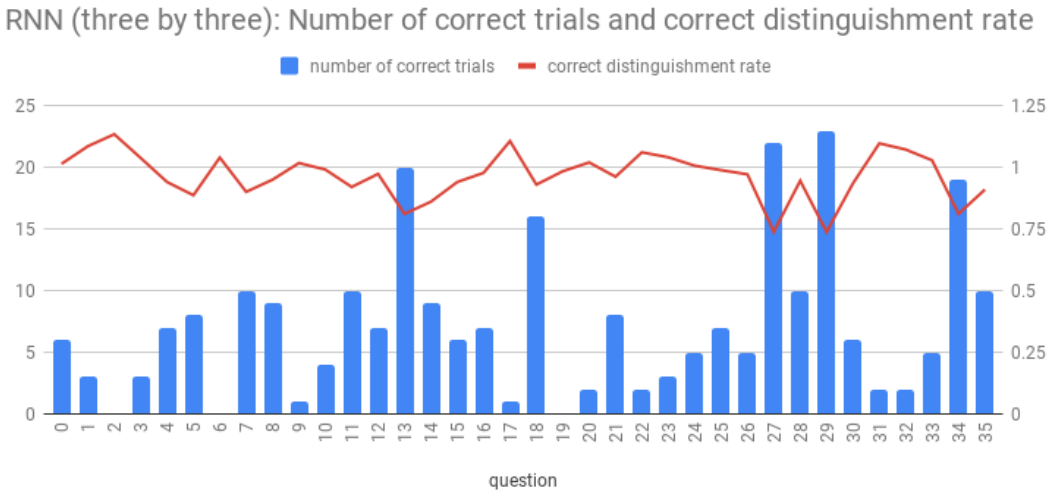

Figure 5.2: This picture shows the number of correct trials (out of fifty) and Correct Distinction Rate produced by the Recurrent Neural Network. As can be seen in the picture, the selections are not as coherence between trials as other cases. But there are still some problems that are solved relatively consistently across fifty trials by CNN+RNN algorithms comparing to chance level (6.25 trials), for instance, problem 13, 18, 27, 29 and 34. The formerly mentioned problems are all considered to be solved according to the average CDR across 50 trials. As the CDR and number of correct trials are shown in the same graph, a clear trend can be seen where the algorithms yield lower average CDR when they tend to give the correct answer.

5.4 Discussion

RNN algorithms show signs of improvement compared to CNN algorithms and FNN algorithms in some cases. It provides meaningful compensations in solving hard integrate cases and three by three cases problems. This result is confirmed by some previous studies where Recurrent Neural Network shows the ability to abstract relationships in picture matrices [25].

Unlike the CNN algorithms, the RNN algorithms show the basic expectation of change continuity. It can solve or achieve better performance in some harder integrated cases Raven's Matrices. This result is meaningful. 'Change continuity expectation' can be important for an algorithm to be able to make predictions or recognize patterns in continues data sets, such as motion [28]. This result can not be generalized into making conclusions about the structural basis of change expectations in human

[56]. However, the result confirms that the proper use of the Recurrent Neural Network enables the representation of stage-related situational prior. This stage-related expectation might be beneficial in surrounding exploration, etc.

Better representation of the stimuli might be needed for the RNN related algorithms (RNN algorithm and CNN+RNN algorithm) to achieve better performance. The stripe representation of pictures Chapter 5.1 or the outputs of the Convolutional Neural Network used in Chapter 5.2 can provide good solutions occasionally, but the lack of meaningful structural representation might lead to unwanted deviations abstracting only noise or inefficient features reserved from the CNN part of the algorithms. Again, co-adjusting of the kernel and recurrent weights might mute the problem, however, more instances might be needed for efficient training to be achieved. **Skip-over relationships** which categorizes the relationship between the first and the third picture in three by three problems are not considered in CNN+FNN and CNN+RNN algorithms. This relationships, however, might be beneficial in solving some problems.

In conclusion, RNN and CNN+RNN algorithms are good at abstracting time/step related relationships, but the ability might be limited by inefficient representations.

Chapter 6

String Representation and Associations

All algorithms fail to solve the hard three by three cases. This can indicate that they failed to abstract hard relationships between the frames. However, another explanation might be more possible and convincing. Even though the focus of the previous chapters is abstracting relationships, the representation of the problem interferes the process to a huge extent. A trial run using CNN+FNN algorithms without error comparison method concludes that the algorithm lacks the basic ability to identify the relationships between parts and integrity. The algorithm, by itself, has limited capabilities that are largely restricted to the simplest cases (see Figure A.14 in the appendix). The two cases might be categorized in the same category according to us, however, the second one is considered to be a lot harder for the algorithm, especially when only trained with crucial features, namely the three circle of different size (this is co-validated in Chapter 4.2, see result A.1 in the appendix). The problems can not simply be solved by training on parts, because the association between patterns and integrate is not understood. Existence of different representations is largely prevalent across the problems.

A similar problem occurs in the application domain of literature translation where different meanings of a word need to be selected by the context. This problem is solved by introducing an efficient representation: with the relationships between different words embedded in the vectors representing the word [61]. The method had been proved to be efficient, generative, and accurate [31]. Can the associations be efficiently represented and understood in the vector representation manner in the current experiment? In this experiment, we will explore this possibility by manually re-encoding the picture frames of hard three by three problems into vectors.

6.1 Method

Each problem contains 16 **picture frames** (8 **question frames** and 8 **answer frames**). We re-encode these picture frames into three key features: size, shape, and position. The default allocation (of space) for those properties is three bits for size (size for three shapes), four for shape (three shapes and other shapes), and six for positions (three for x-direction and three for y-direction). In the last few cases where more crucial shapes are involved in the questions and answers while the positions remain unchanged, four more bits are assigned to shapes and only two for positions. After the re-encoding, each problem can be represented with a 16*13 vector containing questions and answers.

Since the shape recognition is already realized manually, the Feed-forward Neural Network alone is used to abstract the relationships. The algorithms are trained on eight row-relationships including the skip-over relationship (the first and the third picture in a row) and **progressive relationships**(first and second or second and third picture in a row). The same error comparison method is applied to force the algorithm to focus only on the changes. Because there is absolutely no noise, for some simple cases like Figure A.14, there is no divergence in row relationships, the error comparison is applied directly to vectors and the problem is not proceeded into the feed-forward layer (the lack of divergence in relationships can be confusing in training). The maximum training iteration is set to be 500 and allow termination with a 0.95 accuracy level.

Correct Distinction Rate is computed for the third row (to be complete) in the same manner using the average output of the two complete rows as references. Two output is provided, one includes the comparison of skip-over relationships and the other does not. When the skip-over relationship is included, one-third weights of importance is attached to the Correct Distinction Rate produced by this kind of relationship. And Two third is attached to the two continues progressive relationships.

6.2 Results

The result is shown in Figure 6.1 below and A.15 in the appendix. As can be seen from the figures, a lot of problems are successfully solved and the performances are ceiling, especially when including the skip-over relationships in error comparison.

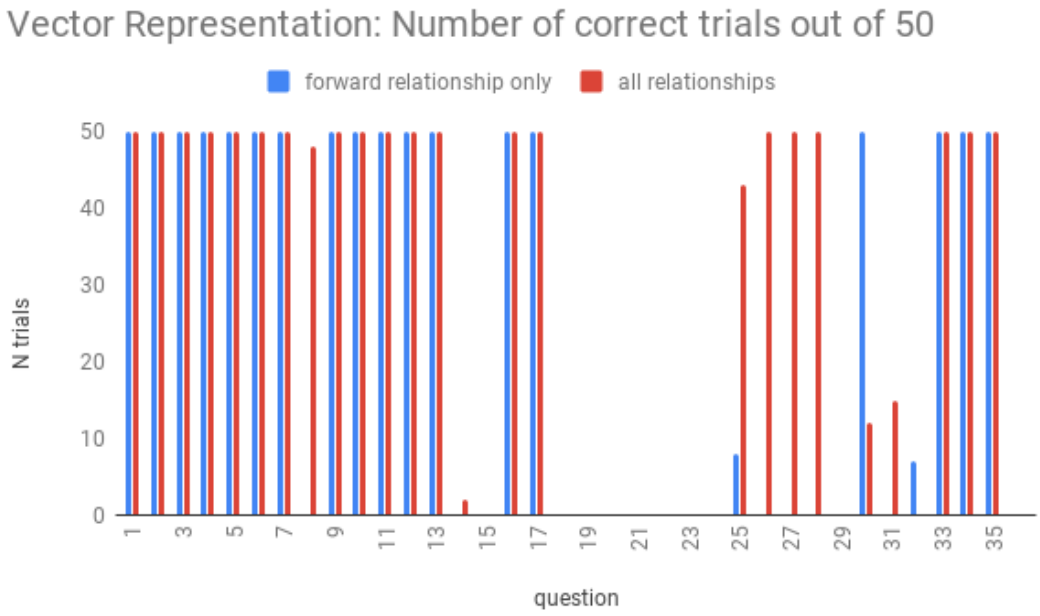

Figure 6.1: This picture shows the number of correct trials (out of fifty) of the three by three case problems using vector representation and feed-forward neural network, contrasting the result including only the progressive relationships and all relationships. The performance of the algorithms is coherent across trials. And it reaches ceiling performance in many problems as well. For the problems that are not solved by the new representation, many require permutations of the given patterns.

6.3 Discussion

It is promising that the various relationships can be successfully abstracted using the new representational methods. This concludes that the inefficiency of the algorithms mainly lays in the ineffective representation instead of capability in relationship abstraction. Of those failed cases, a lot of them require the permutation of three items at least in one property. This might indicate that the algorithms have their limitations in what kind of relationships to be abstracted.

This research confirms the conclusion that feed-forward neural networks are excellent at extracting established relationships with training [64] [9]. In this case, the relationship abstraction process is proved to be efficient within the first feed-forward layer and given only a few examples. This result is promising in that it suggests a possibility of solving highly targeted unseen problems in novel numerical represented environments.

There are, however, two remaining problems to be solved. First and most importantly, there is no established way for algorithms generate vectors automatically facing a new environment. The generation includes (mainly) identifying crucial parts and properties and fill the crucial values accurately. While the filling can partially be realized by the development of mask recurrent neural network [37] [71], there is no obvious solution for the identification. The second problem is, how can the algorithm be trained to understand the 'missing' relationship. This identification can be important in finding the loose chain in logic which is important in creative thinking [6].

In conclusion, the result of string representation is promising but the practical implementation is limited at this stage.

Chapter 7

Intelligence Measurement for Algorithms

In the previous chapters, we measured the intelligence level of algorithms using Raven's Matrices. Algorithms show their problem-solving abilities in abstracting relationships and make generalizations (chapter 3 to 6). The limitations, however, is just as salient. Part of the limitations results from the test itself. There are multiple reasons why Raven's Matrices are not designed for algorithms. The lack of training examples and difficulty in representation make it extremely hard to make direct comparisons between algorithms without induce artificial pre-process that make the calculation happen. In this case, we use error comparison methods and string representations of stimuli. The methods are proved to be efficient while the application of the methods themselves result from the intelligence of the designer. One might ask that, Is there a way to generate an intelligence test for the algorithms that make the ability to abstract and generalize relationships salient? The test also needs to be easily adjustable so that it can be applied directly to any given algorithm with different structures and comparison method?

If the description in the former paragraph about the intelligent test for algorithms is possible, we would expect it to be in the form of strings or vectors (given the result in Chapter 6), embedded with the relationship characteristics that are related to fluid intelligence. This chapter documents a trial experiment aiming to define such an intelligent measurement using number matrices and directly measure the abilities of algorithms in abstracting different kinds of relationships.

7.1 Method

The **Calculation Salient Matrices (Salient Matrices)** are designed based on the relationships/combination of relationships summarized in figure A.1 in the appendix. It contains ten increasingly hard problems which can be divided into four sub-parts. The problems are constructed by a **test generator** The vectors generated by the test generator are arbitrary in shape and size (One to three dimensions), embedded with relationships that are similar to the Raven's Matrices. The first two cases embed relationship match and same. Problem three to five embed single relationship of 'progression' and 'missing'. Problem five to eight embed more than relationships in the vectors such as 'match and progression', 'missing and same divergence', 'progression and progression' the different relationships appear in turns or in different dimensions. Problem eight to ten embed relationships **numerical add** (Fibonacci sequence) and **arithmetic progression** (arithmetic progression with a tolerance of two). The two sequences with numerical properties are attempts to approach the 'add/minus' and 'numerical' relationship in Raven's Matrices without the property of shape and part. The algorithms used to generate matrices are presented in figure C.1. To achieve variety in training and distinguish test set from previous training examples, a random number of zero to twenty is added for each problem during training while a random number of twenty to thirty is added during testing.

Algorithms used in chapter 3 to 5 are directly used to carry out the test. The selected algorithms are the kernel training CNN+FNN algorithm (since the data comparison method is not used, the FNN layer is crucial for obtaining the output) used in Chapter 3, The 3-layer FNN algorithm used in Chapter 4.2, and the RNN algorithm used in Chapter 5.1. All the algorithms have been proved to have some capability in the problem they intend to solve. CNN and CNN+RNN neural network are not selected. The former is insufficient in yield result without additional feed-forward layer and the latter contains possible undesirable interactions between algorithms that might constrain the further analysis.

Four training conditions are used contrasting the first two, five, eight or all ten relationships. Ten novel examples of each trained relationships is used in the test phase. The rear random number added to the matrices ensures the test examples to be progressively hard and unseen.

Fifteen training sections (150 tests * 4 training condition * 3 algorithms) are carried out for the experiments.

7.2 Result

The result of the algorithms solving Calculation Salient Matrices comparing to the result of Raven's Matrices is shown in figure 7.1. For the number of problems that the algorithms are capable of solve, Recurrent neural network has the unique strength comparing to the other algorithms: Feed-forward neural network proceeded to the eighth problem, convolutional neural network proceeded to the second, while recurrent neural network proceeded to the last problem. When it comes to accuracy, the recurrent neural networks achieve the best accuracy for the training sets two (average accuracy, RNN: 78.7 %, FNN: 61.7 % and CNN: 57.5%) and ten (average accuracy, RNN: 63.7 %, FNN: 10.0 % and CNN: 12.5 %) while feed-forward neural networks achieve the best accuracy at training set five (average accuracy, FNN:57.5 %, RNN: 44.5 %, CNN: 12.5 %) and eight (average accuracy, FNN: 75.4 %, RNN: 62.0 % and CNN: 12.5 %). The result has a similar trend with Raven's Matrices problems while the average accuracy level raises. This result is very meaningful because the Calculation Salient Matrices and Raven's Matrices also shares the same Construction structure. The structure ensures that more complicated cases included the relationships of the previous examples but are not limited to them. The result indicates there is at least some similarity in the underlying abilities required by the problem sets. The accuracy of all the training conditions and algorithms is shown in figure A.16.

