NEURAL RUBIK's -THESIS FINAL

by Gunasegarran Magadevan

Submission date: 29-Jul-2019 04:12PM (UTC+0800)
Submission ID: 1155892361
File name: Turninit_Submission.docx (3.11M)
Word count: 6887
Character count: 36462

CHAPTER 1: INTRODUCTION

1.1 Research Background

Neural networks have already proved capable of handling noisy and unstructured data such as hand-written texts, images, sounds, and real-world object classification based on an incomplete description.

They have also lately succeeded in numerous purely combinatorial fields, such as AlphaGo's game. Nowadays, the AlphaGo program (Silver, D, et al., 2016) that uses a Deep Neural Network can beat best human players who just two years ago were impossible. There is currently no other approach well-known to be skilled to play AlphaGo at this level. Neural Networks are also a vital component of DeepStack's best Poker engine (Moravčík, et al., 2017), and several attempts have been made to use them to solve Traveling Salesman Problems and other combinatorial issues (Bello, I. et al., 2017).

Machine learning approaches are already being used in development in several ways, such as selecting the best search algorithm, pre-processing the issue or promoting the search for promising areas. (Okechukwu Sunday Abonyi and Virginia Ogochukwu Umeh, 2014)

Heuristic learning is mainly used to rapidly judgments (of a "target attribute") that are computationally come to a solution that is hoped to be close to the best complex, and they instead substitute a more easily calculated possible answer, or 'optimal solution".(Okechukwu Sunday Abonyi and Virginia Ogochukwu Umeh, 2014)

Heuristic learning, where the undertaking is to automatically tempt a heuristic function from training samples exercising a model of machine learning, has also been given some attention. Models typically used in this area are straightforward and for specific problems are not fine-tuned. A lot of innovative probabilities are now accessible in this area with the recent rapid development of deep learning models. There are now learning algorithms for the practical

training of deep feed-forward networks, and many other types of Neural Networks have also developed and used effectively. For instance, there are Deep Recurrent Networks like Long Short – Term Memory (LSTM) (Morales et al., 2016), Deep Convolutional Networks, and Neural Turing Machines (Samadi, M et al., 2008).

1.2 Research Motivation

Rubik's Cube makes use of mathematical group theory, which has helped deduce specific algorithms. Furthermore, the fact that there are distinct subgroups, where each element of the group is a permutation in the Rubik's Cube. The group enables the puzzle to be learned and mastered by moving through different "difficulty levels", the 43 quintillion ways of solving the Rubik's Cube. These subgroups are the principle underlying the computer cubing methods by Thistlethwaite and Kociemba, which solve the cube by further reducing it to another subgroup. (Troy Fine and Boris Gorshenev, 2018)

There are now several solutions that can solve the cube in less than 100 steps. David Singmaster first published his solution in 1981, which solves the cube layer by layer. A team of researchers who worked with Google in July 2010 has proved that the so-called "number of God" (minimum number of moves to solve any) was 20. The Herbert Kociemba's Two-Phase Algorithm is used for the most move optimal online Rubik's Cube solver programs, which typically calculates a solution of 20 steps or less.

Since all these types of solutions require very tough mathematics to fully understand why they work and needs a tremendous amount of mathematical logic to develop new solutions, it is challenging for ordinary people and even impossible for computers to develop these solutions automatically.

With the hope that computer systems will learn how to solve Rubik's Cube with some general algorithm and after carefully searching previous work online, finally found that the most

popular methods that are used to solve Rubik's Cube problem are reinforcement learning. (Troy Fine and Boris Gorshenev, 2018)

However, this technique requires much computer power, therefore it involves finding alternatives. Since there aren't any neural network methods in heuristic learning, believe that this is because the objective function space is too large for neural networks to learn, but it's still worth attempting because there's no relevant experiment to prove it - and at least this method to solve Rubik's Cube with neural networks in heuristic learning could also fill the gap and provide data for this method. Neural Network represents an interesting and feasible way of joining few heuristics together and for some use-case scenarios, it might be the best decision. A perfect use-case situation for NN is where solved many problems from the same domain, there is enough time to prepare for the search (training the network) and optimal solutions are not strictly required (Robert Brunetto and Otakar Trunda, 2017).

1.3 Research Problem

- Neural Network can learn and model non-linear and complex relationships, extremely significant in light of the fact that, in actuality, a large number of the connections among information sources and yields are non-straight just as mind boggling. After learning from the initial inputs and their relationships, it can infer unseen relationships on unseen data as well, thus making the model generalize and predict on unseen data. Unlike many other prediction techniques, NN does not impose any restrictions on the input variables. Additionally, many studies have shown that Neural Network can better model
- Implementing neural network approaches to solve the Rubik's Cube have struggled to succeed without human help and have had to rely on hand-engineered features and group theory to systematically find solutions.

1.4 Research Question

- How to obtain a valuable function automatically, so that the whole problem-solving process is free of human knowledge, by using any of the traditional ways of the method by memorizing it.
- How to use the rules of an algorithm that performs from a goal state and try to learn from the goal how far it can reach the rules.

1.5 Research Objective

- To gain esteemed heuristic method mechanically, so that the total problem-solving process is free of human knowledge. Neural networks with more than one hidden layer tend to perform better than shallow ones. Determining several network architectures and parameters, all with more than 3 fully-connected hidden layers. For the estimator training, it is essential to generate models with a given distance or the scrambled number from the objective state— data examples of mixed cubes with distance is needed to be generated.
- To practice heuristic rules of the algorithm to perform random walks back from the goal state and try to learn how far the goal has been achieved. Two types of heuristic rules are used, Informed Search Algorithm (admissible and inadmissible).

1.6 Research Significance

- The finding of this research paper is to obtain valuable heuristic functions that the whole problem-solving process is free of human knowledge and as use heuristic algorithm rules that will perform random walks backwards from a goal state and try to learn how far they reached rules are from the goal. In order to do this, the Rubik's scrambled it back randomly from an objective state.

1.7 Research Organization

This thesis consists of five chapters. Firstly, Chapter 1 will discuss the overview of Rubik's Cube implementation on Neural Network, the motivation of the research, research problem, research objective, research question, research delivery, and research significance. Continually, the next chapter is a literature review. In this section, will discuss the overview of Rubik's Cube, and the algorithm used in other research papers (Pattern Database Learning, Boosting Learning, Autodidactic Iteration Learning, and Heuristic Learning) while comparing all these algorithms. In Chapter 3, methodology discussion on Heuristic Learning, using Admissible Learning and Inadmissible Learning. Later in Chapter 4, result and discussion, whereby various tests to compare different heuristics rules. Finally, Chapter 5 will conclude the research objective through the hypothesis made from the result.

CHAPTER 2: LITERATURE REVIEW

This chapter consists of a literature review of Rubik's Cube history as an introduction. Then continually the machine learning methods implemented in other research papers. Firstly, Pattern Database (PDB) Learning – solving Rubik's Cube by a method put away as a query table and applied to permutation problems. Secondly, the improvised machine learning method from PDBs is Boosting Learning - by using a weak prediction rule, the boosting algorithm associations these weak rules into a distinct prediction rule which more accurate. Thirdly, Autodidactic Iteration Learning (ADI) – an advance method of machine learning, also known as self-teaching Iteration, trains neural system esteem and strategy work through an iterative procedure. Finally, Heuristic Learning – a machine learning method which is an experience-based technique that assists in problem-solving, learning, and discovery. Similarly, as ADI but heuristic learning is mainly used to rapidly come to an optimal solution. Additionally, two heuristic algorithm which Admissible and Inadmissible are defined together. (*refer to Figure 2.1: Roadmap of Literature Review.*)