7.3 Discussion

The results are promising as the algorithms show different abilities in solving different kinds of problems. The comparison is direct and genuine. The results co-validate the conclusions in the previous chapters: Convolutional neural networks are efficient in match cases, Feed-forward neural network is efficient in abstract relationships, Recurrent neural network is sensitive to change continuities with the ability to solve some of the most complex cases, the string-representation is 'algorithm-friendly', the representation improves the performance to a huge degree.

The Calculation Salient Matrices is efficient in telling apart the abilities to abstract different kinds of relationships. However, it is equivocal in reflecting the cognitive ability that being considered as 'intelligence'. It is really hard to tell what kind of relationships, expectations or behavioral is considered to be 'beneficial' for the algorithms [18] and it does not directly identify how those abilities will benefit us in different

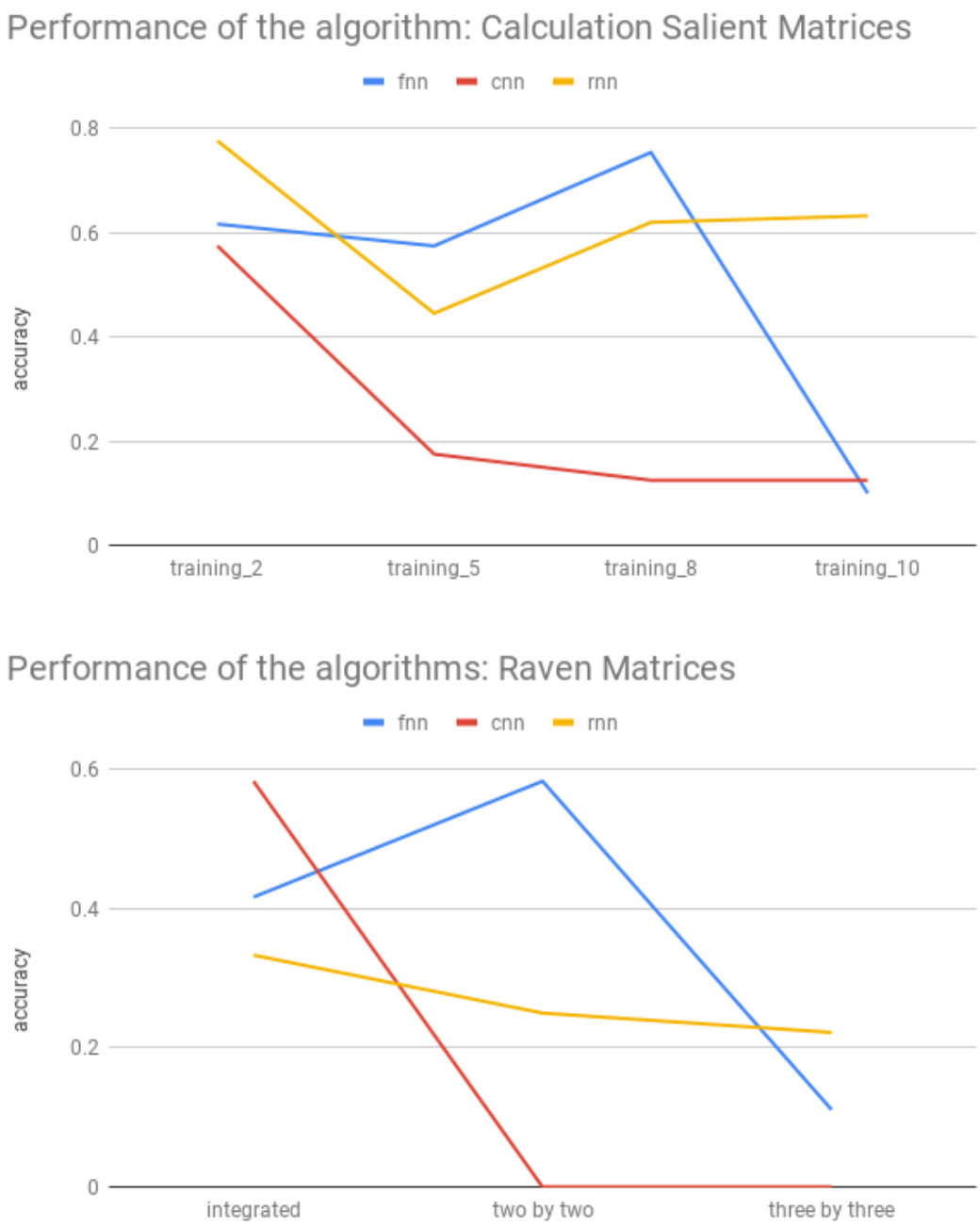

Figure 7.1: The figures show the result of the different algorithms on solving Calculation Salient Matrices (top) comparing to Raven's Matrices (bottom). The x axis defines the training and testing set of problems, for example, 'training_2' means the algorithm is trained and tested with samples generated by the first two generators only. The results mean: different algorithms show different strengths in solving problems with different relationships. The feed-forward neural network performs constantly well in solving the first eight problems. Convolutional neural networks yield acceptable solutions for the first two problems while they can not proceed to the rest of the problems. Recurrent neural network shows its unique strength in solving the first two problems and last two problems.

programming tasks. It does, however, provide useful information on how the structure of the algorithm can constrain or promote the problem solving or relationship abstraction process, which can be useful in validating the efficiency of new structures. It also allows direct comparison between different algorithms that are designed for different tasks.

Additionally, the reliability and validity of the test is needed to be established before the test can be used in practice. Reliability test is carried out for different dimensions: The form of data required by different algorithms can be different. In this experiment, for example, two-dimensional data is required for CNN and RNN algorithm while one-dimensional data is required for FNN algorithm. An additional test using two-dimensional data is carried out for the FNN algorithm and it achieves similar result comparing to the one-dimensional data (see figure A.16, additional validation test can be found in figure A.17 in the appendix). This partially convinced that the relationship embed process is successful and the difficulty level is similar across dimensions. However, the conclusion can not be generalized to three-dimensional cases, which is considered to be unsolvable for the feed-forward neural networks. Further standard can be hard to generate given the limited structure types of existing neural networks and the limitation in required data form for different algorithms. The comparison can be more meaningful facing diverse structures.

For the results that have already been established, CNN algorithms do not have the ability to abstract relationships, they can only solve simple problems with 'matching' or 'same' relationships. The limitation of its ability might induce heavy constrains to addictive layers. The deficiency may be compensated to some degree by the feed-forward or recurrent layers afterward, where the attention can be trained along with the importance of features. But direct representations of the relationships of patterns in CNN algorithms might improve its ability and efficiency. Some variations in structures have been established for algorithms with similar functions, for example, Hopfield neural networks can provide meaningful representations of relationships of patterns. This might provide and more realistic possible improvements in this first stage of image processing [32]. And board it usages in different domains besides visual identification [45].

In conclusion, the merits of algorithms are valid, but the performance can still improve.

Chapter 8

General Discussion

In this thesis, neural networks of different kinds and complexities were implemented to solve Raven's Matrices problems. Comparing to previous experiments [25] [34] [7] [74] [49] [51], the current project 1) induce minimal human categorization and prepossesses, 2) contrasting against different algorithms and identify their respective specialties, 3) gives comprehensive justification and validation in methods and results, and 4) yield constructive foresight in structural improvements and possibilities of hybrid algorithms. The experiments achieve tentative but promising results.

8.1 Main Results

The result of all algorithms (CNN, RNN, CNN+FNN, CNN+RNN) and string representation method is shown in figure A.1 in the appendix. The algorithms have different characteristics in the problems they can solve.

The integrated cases require the expectation of pattern continuity and change continuity. These are mostly basic expectation that starts from human infancy [63] [29]. The algorithms do show restricted, experienced determined expectations after specified training. But their abilities in solving the problems are highly restricted and directly related to the structure of the neural networks: Pattern continuity can be realized by a pattern match function which achieves better performance with all contrast training in CNN algorithms while the pattern change continuity can only be abstracted to some degree by the RNN algorithms.

When it comes to discrete relationships analogies, the CNN+FNN algorithms achieve good and stable performance in two by two cases; while CNN+RNN algorithms have some visible but not efficient improvement in three by three cases. The difference

might result from the ways the relationships are represented: The possibility of direct representation of the progressive row relationships of multiple patterns can have representational advantages over pairs relationships when there is more than one pair in a row. The advantage of CNN+RNN algorithms is also coherent with previous studies: CNN+RNN algorithms have been used and proved to be successful in abstracting continues relationships [25]. The general deficiency of the algorithms in solving three by three Raven's Matrices problems is expected given the complicity of the relationships and limitation in training (or experiences).

The discrepancy between algorithms with different complexity does share some parallel with the structures that are responsible for different functions of fluid intelligence in the brain: Different brain areas are identified to be responsible for 'matching', 'visual reasoning', and 'analytic reasoning' in an fMRI study using figures of Raven's Matrices [48]. According to this study, it should make more sense to view the algorithms in the current study as different functions which serve different purposes in the process of problem-solving instead of separate individuals that are independent in abilities and make separate judgments. The cooperation between different functions (brain areas) is very important in high-level cognitive functions such as intelligence or consciousness [11]. It is hard to conclude, at this stage, the correspondent between different algorithms and different brain areas or layer of information processes. It might be possible, however, that we find some parallel in the structure where these functions are implemented in different types of agents after we have a clearer understanding of the underlying functions.

8.2 Comparative Results

For the properties summarised in figure A.1. More comparative analyses are conducted to better understand relevant processes. The problems that require to identify two relationships to be solved are considered to be harder for all algorithms (37.5 % against 79.6 %, or algorithms only: 18.8 % against 52.3 %). The relationship involving 'missing' or property 'shape and part' is also intangible for all algorithms. The accuracy of 'shape and part' representation improves a lot with the string representation method (27.3 % to 63.6 %) while the 'missing' relationship remains hard for algorithms (only one out of eight problems are solved by the algorithms and another one is solved by the string representation). These contrasts provide useful information about how the algorithms represent relationships and how the new representation

validly improves the performance of the algorithms.

The algorithms are not as efficient in abstracting multiple relationships. This defect might be a direct consequence of the error comparison method we used. The sum of differences in the crucial layers regardless of their weights in deciding different relationships precludes the possibility of isolating relationships in the final judgments. Instead of separately identify co-varying relationships, this error comparison method likely aims to deal with the high dimensional relationship comparison all at once. Isolating relationships and making separate comparisons might be beneficial for simplifying the problems. Integrated training between algorithms with automatic attention might enable such separation [65], but the training is highly unlikely to be realized with limited training data (see section limitations and future direction).

Inefficient integrity and part representation is the main reason that three by three cases are not solved by the feed-forward neural network. When complete representations are presented to the algorithms as strings, a lot more cases are solved. This confirms the ability of the feed-forward neural network in abstracting relationships [35] [70]. The huge improvement draws attention to the need for better intrinsic representation for neural networks, especially when facing a novel environment with limited resources. This representational barrier restricts the ability of algorithms to be involved in discovery or creativity related novel tasks [15] [3].

The algorithms (together) can constantly solve some of the Raven's Matrices problems. If we assume there are some intrinsic general intelligence behind the behavioral consequences, the intelligence level of the combined algorithms can be measured and compare with human results. The intelligence level of the algorithms (indicated by the total number of problems they maximally solved) reaches some average intelligence level of human children. According to the 'Smoothed British Norms' [54] of Children (1979) the algorithms (CNN, CNN+FNN, RNN, and CNN+RNN) reaches the average (50 percentile) performance of 8-year-old children (25 problems), noticeable, the CNN+FNN algorithm alone reaches the level of an average 6.5-year-old children (17 problems), while the algorithms and string representation methods (CNN, CNN+FNN, RNN, CNN+RNN, and string representation) altogether, show a performance level of eleven to 12-year-old children (41 problems) [54]. This result is quite promising in its way of approaching how the 'intelligent' conclusions are generated by human species. It can not be generalized and concluded that the algorithms 'think' the same way as children. However, it opens possibilities for intelligent solutions for novel situations or problems to be generated by artificial algorithms automatically in a visible future.

8.3 Methodology and Potential Implementations

The error comparison method used in the experiments was selected by a set of pre-experiments, it is adapted from predictive coding, where it is used to enhance the divergence and provoke more efficient processing in the next level [52]. The error propagation methodology is adapted here as an error comparison method that use the distinction in the crucial layer itself to make decisions. The method opens new possibilities in training and testing. It opens new training *scopes*. The overlapping divergence rather than predictive features are weighted using this method. The error comparison method engenders multiple, distributed relationships to yield direct contrast so the information can be analyzed or compared beyond the facial level. It enable more divergent *forms*. The method allows variation in representation between candidate answers and training answers. In this case, the training answers are different kinds of (pair-wise) relationships (Categorical), and the selection of candidate answer requires to contrast the information of how multiple relationships (in training answers) are embedded to different degrees in the frames (Distributional). The method retains maximally amount of information in overall relationship similarity, the analyzation would be inefficient otherwise. The efficiency of the method is proved by the positive result across the chapters. And it does yield highly promising result given well-structured representations (see result chapter 6).

The method is highly demanded in the circumstance with restricted training sets which are repeatedly presented. Since the training phase is isolated with the testing phase the character of previous training answers can by no mean have any impact on the final result. The efficiency of the method and the resemblance it has with the human visual system is already studied in the context of predictive coding [26]. It can be possible that the same process how the algorithms learn from error is realized in the brain itself [42]. The implementation of the method is still basic in this project. Fully understand the biological level of error propagation in the brain and take full advantage of the methodology can be expected to be a possible and valid way for to improve the efficiency of training, solve less well-defined problems, and achieve more human-like performances.

8.4 Calculation Salient Matrices

Calculation Salient Matrices is an additional testing method constructed in the experiment. It provides a direct comparison between algorithms with different structures. Well defined relationships and generation rules make it possible to generate an unlimited amount of train examples and testing incidents with no overlapping. The algorithm oriented test is designed to measure a general cognitive ability that is required for relationship abstraction. This ability enables the algorithm to explore a novel environment voluntarily. The automation is highly desirable feeding the needs of dealing with the ever-changing environments in our daily life [44]. The validity of the test is partially established in this experiment. Measuring the convergence of the current test and Turing test should provide more convincing conceptual validation. And if more divergent algorithm structures are available, comprehensive comparison and Norms can be established. Ceiling performance Solving Calculation Salient Matrices does not mean that the algorithm is 'intelligent', the matrices contain basic sequences that are ordinary for human. The relationship abstraction ability measured by this test is transferable to different domains and it converges with the common intelligence measurement of Raven's Matrices.

The results of the test indicate that the structure of an algorithm can induce constraints to its abilities in abstracting different kinds of relationships. This is consistent with the integrated information theory [66]. None of the relationships reached ceiling performance at the current stage. There is still room for further improvement. Researches in domain neuroscience might help to fill in the gap and analogically improve the performance of algorithms [30] while the study of computational methodologies might help to take full advantage of the capacity of algorithms [19].

8.5 Limitation and Future Direction

Raven's Matrices is the *intelligent problem-set* that we used in the current project. The sixty-item original version was developed by Raven in 1936. Co-validation, Simplification, and other progresses took place over the years [53] [54]. The test is proved to be decisive and distinctive [54], however, it did not capture the full scale of this promising ability of human even at that time [22]. It can be questionable whether it is ever possible to put a ceiling for what human can do with their brightest intelligence. Nevertheless, the incapability the algorithms shows in solving the Raven's Matrices

can be dramatically more discouraging than it presents. The parasitic relationship between artificial intelligence and human species had existed for decades, it is interesting then, what state will the new developments take us. Maybe someday will come that we can rely on artificial intelligence on their performances and judgments, and then we will be able to find out what form these intelligence algorithms take.

For the *abilities* that the algorithms have already shown, there is still a long way to go before they can be implemented in relevant tasks. As mentioned in the above sections, the algorithms have their own merits and defects in solving different kinds of problems, it is sometimes indistinguishable, however, which answer or algorithm to stick to. Admittedly, the Correct Distinction Rate does provide useful information about the confidence level of the algorithm (Pearson correlation 0.6069 for different method solving the three by three cases). The discrimination is not definite. The algorithms yield high confidence for wrong answers constantly in many different cases. Further improvement and integration of algorithms are needed before the algorithm with advertised relationship abstraction ability in a novel environment to be able to play its role in tasks.

A hybrid level *integration of different algorithms* can be realized through automated regulation. The weights of different algorithms in which can only be gained from the previous experiments, or training in an integrated algorithm. The former would need a comprehensive literature review of similar context. It would save efforts in training, however, some result might not be transferable facing new contexts. The latter method can be promising and automatic in the way it learns automatically the calculations, attentions, and regulations cooperatively, while massive training data would be needed. A training data generator which can generate similar training examples basing on limited resources might be one valid direction of improvement (see [62]). However, the more automatic the system can be, the harder it is to control and analyze the function of different components. At that stage, the improvement in intelligent problem-solving abilities and emergence of consciousness can be very hard to be manifested. And it would be harder to make analogies to human intelligence with less control. It is still interesting if we can develop a constrained method approximating the desired state constantly till the point a computational goal can be reached. Or is that the cognitive abilities of human are as mysterious as they can be, existing in a system-level where any attempts starting from partials approximating are doomed to fail, while the approximation in function, similarity in underlying mechanisms might only happen in a global level beyond the sum of its parts, and beyond conscious understanding [33].

Chapter 9

Conclusions

It is possible to use neural networks to solve intelligence problems in some novel environments. Especially when they are organized in an efficient way enhancing their advantages. The knowledge of the merits and defects of the algorithms could provide useful information in *pre-selection* of algorithms and answer confirmation. Effective *training* with maximum contrast information can be beneficial in expectation generation. Clever and direct *answer selection* method provide the possibility of overcoming the format difference in training and testing phases, making generalization easier. Effective *representation* embedding associations might supersede the need for heavy attention tuning and provide an accurate result. The progress made by the algorithms is still elementary, artificial and insufficient. We do believe structures of algorithms constrains its abilities in problem-solving or relationship abstraction. More experiments are needed for a more efficient algorithm (structure) to be built.

The experiment is part of the trend of experiments from Psychology, Neuroscience, and Cognitive Science attempting to study, mimic and understand the mystery of human consciousness and intelligence. In this experiment, the knowledge of previous stages and experiences is communicated between trials and algorithms though kernel weights and connection weights. The process enables the algorithm to be ready to perform similar operations to brand new stimuli and situations. But the efficiency of communication is still limited in form and content and the effectiveness of performance is highly restricted. We believe the goal of understanding human intelligence and build intelligent algorithms is still out of reach at the moment, but it is not impossible in a theoretical or methodological way.

Bibliography

[1] Mohammad Ali Ahmadi. Prediction of asphaltene precipitation using artificial neural network optimized by imperialist competitive algorithm. *Journal of Petroleum Exploration and Production Technology*, 1(2-4):99–106, 2011.

[2] Bernard J Baars. *A cognitive theory of consciousness*. Cambridge University Press, 1993.

[3] John Baer. *Creativity and divergent thinking: A task-specific approach*. Psychology Press, 2014.

[4] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[5] ME Bitterman. The evolution of intelligence. *Scientific American*, 212(1):92–101, 1965.

[6] Patricia Bowers. Hypnosis and creativity: The search for the missing link. *Journal of Abnormal Psychology*, 88(5):564, 1979.

[7] bradware. ravens-progressive-matrices. https://github.com/bradware/ravens-progressive-matrices.git. published three years ago.

[8] Emre Cakır, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6):1291–1303, 2017.

[9] James Cannady. Artificial neural networks for misuse detection. In *National information systems security conference*, volume 26. Baltimore, 1998.

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

[11] Rodney MJ Cotterill. Cooperation of the basal ganglia, cerebellum, sensory cerebrum and hippocampus: possible implications for cognition, consciousness, intelligence and creativity. *Progress in Neurobiology*, 64(1):1–33, 2001.

[12] Daniela Danciu. A cnn based approach for solving a hyperbolic pde arising from a system of conservation laws-the case of the overhead crane. In *International Work-Conference on Artificial Neural Networks*, pages 365–374. Springer, 2013.

[13] Ronna F Dillon, John T Pohlmann, and David F Lohman. A factor analysis of raven's advanced progressive matrices freed of difficulty factors. *Educational and Psychological Measurement*, 41(4):1295–1302, 1981.

[14] Gerald M Edelman. *The remembered present: a biological theory of consciousness.* Basic Books, 1989.

[15] Andreas Fink, Roland H Grabner, Mathias Benedek, Gernot Reishofer, Verena Hauswirth, Maria Fally, Christa Neuper, Franz Ebner, and Aljoscha C Neubauer. The creative brain: Investigation of brain activity during creative problem solving by means of eeg and fmri. *Human brain mapping*, 30(3):734–748, 2009.

[16] Robert M French. The turing test: the first 50 years. *Trends in cognitive sciences*, 4(3):115–122, 2000.

[17] Douglas J Futuyma. *Science on trial: the case for evolution*. Pantheon Books, 1983.

[18] David C Geary. *The origin of mind*. Washington, DC: American Psychological Association, 2005.

[19] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[20] Peter D Grünwald. *The minimum description length principle*. MIT press, 2007.

[21] Umut Güçlü and Marcel AJ van Gerven. Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream. *Journal of Neuroscience*, 35(27):10005–10014, 2015.

[22] Julia C Hall. Correlation of a modified form of raven's progressive matrices (1938) with the wechsler adult intelligence scale. *Journal of Consulting Psychology*, 21(1):23, 1957.

[23] Michael E Hasselmo, Eric Schnell, and Edi Barkai. Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region ca3. *Journal of Neuroscience*, 15(7):5249–5262, 1995.

[24] W-D Heiss. Pet in coma and in vegetative state. *European journal of neurology*, 19(2):207–211, 2012.

[25] Felix Hill, Adam Santoro, David GT Barrett, Ari S Morcos, and Timothy Lillicrap. Learning to make analogies by contrasting abstract relational structure. *arXiv preprint arXiv:1902.00120*, 2019.

[26] Yanping Huang and Rajesh PN Rao. Predictive coding. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(5):580–593, 2011.

[27] Christos C Ioannou. Swarm intelligence in fish? the difficulty in demonstrating distributed and self-organised collective intelligence in (some) animal groups. *Behavioural processes*, 141:141–151, 2017.

[28] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5308–5317, 2016.

[29] Lisa N Jefferies, Richard D Wright, and Vincent Di Lollo. Inhibition of return to an occluded object depends on expectation. *Journal of Experimental Psychology: Human Perception and Performance*, 31(6):1224, 2005.

[30] Rex Eugene Jung, Brittany S Mead, Jessica Carrasco, and Ranee A Flores. The structure of creative cognition in the human brain. *Frontiers in human neuroscience*, 7:330, 2013.

[31] Hiroaki Kitano and Tetsuya Higuchi. High performance memory-based translation on ixm2 massively parallel associative memory processor. In *AAAI*, pages 149–154, 1991.

[32] Masaki Kobayashi. Attractors accompanied with a training pattern of multivalued hopfield neural networks. *IEEJ Transactions on Electrical and Electronic Engineering*, 10(2):195–200, 2015.

[33] Wolfgang Köhler. Gestalt psychology. *Psychological research*, 31(1):XVIII–XXX, 1967.

[34] Maithilee Kunda, Keith McGreggor, and Ashok Goel. Taking a look (literally!) at the ravens intelligence test: Two visual solution strategies. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 32, 2010.

[35] Ernst Kussul and Tatiana Baidyk. Improved method of handwritten digit recognition tested on mnist database. *Image and Vision Computing*, 22(12):971–981, 2004.

[36] Michael Lewis and Harry McGurk. The evaluation of infant intelligence: Infant intelligence scores–true or false? *ETS Research Bulletin Series*, 1972(2):i–15, 1972.

[37] Xiaoxiao Li and Chen Change Loy. Video object segmentation with joint re-identification and attention-aware mask propagation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 90–105, 2018.

[38] Alfred J Lotka. Contribution to the energetics of evolution. *Proceedings of the National academy of Sciences of the United States of America*, 8(6):147, 1922.

[39] William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016.

[40] Alianna J Maren, Craig T Harston, and Robert M Pap. *Handbook of neural computing applications*. Academic Press, 2014.

[41] Henry Markram. The blue brain project. *Nature Reviews Neuroscience*, 7(2):153, 2006.

[42] William J Matthews, Devin B Terhune, Hedderik Van Rijn, David M Eagleman, Marc A Sommer, and Warren H Meck. Subjective duration as a signature of coding efficiency: emerging links among stimulus repetition, predictive coding, and cortical gaba levels. *Timing & Time Perception Reviews*, 1, 2014.

[43] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[44] CL Miller. Where do we go from here with automation systems? *AUTOMATION SYSTEMS FOR HIGHWAY ORGANIZATIONS*, page 125, 1972.

[45] Xiao-Xiao Niu and Ching Y Suen. A novel hybrid cnn–svm classifier for recognizing handwritten digits. *Pattern Recognition*, 45(4):1318–1325, 2012.

[46] Adrian M Owen, Nicholas D Schiff, and Steven Laureys. The assessment of conscious awareness in the vegetative state. In *The Neurology of Conciousness*, pages 155–166. Elsevier, 2016.

[47] Se Rim Park and Jinwon Lee. A fully convolutional neural network for speech enhancement. *arXiv preprint arXiv:1609.07132*, 2016.

[48] Vivek Prabhakaran, Jennifer AL Smith, John E Desmond, Gary H Glover, and John DE Gabrieli. Neural substrates of fluid reasoning: an fmri study of neocortical activation during performance of the raven's progressive matrices test. *Cognitive psychology*, 33(1):43–63, 1997.

[49] prshrestha. Ravens-progressive-matrices. https://github.com/prshrestha/Ravens-Progressive-Matrices.git.

[50] rafiyajaved. Raven's matrices problems. https://www.academia.edu/35518613/Raven$_{s}Progressive_{M}atrices_{I}Q_{T}est?email_{w}ork_{c}ard = view - paper.$

[51] rafiyajaved. Ravensprogressivematrices. https://github.com/rafiyajaved/RavensProgressiveMatri

[52] Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79, 1999.

[53] John Raven. The raven's progressive matrices: change and stability over culture and time. *Cognitive psychology*, 41(1):1–48, 2000.

[54] John Raven and Jean Raven. *Uses and abuses of intelligence: Studies advancing Spearman and Raven's quest for non-arbitrary metrics*. Royal Fireworks Press, 2008.

[55] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[56] Michael Rutter. *Genes and behavior: Nature-nurture interplay explained.* Blackwell Publishing, 2006.

[57] Sudipto Saha and GPS Raghava. Prediction of continuous b-cell epitopes in an antigen using recurrent neural network. *Proteins: Structure, Function, and Bioinformatics*, 65(1):40–48, 2006.

[58] Ayse Pinar Saygin, Ilyas Cicekli, and Varol Akman. Turing test: 50 years later. *Minds and machines*, 10(4):463–518, 2000.

[59] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[60] Thomas Serre, Lior Wolf, and Tomaso Poggio. Object recognition with features inspired by visual cortex. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE DEPT OF BRAIN AND COGNITIVE SCIENCES, 2006.

[61] Donald F Specht. Probabilistic neural networks for classification, mapping, or associative memory. In *IEEE international conference on neural networks*, volume 1, pages 525–532, 1988.

[62] Donald F Specht. A general regression neural network. *IEEE transactions on neural networks*, 2(6):568–576, 1991.

[63] Christopher Summerfield and Tobias Egner. Expectation (and attention) in visual cognition. *Trends in cognitive sciences*, 13(9):403–409, 2009.

[64] Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. Introduction to multilayer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1):43–62, 1997.

[65] Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. Deep semantic role labeling with self-attention. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[66] Giulio Tononi, Melanie Boly, Marcello Massimini, and Christof Koch. Integrated information theory: from consciousness to its physical substrate. *Nature Reviews Neuroscience*, 17(7):450, 2016.

[67] Anne M Treisman and Garry Gelade. A feature-integration theory of attention. *Cognitive psychology*, 12(1):97–136, 1980.

[68] Max Wertheimer. Über gestalttheorie. *Symposion*, 1925.

[69] David Wood, Jerome S Bruner, and Gail Ross. The role of tutoring in problem solving. *Journal of child psychology and psychiatry*, 17(2):89–100, 1976.

[70] Wenzu Wu, Kunjin Chen, Ying Qiao, and Zongxiang Lu. Probabilistic short-term wind power forecasting based on deep neural networks. In *2016 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, pages 1–8. IEEE, 2016.

[71] Seoung Wug Oh, Joon-Young Lee, Kalyan Sunkavalli, and Seon Joo Kim. Fast video object segmentation by reference-guided mask propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7376–7385, 2018.

[72] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

[73] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[74] Welly Zhang. Raven: A dataset for relational and analogical visual reasoning. https://github.com/WellyZhang/RAVEN.git.

Appendix A

Additional Figures

A.1 Overview

Figures with the note ‘see appendix’ will appear in the following pages. The figures presented in this section contains useful information that are independent to the main result but enable comprehensive understanding of the experiment and the results, such as overview of the problems and performance (figure A.1), details of experimental materials (two figures for Chapter 2), Correct Distinction Rate of the correct answer (Chapter 3 to 6) and other experiment results that are not presented in relevant chapters due to the page limits.