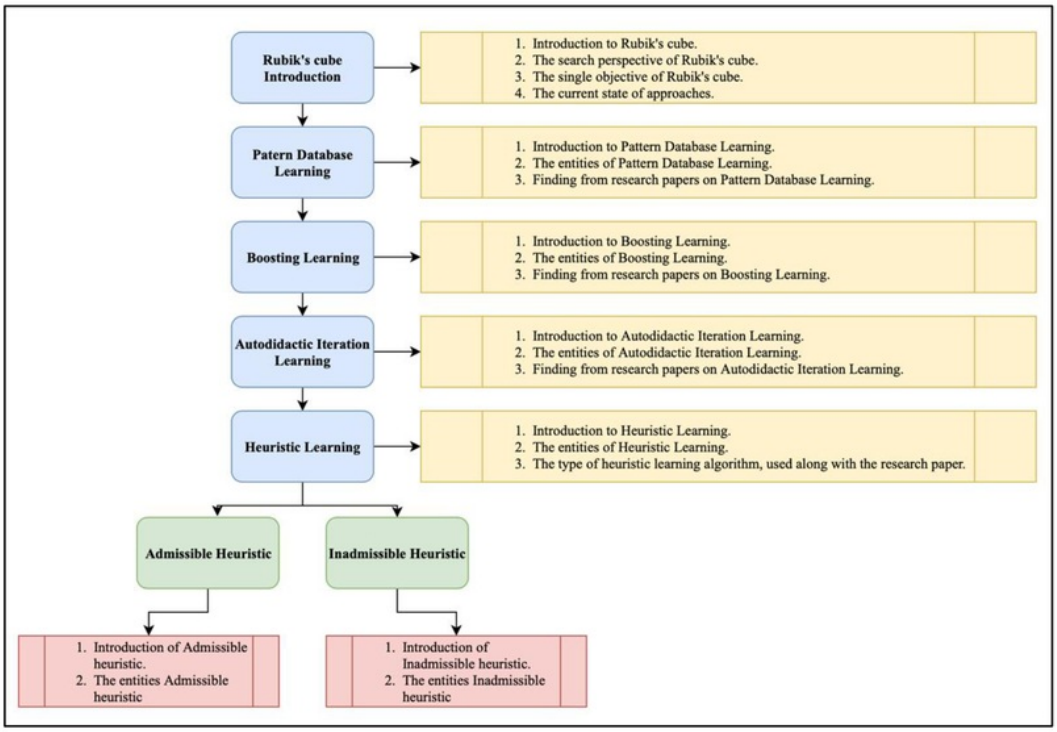

Figure 2.1: Roadmap of Literature Review.