Question	Type	Relationship	Property	Minimum Definitive Relationships	CNN	RNN	CNN+FNN	CNN+RNN	representation
1	missing part	match	pattern	1	√	√	√	-	-
2	missing part	match	pattern	1	√		√	-	-
3	missing part	match	pattern	1	√	√		-	-
4	missing part	match	pattern	1	√		√	-	-
5	missing part	match	pattern	1	√		√	-	-
6	missing part	match	pattern	1	√		√	-	-
7	missing part	same	pattern	1	√	√		-	-
8	missing part	progression	pattern	1				-	-
9	missing part	progression	pattern	1		√		-	-
10	missing part	progression	pattern	1				-	-
11	missing part	progression	pattern	1				-	-
12	missing part	progression	pattern	2				-	-
13	two by two	match	shape	1	-	-	√	√	-
14	two by two	match	shape	1	-	-	√	√	-
15	two by two	relationship_same	direction	1	-	-	√		-
16	two by two	relationship_progression	direction	1	-	-			-
17	two by two	relationship_progression	direction	1	-	-			-
18	two by two	relationship_add	direction and pattern	1	-	-			-
19	two by two	relationship_add	direction and pattern	1	-	-			-
20	two by two	relationship_same	shape/shape and pattern	1	-	-	√		-
21	two by two	relationship_add	shape and pattern	2	-	-			-
22	two by two	relationship_same divergence	shape and part	1	-	-	√	√	-
23	two by two	relationship_same divergence	shape and part	1	-	-	√		-
24	two by two	relationship_same divergence	shape/shape and part	1	-	-	√		-
25	three by three	relationship_same/progression	size/size and shape	1	-	-			√
26	three by three	relationship_progression/progression	number	1	-	-	√		√
27	three by three	relationship_progression/progression	part and number	2	-	-			√
28	three by three	relationship_progression/progression	number	2	-	-			√
29	three by three	relationship_progression/progression	size and shape and uniform expectation	1	-	-		√	√
30	three by three	relationship_progression/progression	possion and part	1	-	-			√
31	three by three	relationship_progression/progression	pattern and size	1	-	-			√
32	three by three	relationship_progression/progression	shape and possition	2	-	-		√	√
33	three by three	relationship_progression/same divergence	possition	1	-	-			√
34	three by three	relationship_progression/same divergence	part and place	1	-	-			√
35	three by three	relationship_same divergence/same divergence	pattern and size and place	1	-	-	√		√
36	three by three	relationship_same divergence/same divergence	shape	2	-	-		√	√
37	three by three	relationshape_same/order	shape	1	-	-	√		√
38	three by three	relationship_missing/missing	shape (integreted)	1	-	-		√	
39	three by three	relationship_missing/missing	shape and part	1	-	-			
40	three by three	relationship_same divergence/same divergence	shape and part	1	-	-			√
41	three by three	relationship_same divergence/same divergence	shape and part	1	-	-			√
42	three by three	relationship_same divergence/same divergence	shape and part	2	-	-			
43	three by three	relationship_same divergence/same divergence	shape and pattern	2	-	-		√	
44	three by three	relationship_missing missing/missing missing	shape and part and pattern	2	-	-			
45	three by three	relationship_missing missing/missing missing	shape and part	2	-	-			
46	three by three	relationship_missing missing/missing missing	curvature and shape	2	-	-			
47	three by three	relationship_missing missing/missing missing	shape and number	2	-	-			
48	three by three	relationship_missing missing/missing missing	shape and part	2	-	-			
49	three by three	relationship_add/add	shape and part	1	-	-			√
50	three by three	relationship_add/add	shape and part	1	-	-			√
51	three by three	relationship_add/add	shape and part	1	-	-			√
52	three by three	relationshio_minus/minus	shape and part	1	-	-			√
53	three by three	relationshio_minus/minus	shape and part	1	-	-			
54	three by three	relationshio_minus/minus	shape and part	1	-	-	√	√	
55	three by three	relationshio_add and minus/add and minus	shape and part	2	-	-			
56	three by three	relationshio_add and minus/add and minus	shape and part	2	-	-			
57	three by three	relationship_same and missing/ same and missing	shape and part	2	-	-			√
58	three by three	relationship union/union	shape	1	-	-			√
59	three by three	relationship union/union	shape and part	1	-	-		√	√
60	three by three	relationship_numeric	number	1	-	-		√	
sum					7	4	16	>11	22
method sum								(algorithms only) 26	41

Figure A.1: This figure shows the overall result of all the representative algorithms and representational methods. The checkmarks specify the trials which the average Correct Distinction Rate is lowest for the correct answer; and the '-' mark means that the method is not feasible for the problems. The Relationships specifies the relationship type: match (same pattern), same (same pattern combination), progression (continues increasing or decreasing in the same dimension), missing (permutation of three properties), add or minus (combination or separation of different parts), union (identify the same parts), numeric (count and algebra), and combinations of the relationships. Property column documents property that shows the relationship. Minimum Definitive Relationships refers to the minimum number of relationships the algorithms need to

A.2 Chapter 2

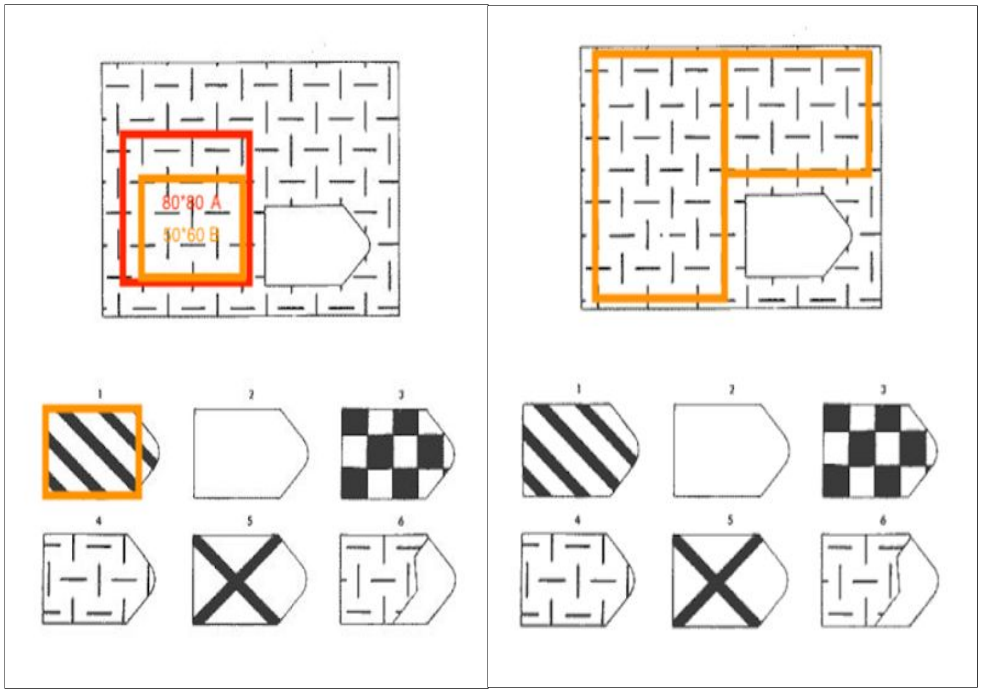

Figure A.2: This picture shows an example of how the problems are presented to the algorithm we designed. The picture contains the third integrated Raven's Matrices problem (the correct answer is answer '4'). In experiment one (chapter 2), the training and testing frames are constructed from the pictures above. The answers are cut into rectangular shaped numpy matrices (60 by 50) just like the orange rectangular at the answer '1'. The red square (80 by 80) is used in integrated training. Error comparison is made contrasting the output of the second convolutional layer and the Orange square within the integrated training range. training range for RNN algorithms in chapter 5 and the random attention attempt mentioned in the discussion and appendix expends the error comparison region to the regions within the orange squares in the picture in the right-hand-side (Left rectangular only for RNN algorithm, the question picture above the blank is constructed as part of the answers).

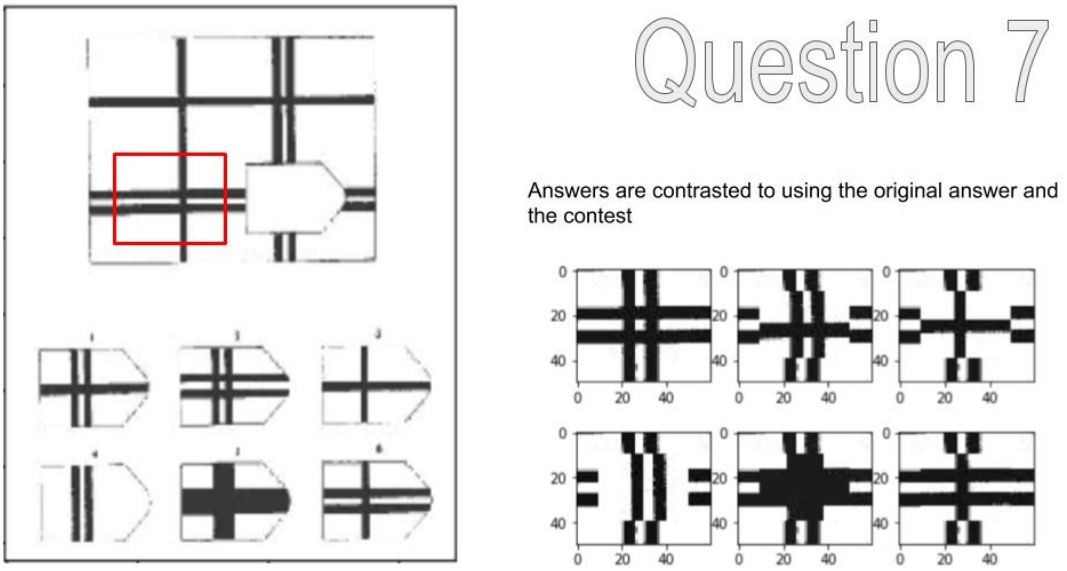

Figure A.3: This picture shows an example of how integrated Raven's Matrices problems are presented to the algorithm we designed to make the incongruous more understandable. The problem in the figure is the eighth problem (problem number seven counting from zero, correct answer: '2'). The answers are constructed with a five-pixel boundary of the correct answer (which can be viewed as the extension of the question), as shown in the right-hand-side of the figure.

A.3 Chapter 3

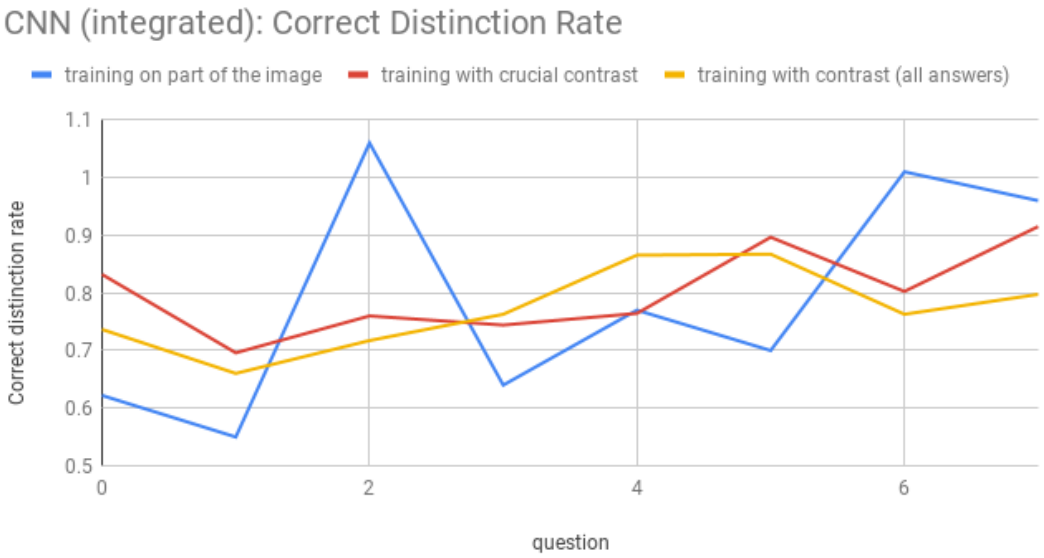

Figure A.4: This picture shows the Correct Distinction Rate of the correct answer to the tree training methods for the first seven problems (chapter 3). As can be seen from the figure, the fluctuation is smaller for crucial and all contrast training. This is consistent with their relatively stable performance. Training on contrasting methods yields a lot higher CDR given the problems that it is less likely to solve (problem two and six). Note: problem number one contains only one pattern, and thus the distinction of crucial training and all contrast training is less salient.

A.4 Chapter 4

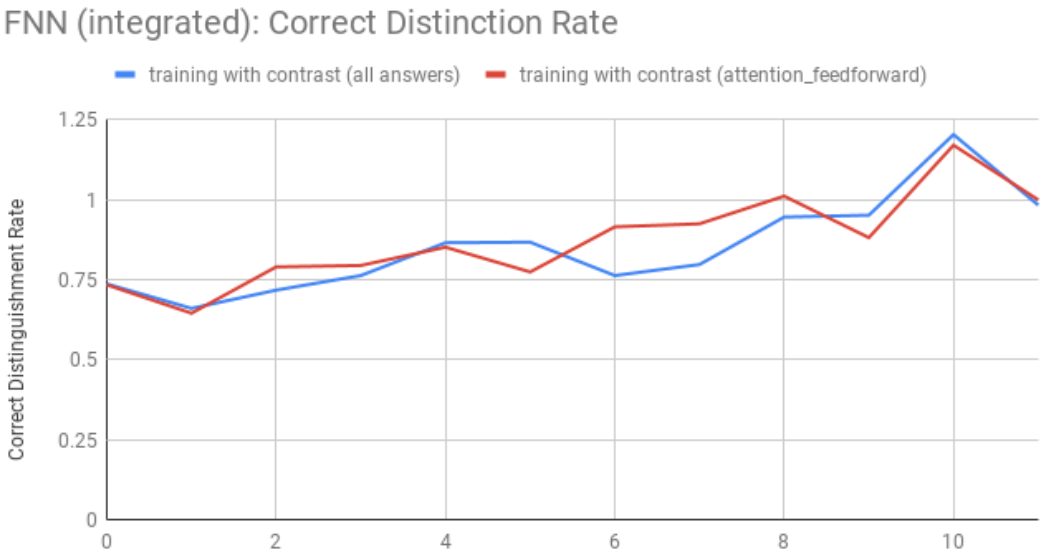

Figure A.5: The figure shows the Correct Distinction Rate of CNN+FNN algorithms solving the first 12 problems, comparing with the result of all contrast method used by CNN algorithms The Correct Distinction Rate is very similar for the two methods. Neither of them can solve problem seven to twelve with acceptable consistency. And the CNN algorithm performs slightly better in the sixth problem.