2.1 Rubik's Cube Introduction

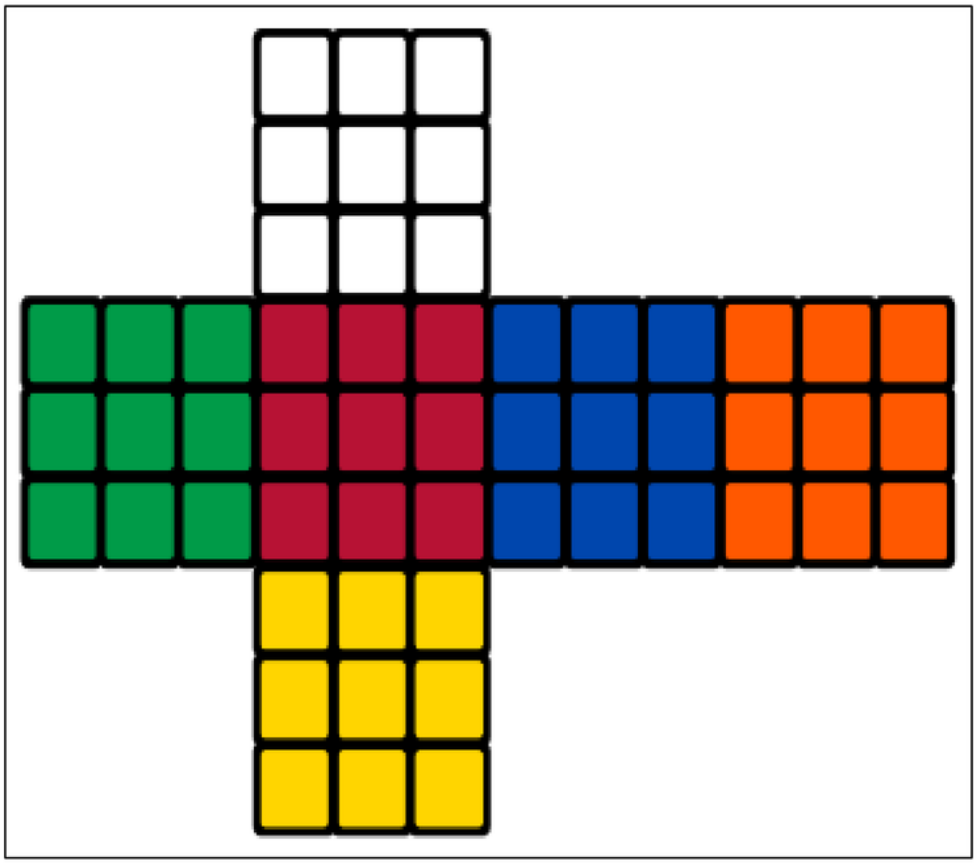

Figure 2.2: Rubik's Cube in 2 – Dimensional View.

Rubik's cube is a 3-Dimension (3D) puzzle, with 6 faces of a different colour. It consists of $3^3 = 27$ cubes. From a search perspective, the number of different states of the standard 3 by 3 cube is $\mathbf{8!} * \mathbf{3^7} * \frac{\mathbf{12!}}{\mathbf{2}} * \mathbf{12^{11}}$, which are over 43 quintillion legal positions of the Rubik's Cube, to be specific $\mathbf{43,252,003,274,489,856,000}$.

There is a solo objective state, and the fanning element is 12 when using symmetry breaking representation — considering a model utilizing quarter-move meaning that it is permitted to

turn layers just by 90◦. The half-move portrayal enables sides to be pivoted by 180◦ in a single move.

To be able to describe some different properties of the cube, firstly need to define a couple of additional thoughts. The cubes solid shape comprises of 27 little blocks which are known as cubes (*refer to Figure 2.2: Rubik's Cube in 2 – Dimensional View.)*. As discussed earlier, some faces of these cubes are coloured, or more precisely they carry coloured stickers. In the objective express, all appearances of the substantial 3 by 3 shape contain just stickers of a similar shading.

In the event dismantled the cube-square, by getting 8 corner- cubes, each of which carries 3 coloured stickers, 12 edge-cubes, each with 2 stickers on them and 6 central-cubes each having 1 colour (*refer to Figure 2.2: Rubik's Cube in 2 – Dimensional View.)*.

A cube shape of any size can without much of a stretch be understood not well utilizing a straightforward calculation that runs in time $\emptyset(n^2)$, where n is the size of the cube. Finding optimal solutions seems to be substantially harder, even though the complexity class of this task has long been unknown. It has only recently demonstrated that unravelling Rubik's cube optimally is certainly NP-hard (Demaine, E. et al., 2018). An issue is NP-hard if a calculation for solving it very well may be converted into one for solving of any NP-problem (nondeterministic polynomial time) issue. NP-hard in this manner signifies "in any event as hard as any NP-issue," in spite of the fact that it may, actually, be more enthusiastically (Weisstein, 2019). Current state-of-the-art approaches for finding an ideal solution or close ideal solution frequently utilize forward state-space search using a pattern database as a heuristic (Sturtevant, N. R et al., 2014).

2.1.1 Pattern Database Learning

Pattern database (PDB) is a heuristic method that has been removed as a query table and applied to permutation issues. (Culberson J.C., 1996) PDBs were initially proposed to solve the problem of a solo agent. Nowadays, they are unique of the most popular heuristics methods. (Culberson J.C., 1996)

The arrangement of each database pattern frames the database area. Assumed any variation, the example can be gazed upward in the pattern database to locate the insignificant number of tasks essential to put this *N* element in their right areas. A pattern database can be seen as a gathering of answers for sub-goals that must be accomplished to solve the problems. (Alevizos – A – Bastas, 2017)