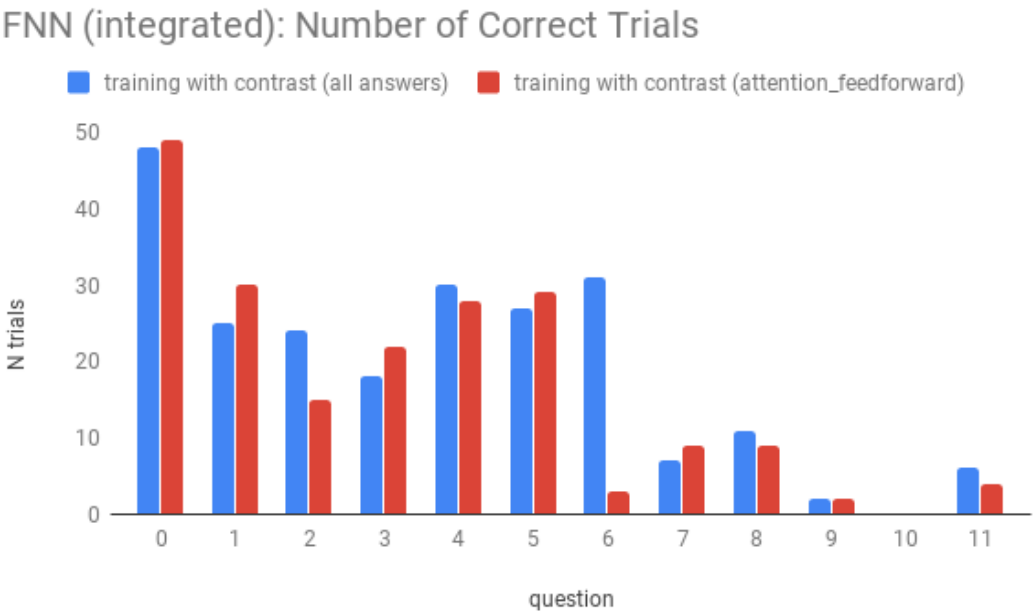

Figure A.6: The figure shows the number of trials out of fifty for CNN+FNN algorithms solving the first 12 problems, comparing with the result of all contrast method used by CNN algorithms. Generally speaking, the problems are not solved more often by the CNN+FNN algorithms. They do show a limited increase in hard problem like eight or eleven, but the difference is neglectable and the accuracy is still unacceptable.

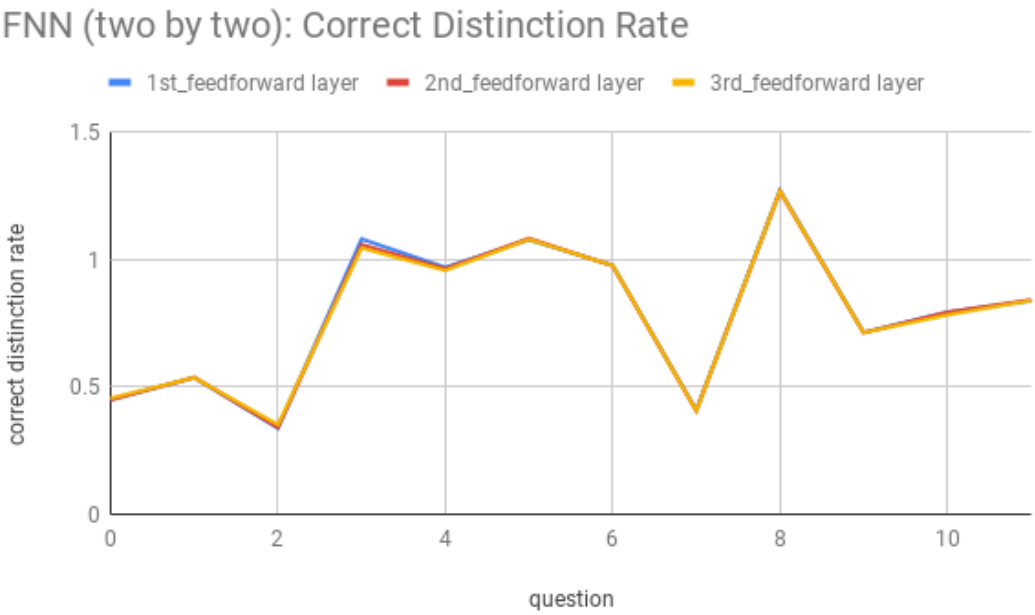

Figure A.7: This picture shows the Correct Distinction Rate of the two by two Raven's Matrices problems (problem 12 to 24, chapter 4.2), contrasting the three layers of the feed-forward neural network. The CDR tend to be higher for the trials that they did not solve (problem three to six and eight). And it is nearly identical for all three layers. It might be tempting to conclude that, at least for the relationships that are identified by the neural networks, they tend to be abstracted at a very early stage (in the first feed-forward layer). See the discussion section of this chapter for details.

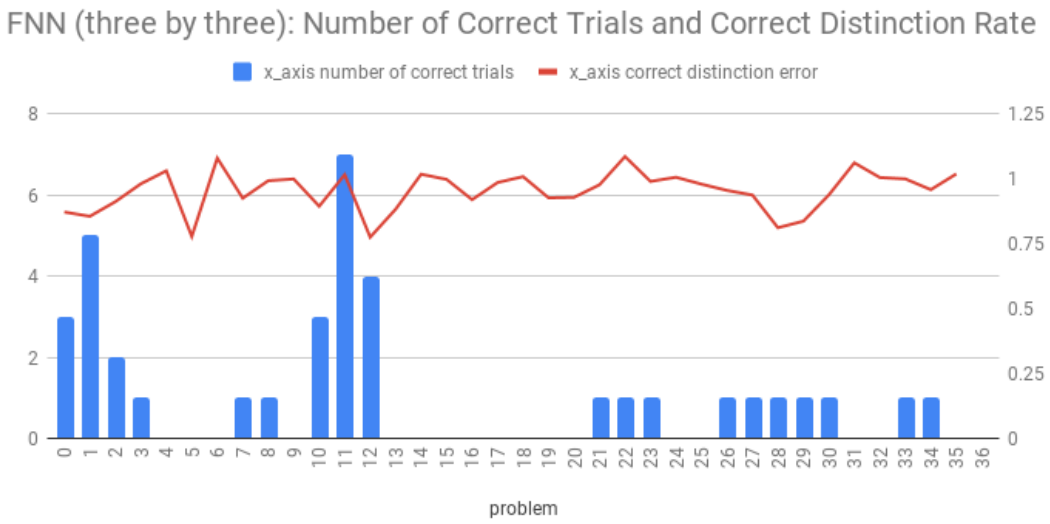

Figure A.8: This figure shows the number of correct trials (out of 5) and Correct Distinction Rate produce by the CNN+FNN algorithm in Chapter 4.3 focusing on row relationships in three by three cases. The calculation is very slow and thus only five trials are carried out for each problem. As can be seen from the figure the accuracy is quite low after the first few problems. No follow-up experiment with more trials or method adjustment was carried out given the unpromising result.

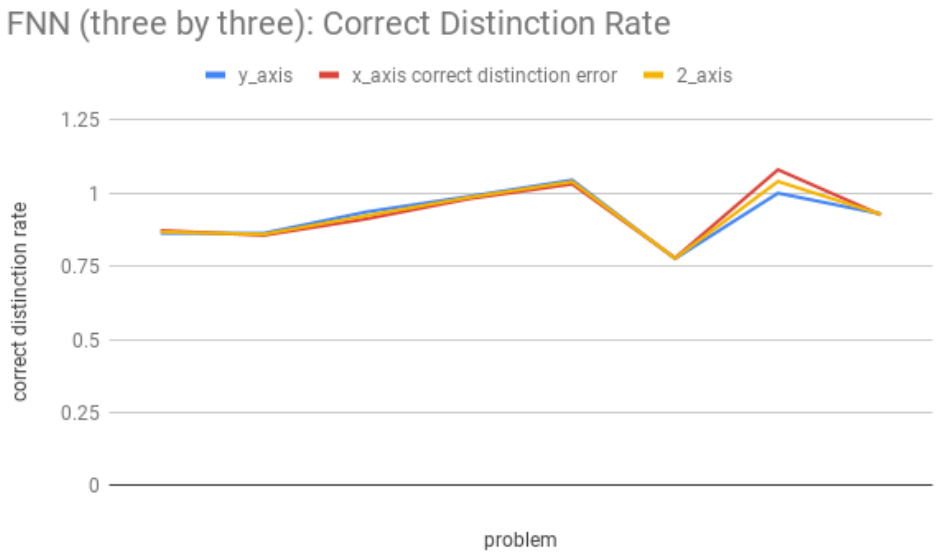

Figure A.9: This figure shows the Correct Distinction Rate of the first few three by three problems in Chapter 4.3, comparing the result of training on x_axis (row relationships), y_axis (column relationships), and the average result of x_axis and y_axis. The additional axis did not improve the judgments. The unpromising result might result from the fact that the overall correct judgments are very limited. So it can be even harder for the two axes to yield accurate judgment at the same time without distinguishing against their result. Due to the unpromising result, the two axes comparison method is not applied to three by three problems with even harder relationships.

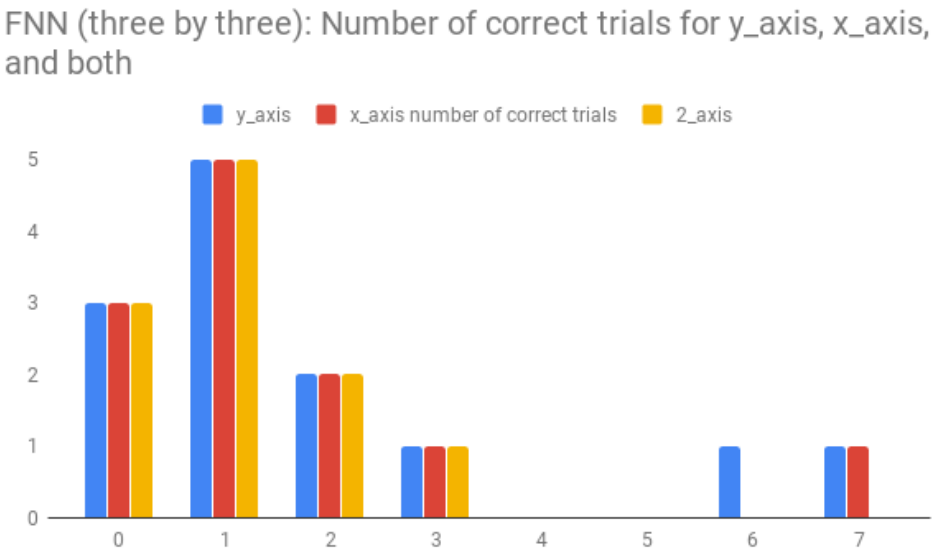

Figure A.10: This figure shows the number of correct trials of the first few three by three problems in Chapter 4.3, comparing the result of training on x_axis (row relationships), y_axis (column relationships), and the average result of x_axis and y_axis. The overall correct trials drop very quickly after the third problem. The remaining problems are only solved occasionally by one of the axes. The integration of the result produced by different axes is inefficient and it might only be able to yield good results if the condition is similarly acceptable for the axes. The similarity between axes indicates that the algorithm did not represent the column relationship and row relationship efficiently. And this might be the reason why the problems are not solved by CNN+FNN algorithm.

A.5 Chapter 5

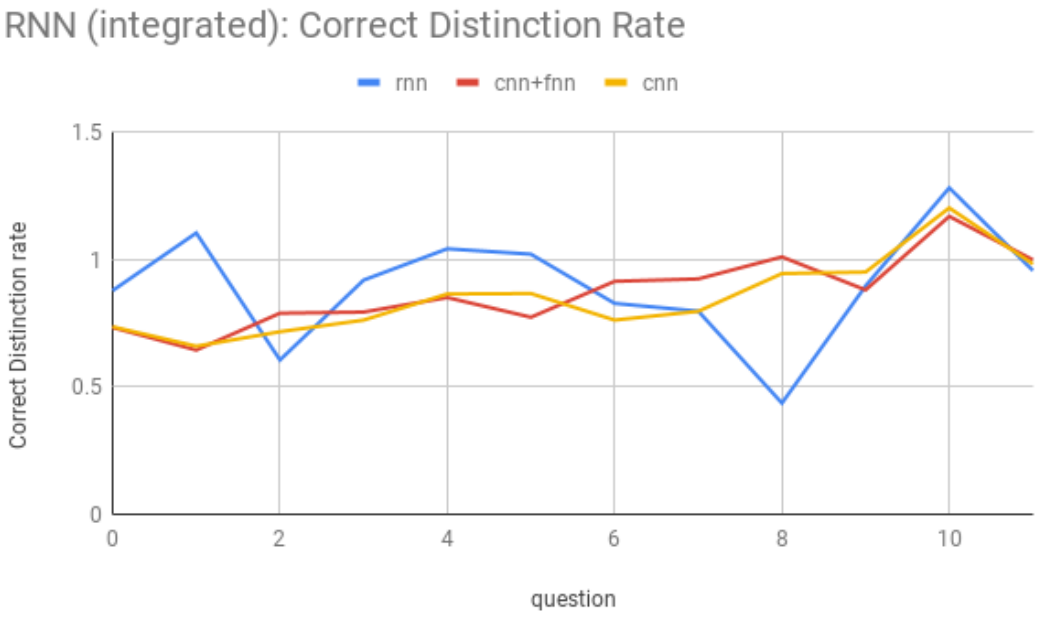

Figure A.11: This picture shows the Correct Distinction Rate of the integrated case problems, contrasting the convolutional neural network, convolutional neural network and feed-forward neural network and recurrent neural network (chapter 5.1). As can be seen from the figure, the recurrent neural network did not yield lower Correct Distinction Rate in the first six problems comparing to CNN or CNN+FNN algorithms. It does show improvement, however, in problem six to twelve, especially given the cases that are concretely solved by RNN algorithms.

Figure A.12: This figure shows the Correct Distinction Rate of the two by two problems in Chapter 5.2, contrasting the convolutional neural network with feed-forward neural networks and convolutional neural networks with recurrent neural networks. The CNN+RNN algorithm did not provide addition improvement comparing to CNN+FNN algorithm. This might result from the low recurrent time-steps are not enough for the recurrent neural network to fully show its capability.

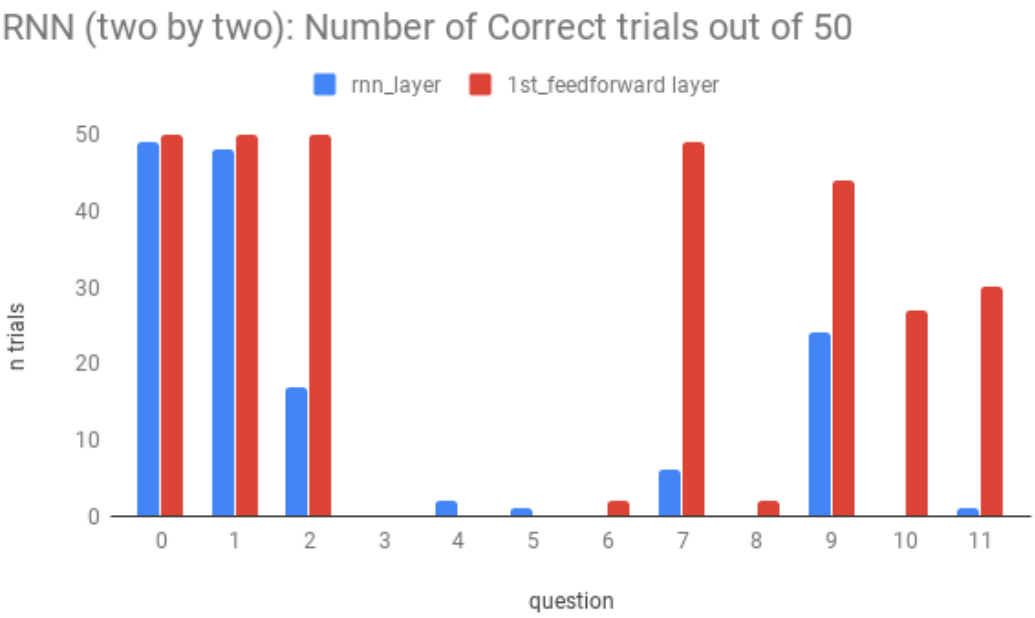

Figure A.13: This figure shows the number of correct trials of the two by two problems in Chapter 5.2, contrasting the convolutional neural network with feed-forward neural network and convolutional neural network with recurrent neural networks. The performance of CNN+ RNN algorithm drops dramatically after the first two problems. CNN+RNN algorithm did not reach the performance level of CNN+FNN, especially in more complicated cases.

A.6 Chapter 6

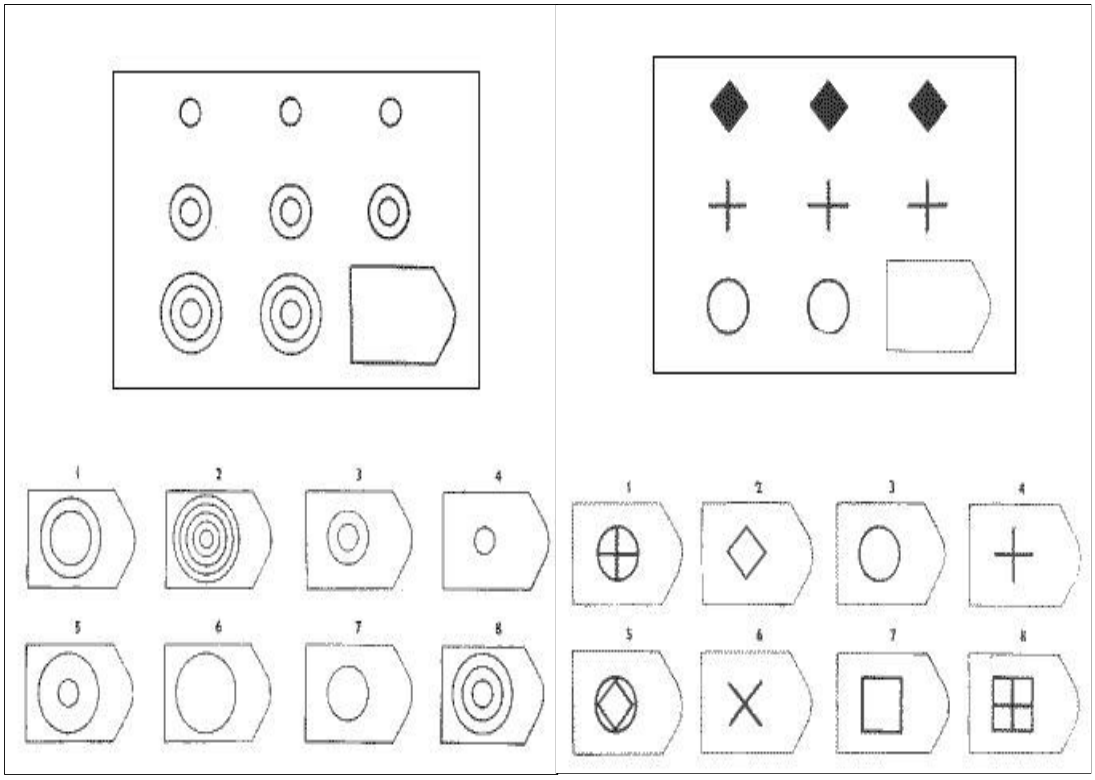

Figure A.14: This figure shows the first (left, problem number 25, answer 8) and thirteenth (right, problem number 37, answer 3) three by three Raven's Matrices problems. The former is not solved be the CNN+FNN algorithm will the latter is considered to be solved. This means that the inefficiency in representing the integrity or the relationship between integrity and parts can impact the capability of the feed-forward neural network in abstracting relationships.

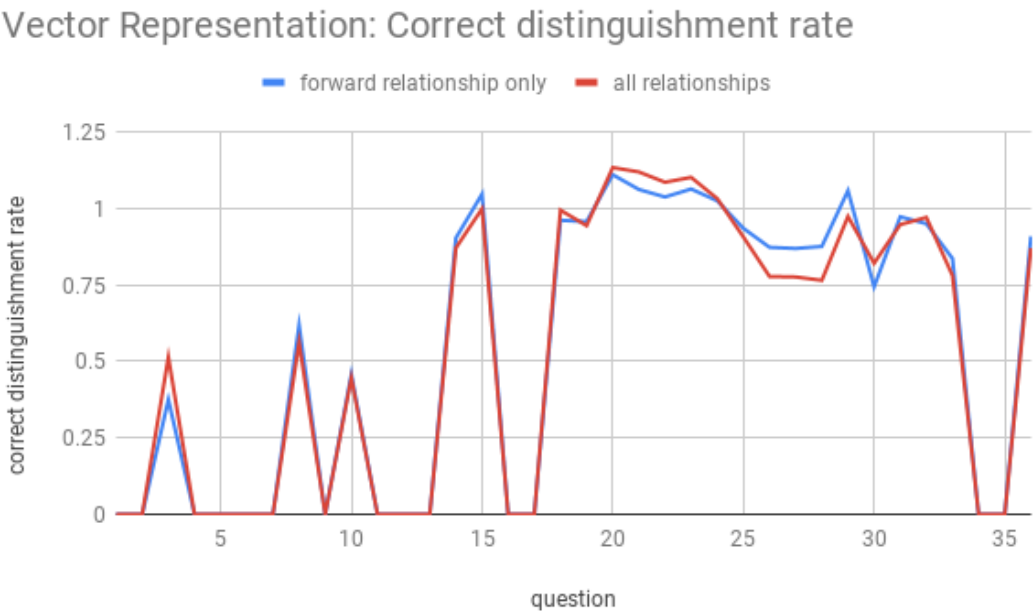

Figure A.15: This picture shows the Correct Distinction Rate of the three by three case problems using vector representation and feed-forward neural network, contrasting the result including only the progressive relationships and all relationships (chapter 6). As can be seen in the figure, the Correct Distinction Rate is typically low for the majority of problems. however, it is relatively high and indistinguishable for problem eighteen to problem 24. Those problems are not solved by the new representation in both comparisons.

A.7 Chapter 7

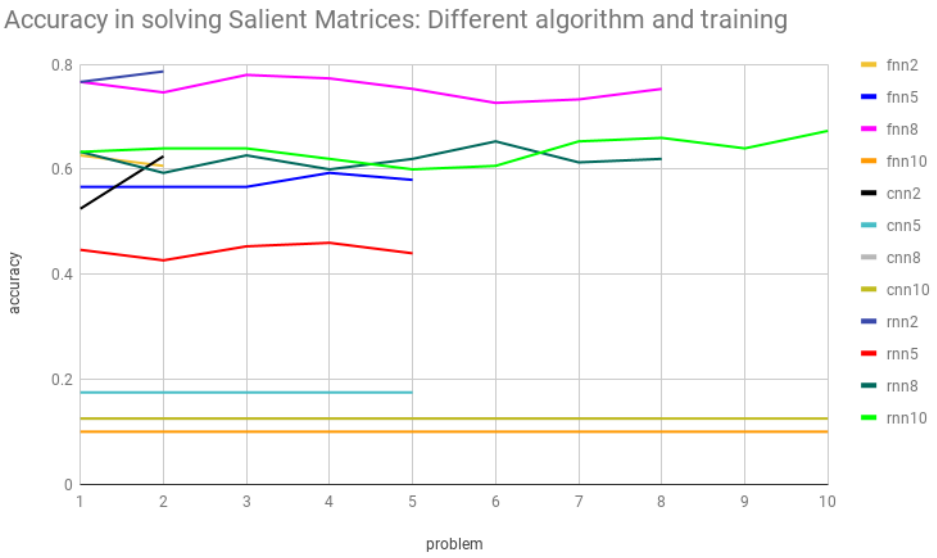

Figure A.16: This figure shows the performance of all training method and algorithms solving the ten Calculation Salient matrices (chapter 7). The name tags are generated with the name of the algorithm and the number of problems it is trained and test with. The result suggests that: The performance is stable above chance level for FNN till the eighth problem, RNN till the tenth problem and CNN till the second problem. The best performance for the first two problems is achieved by RNN training to the second problem, the best performance for the second to the eighth problem is achieved by FNN training to the eighth problem. The best performance of problem eight to ten is achieved by RNN training to the tenth problem. This figure also includes a reliability test across dimensions (see fnn8 and fnn8_2d, the data used to train the latter is two dimensional). The data generated as one dimension (180 inputs * 128 batch size) and two dimensions (18*10 inputs *128 batch size) achieve very similar accuracy level in solving the first eight problems.

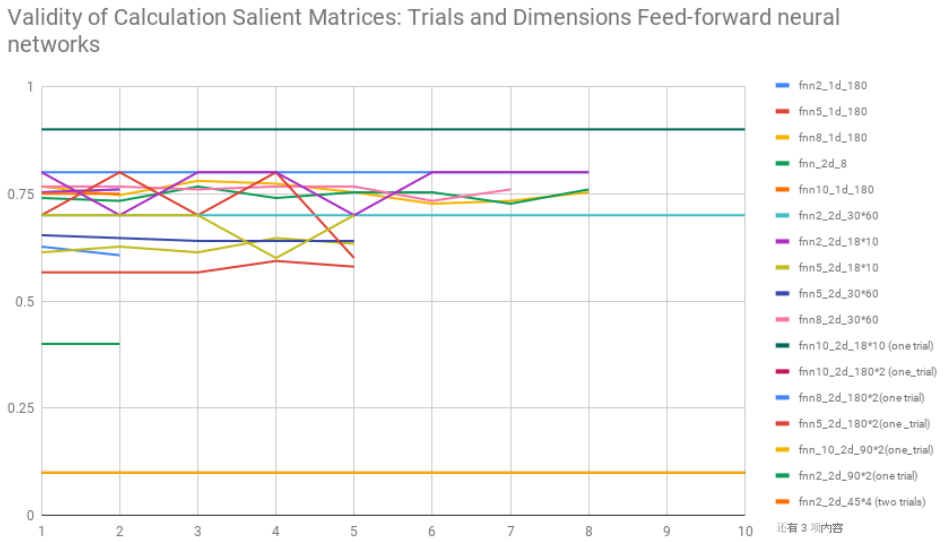

Figure A.17: The figure shows the viability test trials carried by feed-forward neural networks for Calculation Salient Matrices between different problems and dimensions. Fifteen trials are calculated for each parameter set unless otherwise indicated. The name of the trials are constructed with algorithm name, the number of problems trained and tested slash dimension (1d or 2d) slash data shape. As can be seen in the figure, the results show consistency between trials and dimensions. The accuracy of the problems clustered in the range of 0.6 to 0.8 with occasional disturbance. Problem size has an impact on the algorithms' performance. Problem number nine and ten can be solved at some accuracy level with more balanced height and width. This might need to be controlled to achieve a better comparison between algorithms.

Appendix B

Pre-experiments and additional Attempts

B.1 Overview

Pre-experiment (two pre-experiment, related to Chapter 2 and Chapter 6) and other additional attempts (Chapter 3 to Chapter 4) are documented in this section (description of the experiment, method, and result figures). Some of those are mentioned briefly in the discussion or introduction sections as justifications of methods or additional experiment attempts. These experiments are not directly related to the main results in the thesis, but it supports the choice of methods, provides additional validation of the main results, and shine diagnose insights in providing or rejecting solutions for existing problems.

B.2 Pre-experiments

Pre-experiments are carried out with the three by three case problems. The capabilities of algorithms are quite limited without the error comparison method. The training is limited in two row-relationships aginst different permutations. and the troublesome representation makes it hard to generalized between cases. The ability of Convolutional neural network with feed-forward neural networks is limited to the basic cases such as problem thirteen.

Another trial run was carried out using a mask-rnn algorithm. A lot of noise-free training samples are generated to train it. The position, shape, and size can be automatically captured by the algorithm. However, the specific patterns in the problem set can

be hard to be generalized in other cases and the hollow patterns that are overlapping are tricky for the algorithm for exact position isolation. None of the noisy original patterns can be recognized by the tuned algorithm. And a lot of artificial processes can not be automatically realized in every stage.

B.3 Additional Attempts

Chapter 3: An additional trial run was tested with the idea of random attention in the original picture and answers. In the main result, the error comparison is made with a fixed part of the original problem image (See figure A.2). This is replaced by a random selection process from the top part of the image above the missing part or the left part of the image (See figure A.2). And answer attention is variated by dividing the fifty trials into the top half, bottom half, left half, right half of the candidate answer and the other half of the real pattern and another 10 trial for the original central construction like figure A.3. This divided attention was added for the intention to enable the algorithm to generate a more comprehensive sense of view of the problem and not be biased for nonuniform information which is not transferable in general. The method is proved to be unsuccessful. The result is shown in Figure B.1 B.2. These results indicate that the algorithms do nor rely on continuity expectations to solve the problems. A comprehensive overall match function seems to be more explainable in the given cases.

Chapter 3: There is another attempt in Experiment one targeting at solving problems seven to twelve. We tried to emphasize shape continuity by emphasizing the boundaries in of the first convolutional layer. In this case, the normalized Correct Distinction Rate of the first convolutional layer instead of the output layer is computed. In addition to the normalized Correct Distinction Rate as mentioned in section 2.1, normalized Correct Distinction Rate of edges is computed separately. All contrast training method is used to full-fill the intention of achieving better performance. The method is first tested on the 6 simple cases in the integrated case Raven's Matrices. It turns out that emphasizing the edges does not improve the performance of the neural network. three out of six cases are solved by the mean overall Correct Distinction Rate or mean Correct Distinction Rate of edges alone. Only two out of seven cases are solved integrating the two error rate. The method has not proceeded to the harder problems. Other details of the experiment (Correct Distinction Rate and number of correct trials) are shown in Figure B.3 and B.4.