Although using all PDBs and reflecting through the centre symmetry gave the best search results, the speediest arrangement used only the corner, the final seven edges and the first seven edges PDBs. To recap, there is a trade-off among the search effort and the node generation cost. The further the lookups, the lesser the search effort but, the more significant the cost in terms of time to generate a node. This is likewise where the power of locality preservative pattern databases lies. Neighbouring values can be trimmed without having to do a separate lookup, and this means that all searches required to generate the pruned nodes are translated into again in terms of time. (Alevizos – A – Bastas,2017)

An interesting observation from the research paper is that exercising the corner and the two edge PDBs gave nearly the same results in terms of search effort as using the corner and the last seven edge PDBs along with replication through the symmetry of the front face. They had the worst performance with regard to dual lookups. This can be explained as the neighbouring values of the dual state can not be used to prune the neighbours of the normal state. (Alevizos – A – Bastas,2017)

2.1.2 Boosting Learning

Boosting, a machine-learning method is the perception in discovering many unpleasant general guidelines can be significantly simpler than finding a solitary, profoundly exact expectation rule. To apply the boosting approach, start with an algorithm for finding the uneven rules of thumb. This also called as a "weak" or "base" learning algorithm constantly, each time filling it a different subset of the training examples. Each time the algorithm is called, the base learning algorithm generates a new weak prediction rule, and after numerous rounds, the boosting algorithm must combine these weak rules into a single prediction rule that, hopefully, will be much more accurate than anyone of the weak rules. (Robert E. Schapire)

Following research, (Alexander Irpan, 2016), started by rereading boosting algorithms from the text, before recommending and testing their variation. For the rest of this paper, f^{th} signifies the weak learner for the f^{th} iteration, Dt is the distribution throughout data that f^{th} is trained on, and f^{th} is the active learner at the end of the t^{th} iteration. Also use standard classification notation, where X is the data space, and Y is the label space.

As a conclusion from the resulting paper, there are some hypotheses for why boosting failed to improve presentation. Assuming that boosting works healthier when weak learners are cheap to train and evaluate. This allows the researcher to generate an ensemble of many hundreds or thousands of learners, instead of tens. Furthermore, weak learners should oppose on the optimal action when given the same input. Deep neural networks are costly to train, so do not have the time to train many weak learners - in testing, the researcher only had time to train 20 weak learners. Besides, since all gradient apprises were applied to the same neural network, the neural network was standardized against changing too much each iteration. Upon manual examination, weak learners almost always agree on the optimal action

(Alexander Irpan, 2016). Hypothetically, a boosting method will work with sufficient iterations, but boosting large neural networl appears to be less effective than only applying all data into a single neural network.

2.1.3 Autodidactic Iteration Learning

Autodidactic Iteration (ADI) or accepted as self-teaching Iteration, trains neural system esteem and approach work over an iterative process. In respectively iteration, the inputs to the neural network are produced by beginning from the goal state and randomly taking actions.

The aims achieve to approximation the optimal value function by accomplishment a breadth-first search from each input state-run and using the existing network to conclude the value of each of the leaves in the tree. Simplified value estimates for the root nodes are obtained by recursively backing up the benefits for each node using a maximum operator. The strategy network is likewise trained by generating targets from the move that maximizes the value. (McAleer, S. at el., 2018)

ADI is an iterative supervised learning technique that trains a deep network of neural parameters for the input state and the output of a value and policy pair (v, p). The strategy output p is a vector with the possibilities to move from the state for each of the 12 moves. The strategy will be used to cutback breadth and the value to reduce depth in the Monte Carlo Tree Search (MCTS) once the network is trained. Training examples for f are generated from the solved cube for each Autodidactic Iteration. This guarantees that specific training inputs are satisfactorily close to have a positive reward for a shallow search. Targets will then be created by conducting the first-width breadth search (BFS) of each sample of training. To estimate the value of each child, the current value network is used. The maximum value and

reward for each sample of their children is the value target, and the strategy objective is the action foremost to this maximum value. (McAleer, S. at el., 2018)