Chapter 4: Escape connection which pass the original output of the convolutional

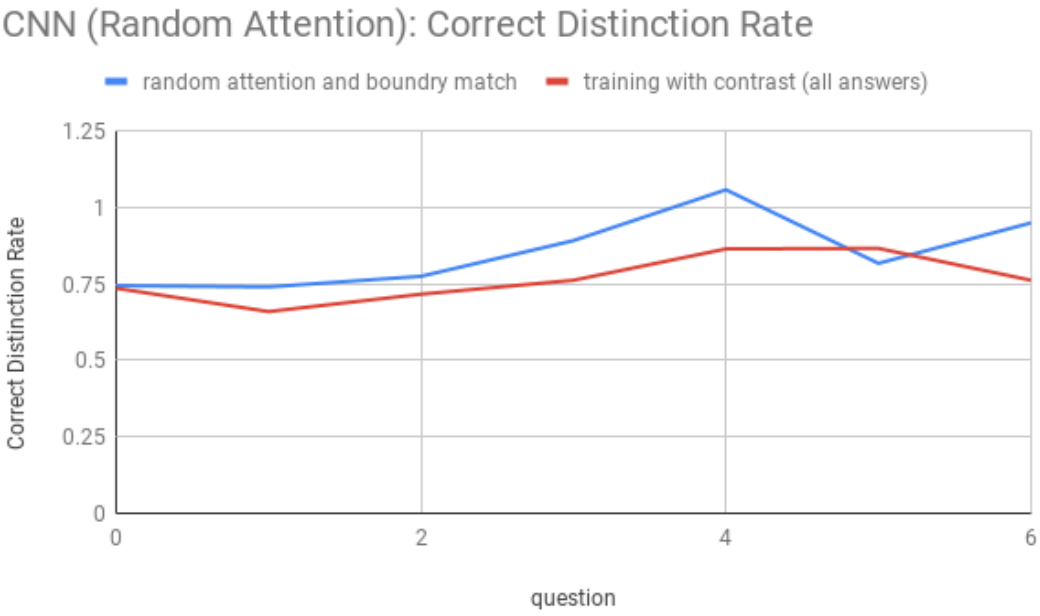

Figure B.1: This figure shows the Correct Distinction Rate of the first 6 problems, comparing the all contrast method and the random attention method. The experiment is included briefly in the discussion and section B.3 in the appendix. Random attention in questions and answers are used to emphasize the binderies and discontinues at the boundaries in the answer. This method turns out to be rather confusing for the algorithm. Since the algorithms might not be relying on the continuity to make judgments, but rather a matching function. Including part of the correct patterns will increase the difficulties of the problems.

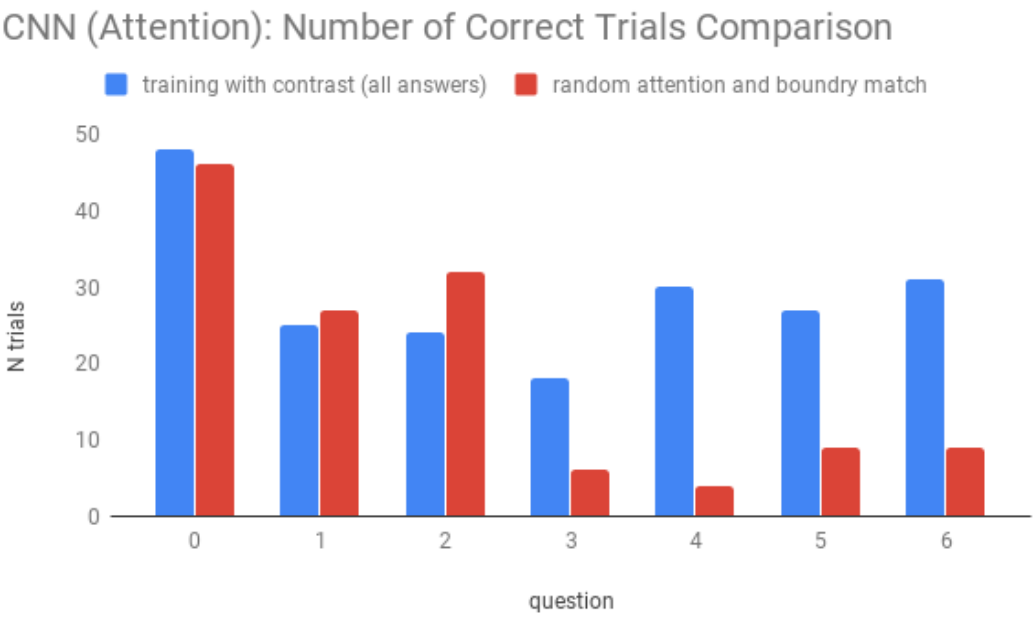

Figure B.2: This figure shows the number of correct trials of the first 6 problems, comparing the all contrast method and the random attention method. The results suggest that the problems are not being solved more often by the new methods emphasizing different boundaries, the number of correct trials drops dramatically after the third problem.

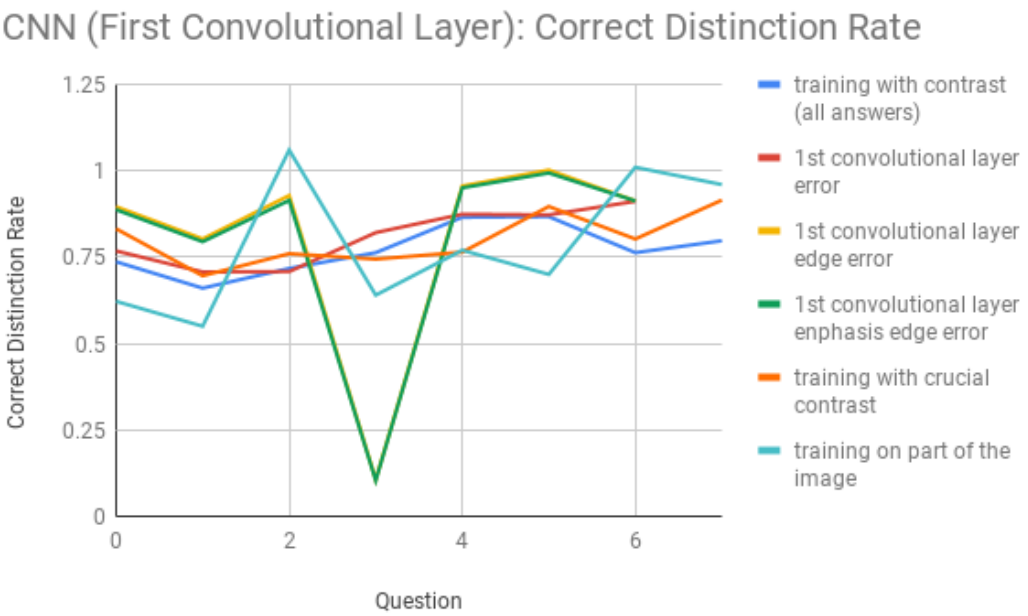

Figure B.3: This picture shows the Correct Distinction Rate of the first seven problems, including the additional edge emphasizing method in the discussion. The results suggest that the first convolutional layer is already efficient in solving some of the integrated problems. The algorithm yield very good performance in the third and fourth problem. But the efficiency did not last in the more complicated cases.

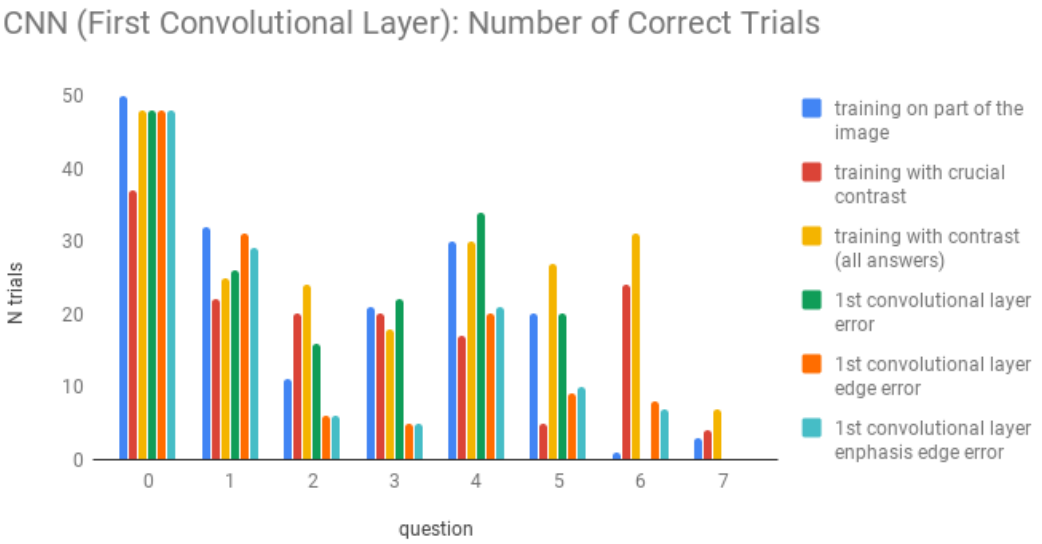

Figure B.4: This picture shows the number of correct trials of the first seven problems, including the additional edge emphasizing method in the discussion. Emphasizing the edges do not provide any improvement in any problem. Rather, it can be confusing in many cases. This result indicates that the algorithm did not tend to focus on edges as human does in making judgments. They do not understand how to use the discontinuance in the edges to identify the concept of fit. They are not expected to be able to understand slightly distorted images to be different than the original ones.

layer along with the output of the previous feed-forward layer to the next feed-forward layer, is applied as an attempt to maintain the original information and induce possible incubation that enable the algorithm to escape from the original wrong attractor integrating the new information from the feed-forward layer, the result is presented in figure B.5 and B.6.

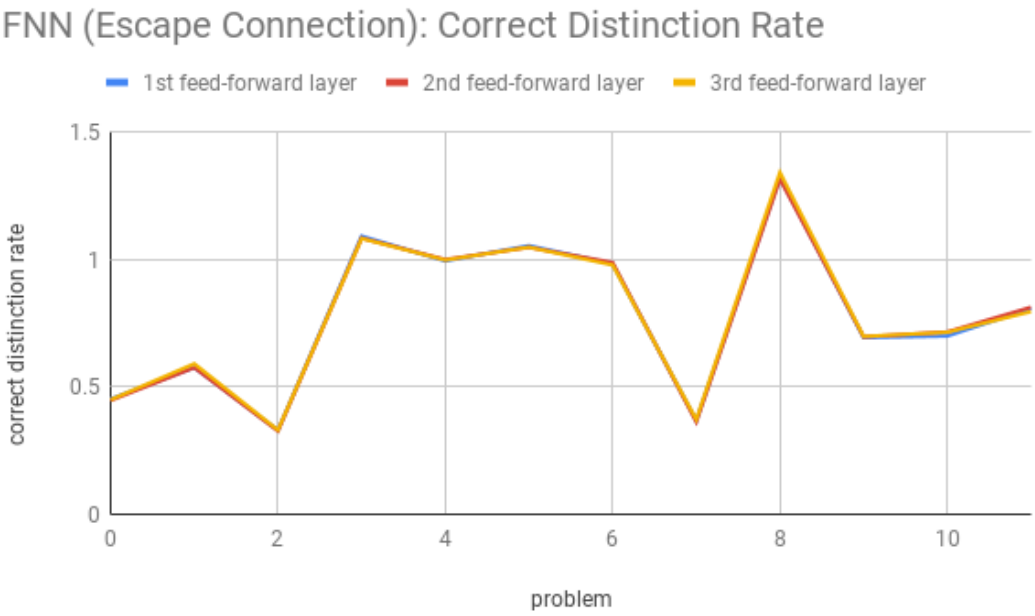

Figure B.5: This picture shows the Correct Distinction Rate of the twelve two by two problems. The results suggest that: escape connection failed to provide new information and increase the performance of the algorithms. The performances of the three layers are still similar to each other. This might mean that the algorithm can only abstract a subset of relations and it happens very fast in the first feed-forward layer.

Chapter 4: An additional experiment is carried out for the Recurrent neural network used in Chapter 4.1 (chapter 4). In this experiment, the recurrent time-steps is adjusted to be 40 (length of the answer) and the candidate patterns are presented to the algorithm in the training phases as part of the image. This attempt aims to make the experiment more similar to the experiment described in Chapter 3 where the cases are successfully solved by the convolutional neural network. The result can be seen in

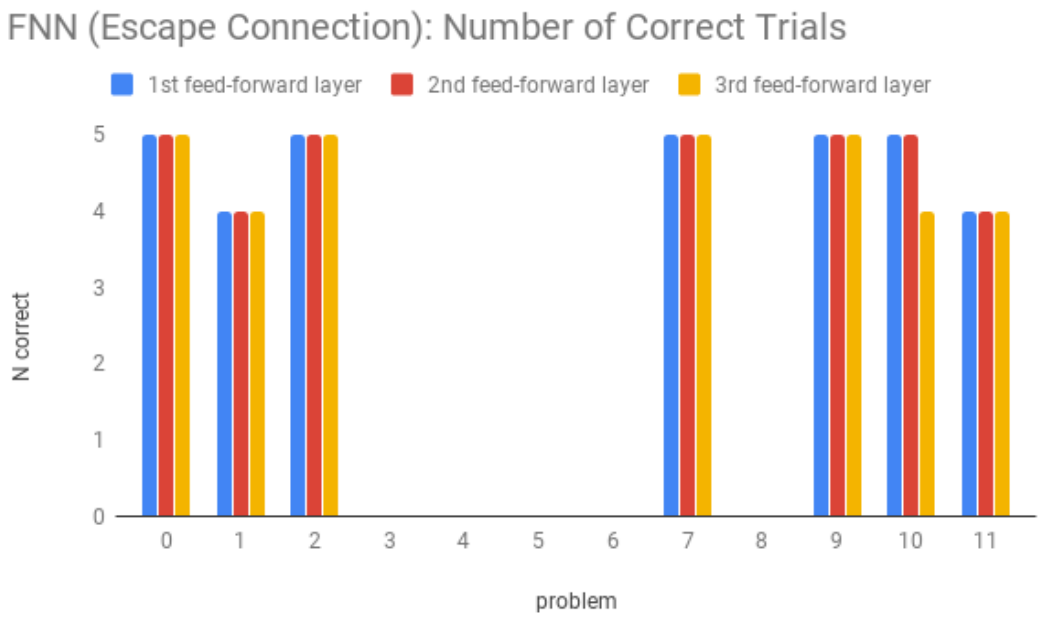

Figure B.6: This picture shows the number of correct trials of the twelve two by two problems (mentioned in the discussion of chapter 3, experiment 2.2), As can be seen from the figure, some problems are concretely solved by all layers of the feed-forward neural networks. This co-validated the result in chapter 4.2. However, the escape connection didn't enable the algorithm to solve more trials or problems, the attempt is considered to be unsuccessful.

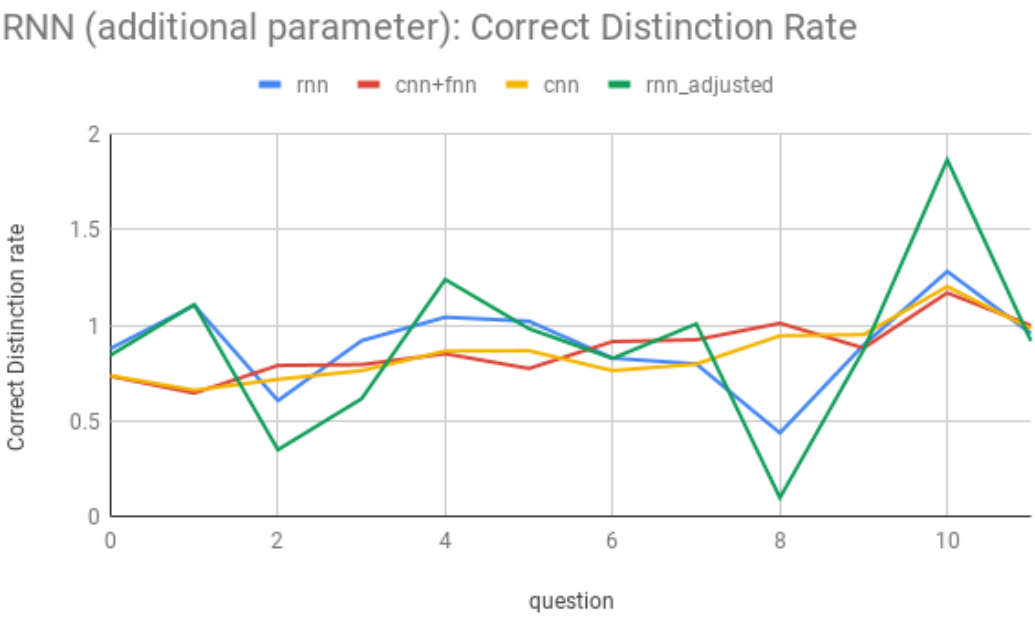

Figure B.7: This picture shows the Correct Distinction Rate of the twelve integrated problems (mentioned in Chapter 4.1 discussion and appendix). The results suggest that: the Correct Distinction Rate bounced around more dramatically in this example. The integrated answer instead of two separate ones make it impossible to smooth the output with an average function. But as a confident judgment, the new version carries out more meaningful information comparing to the original one used in Chapter 4.1.