In the following research, (McAleer, S. at el., 2018), compares DeepCube against two different solvers. The main standard is the Kociemba two-stage solver. This algorithm depends on human area learning of the gathering hypothesis of the Rubik's Cube. Kociemba will dependably understand any cube given to it, and it runs in all respects rapidly. In any case, in light of its broadly useful nature, it regularly finds an increasingly stretched out arrangement contrasted with different solvers. The distinctive standard is the Korf Iterative Deepening A* (IDA*) with an example database heuristic. Korf's algorithm will dependably locate the ideal arrangement from some random beginning state; be that as it may, since it is a heuristic tree search, it will frequently need to investigate a wide range of states, and it will set aside an extended effort to process an answer. The specialist additionally looks at the full DeepCube solver against two variations of itself. To start with, they don't figure the most limited way of our hunt tree and rather extricate the underlying way from the MCTS: this will be termed Naive DeepCube. Additionally, utilize the prepared esteem organize as a heuristic in a voracious best-first scan for a straight-forward assessment of the esteem arrange: this will be named Greedy. Subsequently, the researcher unable include Korf in the past examination since its runtime is restrictively moderate Note that Korf has one anomaly that is over 15 moves.

2.1.4 Heuristic Learning

Heuristic - a word derivative from the Greek language for ‘find’ or discover is an adjective experience-based technique that helps in problem-solving, learning, and discovery. (Katazyna at el., 2006)

A heuristic learning is mainly used to rapidly judgments (of a "target attribute”) that arecomputationally come to a solution that is hoped to be close to the best complex, and they instead substitute a more easily calculated possible answer, or 'optimal solution". A heuristic learning is mainly used to rapidly come to a solution that is hoped to be close to the best possible answer, or 'optimal solution'. (Okechukwu Sunday Abonyi and Virginia Ogochukwu Umeh, 2014)

The term heuristic is consumed for algorithms which determine preparations among every possible one, though, they don't guarantee that the best will be seen, in this manner, they might be considered as around and not accurate calculations. These algorithms usually find a solution close to the best one, and they find it fast and efficiently. The method utilized from a heuristic calculation is one of the visible methods, such as greediness, yet to be agreeable and fast, the calculation supervises. (Okechukwu Sunday Abonyi and Virginia Ogochukwu Umeh, 2014)

Two types of heuristics algorithm are utilized, firstly, an admissible heuristic or informed search is utilized to assess the expense of achieving the objective state in the heuristic algorithm. All together for a heuristic to be permissible to heuristic, the assessed cost should dependably be lower than or equivalent to the actual cost of achieving the objective state. The heuristic algorithm utilizes the admissible heuristic to discover an expected ideal way to the objective state from the current state. While an inadmissible heuristic might overemphasize the cost of accomplishment the goal. It might or might not result in an optimal solution. Though, the lead is that sometimes, an inadmissible heuristic multiplies much fewer

nodes. Therefore, the total cost (= search cost + path cost) may essentially be lower than an optimal solution using an admissible heuristic. (Okechukwu Sunday Abonyi and Virginia Ogochukwu Umeh, 2014)

2.1.5 Comparison of Learning

Different machine learning methods that can be used to solve the cube problem of the Rubik have been studied based on the literature review. A "heuristic" is an algorithm that does not ensure right solution. A "good" heuristic is one that will get either the right or an adequate solution more often than not. Heuristics are utilized when either there is no known solution calculation, or keen on an option that is quicker than the known arrangement algorithm. To conclude, heuristics learning is the most appropriate approach and has the potential for improvement. (Table 2.1: Learning comparison)

As concluded, this research paper will implement two types of heuristics algorithms.

When solving a problem, one can have two kinds of objectives in mind (minimize the path cost of the solution and minimize the time taken to find the solution). Of course, in most cases, both objectives are important. A balance has to be struck between the two, and that is where the heuristic comes in. An admissible heuristic is optimal, it will always find the cheapest path solution. On the other hand, an inadmissible heuristic is not optimal, it may result in a suboptimal solution but may do so in a much shorter time than that taken by an admissible heuristic.

Table 2.1: Comparison of Learnings

Learning	Approach
Pattern Database Learning	PDB is a heuristic method put away as a query table and were applied to permutation problems.
Boosting Learning	Boosting, a machine-learning method that is the observation that finding various rough rules of thumb can be a lot easier than finding a single, highly accurate prediction rule.
Autodidactic Iteration	ADI trains neural system esteem and strategy work through an iterative procedure.
Heuristic Learning	Heuristic learning s a solution that is hoped to be close to the best possible answer, or 'optimal solution'

CHAPTER 3: RESEARCH METHODOLOGY

This chapter consists of research methodology used along with this research paper, which are Quantitative Research, Informed Search Algorithm and Learning the trained model. (*refer to Figure 3.1: Roadmap of Research Methodology.*)

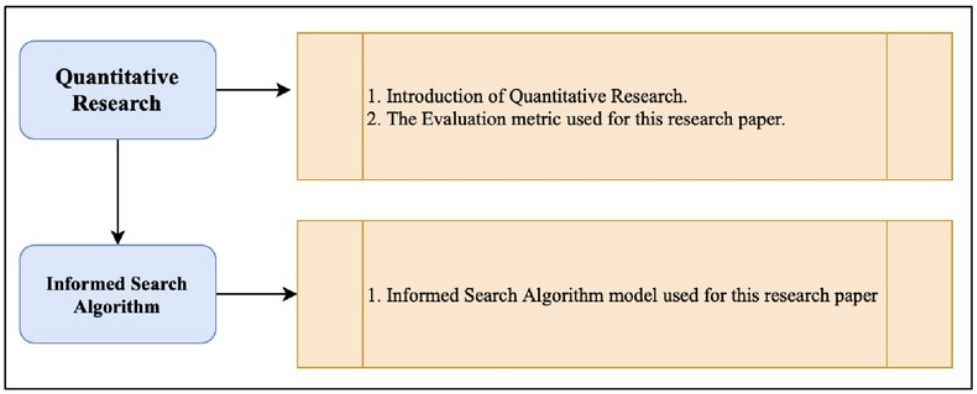

Figure 3.1: Roadmap of Research Methodology.

3.1 Quantitative Research

Quantitative research techniques are examined strategies managing numbers and anything quantifiable in an orderly method for examination of wonders and their connections. It is applied when a question is answered based on relationships within measurable variables to explain, predict and control a fact (Paul D. Leedy et al. 2018).

Quantitative is applied in this research paper, where one way to accomplish the research objective is to exploit statistical regularities of the problems.

3.2 Informed Search Algorithm

In this research paper, an informed search A^* with different heuristic used and as well as machine learning estimator. The informed search A^* heuristics are described below (*refer to Figure 3.2: Flowchart of Informed Search Algorithm*).

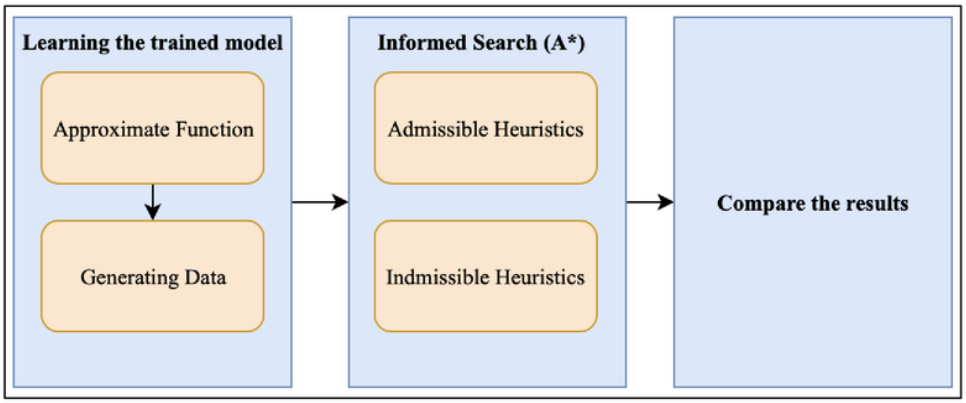

Figure 3.2: Flowchart of Informed Search Algorithm.