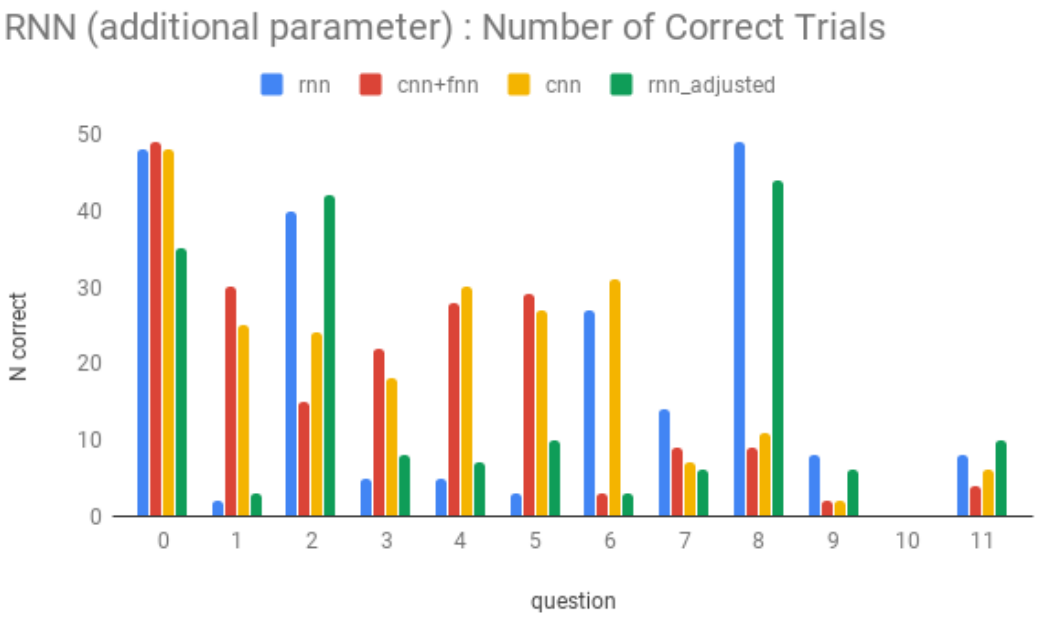

Figure B.8: This picture shows the number of correct trials of the twelve integrated problems (mentioned in Chapter 4.1 discussion and appendix). The exposures to the candidate answer did not enable the algorithm to solve more problems. The result is coherent with the result in Chapter 4.1. This experiment helps to conclude that it is the less meaningful representation instead of the experience of candidate answer that is responsible for the defect of RNN neural network in integrated problems.

Appendix C

Algorithms

C.1 Overview

The algorithms used in the previous sections are documented here. The figure below shows the Calculation Salient Matrices used in Chapter 7. The implementation of well-established algorithms is not directly illustrated here. They can be found in Github dictionary: https://github.com/qynglang/Algorithm-intelligence.git

All algorithms are programmed in python implementing with Tensor-flow toolbox. Some of the pre-experiments are carried out on the Google cloud platform, other experiments are carried out locally in Linux environment. The hyper-parameters are set referencing previous experiments. Because the error comparison method is nontraditional, validation set is only used in Chapter 7's experiment, where the step with the highest validation accuracy in 3000 steps is selected. In other experiment, where the training examples are very limited, 100% training accuracy is required instead. The 'over-fitting' is not expected to affect the performance of the algorithms to any degree because the comparisons are made excluding the soft-max layers.

```
def generator1(shape):
#match
    D=len(shape)
    if D==1:
        data=np.ones((shape[0]))
        answer=np.zeros((1,10))
        answer[:,0]=1
    if D==2:
        data=np.ones((shape[0],shape[1]))
        answer=np.zeros((1,10))
        answer[:,0]=1
    if D==3:
        data=np.ones((shape[0],shape[1],shape[2]))
        answer=np.zeros((1,10))
        answer[:,0]=1
    return data, answer

def generator2(shape):
#shape
    D=len(shape)
    if D==1:
        data=np.ones((shape[0]))
        d=np.full((int(shape[0]/2)+1,2),
[1,2]).reshape(int(2*(int(shape[0]/2)+1)))
        data[0:shape[0]]=d[0:shape[0]]
        answer=np.zeros((1,10))
        answer[:,1]=1
    if D==2:
        data=np.ones((shape[0],shape[1]))
        d0=np.full((int(shape[0]/2)+1,2),
[1,2]).reshape(int(2*(int(shape[0]/2)+1)),1)
        d1=np.full((int(shape[0]/2)+1,2),
[2,1]).reshape(int(2*(int(shape[0]/2)+1)),1)

        data[0:shape[0],::2]=d0[0:shape[0]]
        data[0:shape[0],1::2]=d1[0:shape[0]]
        answer=np.zeros((1,10))
        answer[:,1]=1
    if D==3:
        data=np.ones((shape[0],shape[1],shape[2]))
        d0=np.full((int(shape[0]/2)+1,2),
[1,2]).reshape(int(2*(int(shape[0]/2)+1)),1)
        d1=np.full((int(shape[0]/2)+1,2),
[2,1]).reshape(int(2*(int(shape[0]/2)+1)),1)
        for i in range (0,shape[2]):
            if i%2==1:
                data[0:shape[0],::2,i]=d0[0:shape[0]]

    d1=np.full((int(shape[0]/3)+3,3),
[1,2,3]).reshape(int(3*(int(shape[0]/3)+3)))
    if D==1:
        data=np.zeros((shape[0]))
        data[0:shape[0]]=d[0:shape[0]]
        for i in range (0,shape[0]):
            data[i]+=2*int(i/6)
        answer=np.zeros((1,10))
        answer[:,5]=1
    if D==2:
        data=np.zeros((shape[0],shape[1]))

data[0:shape[0],:]=d1[0:shape[0]].reshape(shape[0],1
)
        answer=np.zeros((1,10))
        answer[:,5]=1
    if D==3:
        data=np.zeros((shape[0],shape[1],shape[2]))

data[0:shape[0],:,:]=d1[0:shape[0]].reshape(shape[0],
1,1)
        answer=np.zeros((1,10))
        answer[:,5]=1
    return data, answer

def generator7(shape):
#progression and same divergence
    D=len(shape)
    #d=np.arange(shape[0])
    #d0=np.full((int(shape[0]/9)+3,9),
[1,2,3,2,3,4,3,4,5]).reshape(int(9*(int(shape[0]/9)+3)))
    d1=np.full((int(shape[0]/3)+3,3),
[1,2,3]).reshape(int(3*(int(shape[0]/3)+3)))
    if D==1:
        data=np.arange((shape[0]))+1
        for i in range (0,shape[0]):
            data[i]=data[i]-int(i/3)
        #data[0:int(shape[0]/2)]=d[0:shape[0]]
        answer=np.zeros((1,10))
        answer[:,6]=1
    if D==2:
        data=np.zeros((shape[0],shape[1]))
        for i in range (0,shape[1]):

data[0:shape[0],i]=np.arange((shape[0]))+(i%3)
        answer=np.zeros((1,10))
        answer[:,6]=1
            data[0:shape[0],1::2,i]=d1[0:shape[0]]
        else:
            data[0:shape[0],::2,i]=d1[0:shape[0]]
            data[0:shape[0],1::2,i]=d0[0:shape[0]]
    answer=np.zeros((1,10))
    answer[:,1]=1
    return data, answer

def generator3(shape):
#progression
    D=len(shape)
    if D==1:
        data=np.arange(shape[0])
        #data[:,0:shape[0]]=d[:,0:shape[0]]
        answer=np.zeros((1,10))
        answer[:,2]=1
    if D==2:
        data=np.zeros((shape[0],shape[1]))
        for i in range (0,shape[1]):
            data[:,i]=np.arange(shape[0])+i
        answer=np.zeros((1,10))
        answer[:,2]=1
    if D==3:
        data=np.zeros((shape[0],shape[1],shape[2]))
        for i in range (0,shape[1]):
            for j in range (0,shape[2]):
                data[:,i,j]=np.arange(shape[0])+i+j
        answer=np.zeros((1,10))
        answer[:,2]=1
    return data, answer

def generator4(shape):
#progression_reverse
    D=len(shape)
    if D==1:
data=(np.rot90(np.rot90(np.arange(shape[0]).reshap
e(shape[0],1)))).reshape(shape[0])
        #data[:,0:shape[0]]=d[:,0:shape[0]]
        answer=np.zeros((1,10))
        answer[:,3]=1
    if D==2:
        data=np.zeros((shape[0],shape[1]))
        for i in range (0,shape[1]):
data[:,i]=(np.rot90(np.rot90(np.arange(shape[0]).resh
ape(shape[0],1)))+shape[1]-i).reshape(shape[0])
        answer=np.zeros((1,10))
        answer[:,3]=1

    if D==3:
        data=np.zeros((shape[0],shape[1],shape[2]))
        for i in range (0,shape[1]):
            for j in range (0,shape[2]):

data[0:shape[0],i,j]=np.arange((shape[0]))+(i%3)+(j%
3)
        answer=np.zeros((1,10))
        answer[:,6]=1
    return data, answer

def generator8(shape):
#progression and missing
    D=len(shape)
    d=np.full((int(shape[0]/9)+3,9),
[1,2,3,2,3,1,3,1,2]).reshape(int(9*(int(shape[0]/9)+3)))
    if D==1:
        data=np.zeros((shape[0]))
        data[0:shape[0]]=d[0:shape[0]]
        for i in range (0,shape[0]):
            data[i]=data[i]+int(i/9)

        answer=np.zeros((1,10))
        answer[:,7]=1
    if D==2:
        data=np.zeros((shape[0],shape[1]))
        for i in range (0,shape[1]):
            data[0:shape[0],i]=d[0:shape[0]]+i
        answer=np.zeros((1,10))
        answer[:,7]=1
    if D==3:
        data=np.zeros((shape[0],shape[1],shape[2]))
        for i in range (0,shape[1]):
            for j in range (0,shape[2]):
                data[0:shape[0],i,j]=d[0:shape[0]]+i+j
        answer=np.zeros((1,10))
        answer[:,7]=1
    return data, answer

def generator9(shape):
#add
    D=len(shape)
    if D==1:
        data=np.zeros(shape[0])
        data[0:2]=1
        for i in range (2,shape[0]):
            data[i]=data[i-1]+data[i-2]
    if D==3:
        data=np.zeros((shape[0],shape[1],shape[2]))
        for i in range (0,shape[1]):
            for j in range (0,shape[2]):
data[:,i,j]=(np.rot90(np.rot90(np.arange(shape[0]).res
hape(shape[0],1)))+shape[1]-i+shape[2]-j).reshape(s
hape[0])
        answer=np.zeros((1,10))
        answer[:,3]=1
    data=data-np.min(data)
    return data, answer

def generator5(shape):
#missing
    D=len(shape)
    d=np.full((int(shape[0]/9)+3,9),
[1,2,3,2,3,1,3,1,2]).reshape(int(9*(int(shape[0]/9)+3)))
    if D==1:
        data=np.zeros((shape[0]))
        data[0:shape[0]]=d[0:shape[0]]
        answer=np.zeros((1,10))
        answer[:,4]=1
    if D==2:
        data=np.zeros((shape[0],shape[1]))
        for i in range (0,9):

data[0:shape[0],i::9]=d[i:shape[0]+i].reshape(shape[0
],1)
        answer=np.zeros((1,10))
        answer[:,4]=1
    if D==3:
        data=np.zeros((shape[0],shape[1],shape[2]))
        for i in range (0,9):
            for j in range (0,9):

data[0:shape[0],i::9,j::9]=d[i+j:shape[0]+i+j].reshape(s
hape[0],1,1)
        answer=np.zeros((1,10))
        answer[:,4]=1
    return data, answer

def generator6(shape):
#same and progression
    D=len(shape)
    d=np.full((int(shape[0]/6)+3,6),
[1,2,3,3,3,3]).reshape(int(6*(int(shape[0]/6)+3)))


        answer=np.zeros((1,10))
        answer[:,8]=1
    if D==2:
        data=np.zeros((shape[0],shape[1]))
        data[0:2,0:2]=1
        for i in range (2,shape[0]):
            data[i,0:2]=data[i-1,0:2]+data[i-2,0:2]
        for j in range (2,shape[1]):
            data[:,j]=data[:,j-1]+data[:,j-2]
        answer=np.zeros((1,10))
        answer[:,8]=1
    if D==3:
        data=np.zeros((shape[0],shape[1],shape[2]))
        data[0:2,0:2,0:2]=1
        for i in range (2,shape[0]):

data[i,0:2,0:2]=data[i-1,0:2,0:2]+data[i-2,0:2,0:2]
        for j in range (2,shape[1]):
            data[:,j,0:2]=data[:,j-1,0:2]+data[:,j-2,0:2]
        for k in range (2,shape[2]):
            data[:,:,k]=data[:,:,k-1]+data[:,:,k-2]
        answer=np.zeros((1,10))
        answer[:,8]=1
    return data, answer

def generator10(shape):
#numerical
    D=len(shape)
    if D==1:
        data=np.arange(shape[0])*2
        answer=np.zeros((1,10))
        answer[:,9]=1
    if D==2:
        data=np.zeros((shape[0],shape[1]))
        for i in range (0,shape[1]):
            data[0:shape[0],i]=np.arange(shape[0])*2+2*i
        answer=np.zeros((1,10))
        answer[:,9]=1
    if D==3:
        data=np.zeros((shape[0],shape[1],shape[2]))
        for i in range (0,shape[1]):
            for j in range (0,shape[2]):

data[0:shape[0],i,j]=np.arange(shape[0])*2+2*i+2*j
        answer=np.zeros((1,10))
        answer[:,9]=1
    return data, answer
```

Figure C.1: This figure documented the ten generators used to generate Calculation Salient Matrices. Generator one to ten each contains an algorithm generating matrices for respectively problem one to ten. The generators are called by the data generator where randomness is added and matrices arranged for a given algorithm.