Before the informed search A^* with different heuristic is implemented, a learning method in initiates before. Learning the trained model is used as an approximate function via Deep Neural Networks to train data an generates the trained data by allocating the size of the scramble randomly. Later the Informed Search Algorithm A^* different heuristic using the trained data, the admissible and inadmissible heuristic ruled are applied. Finally, from the admissible and inadmissible rules applies, the results are compared.

3.2.1 Learning the trained models

For this part, the trained model be used as an approximate function to learn, by using the Deep Neural. The steps explained below.

3.2.1.1 Approximate Function

Using a monomorphism algorithm, the idea of monomorphism is the speculation of the thought of injective guide of sets from the classification Set to subjective classifications. Let C be the space of all 3 by 3 of Rubik's Cube. The aim is to train a model that works closely with $h^*: C \rightarrow \mathbb{N}$ that maps every cube setup to its shortest distance from the target state. To achieve this, it is decided to train a regression model of the Deep Neural Network (DNN) in a supervised learning setup.

3.2.1.1.1 Deep Neural Networks

It is known that any other continuous function over compact subsets of $\mathbb{R}^n$ can be combined with arbitrary precision using a neural feed network (Csáji, 2001). Theoretically, neural networks have more than one layer hidden tend to do better than shallow networks. It is termed several network architectures and parameters with more than three hidden layers fully connected.

Table 3.1: Histogram of cube configurations (http://cube20.org/qtm/).

Distance	Count of Positions	Distance	Count of Positions
0	1	14	50,729,620,202,582
1	12	15	472,495,678,811,004
2	114	16	4,393,570,406,220,123
3	1,068	17	40,648,181,519,827,392
4	10,011	18	368,071,526,203,620,348
5	93,840	19	about 3,000,000,000,000,000,000
6	878,880	20	about 14,000,000,000,000,000,000
7	8,221,632	21	about 19,000,000,000,000,000,000
8	76,843,595	22	about 7,000,000,000,000,000,000
9	717,789,576	23	about 24,000,000,000,000,000
10	6,701,836,858	24	about 150,000
11	62,549,615,248	25	36?
12	583,570,100,997	26	3?
13	5,442,351,625,028	>26	0

3.2.1.2 Generating Sample Data

For the estimator training, it is essential to generate examples with a given distance from the objective — data examples of mixed cubes with distanced needed to be generated. In order to do this, went back (to original state) randomly from an objective state. That provides a cube setup that is long, as it cannot be sure that there are no shorter walks that will produce the same result.

With the distance (*up to* $d \approx 20$, *refer to Figure 3.3: Histogram of cube configurations (http://cube20.org/qtm/).*), it is necessary to increase exponentially the number of configurations with a given distance, so that intuitively the probability of a configuration of distance d (not one with a smaller distance) should be high. In order to train the estimator, it

must be generated numerous configurations of random cubes (almost 80,000) per training session, so that although the above is present, configurations may remain shorter than intended.

Table 3.2: Average solution length, taken over 100 examples for each scramble (d).

d	7	8	9
$\bar{h}_3$	7.02	8	9
$\bar{h}_4$	7.02	8	9.14
$\bar{h}_5$	7.02	8	9

Used three methods to remove these cases:

1. (Trivial) Keep a set of visited states and make sure that does not repeat, and already visited the state during the walk.
2. Three or more identical movements disallow. The sequence F, B, F, B, F', for example, is limited to F'.
3. If they commute, two actions are independent. Independent actions, in this case, are actions that operate on the face of the other cube (the front and the back). Later decide to order those actions (the front is smaller than the back) for any two independent actions. If two next actions are independent, the smaller one must come first in any given sequence of actions. For this purpose, generating a more extended sequence of measures in the random walk and then use a variant of Bubble Sorting to exchange inappropriate actions. Finally, used the previous steps to reduce redundant measures after sorting the sequence.

3.2.1.2.1 Allocation for Choosing the scrambles (d)

As noted, each example of training is produced by selecting the length of the random path, indicated by d first and then creating an altered sequence of measures of length d.

As the number of configurations grows exponentially, it does not make much sense to uniformly choose d from a set $\{0,1,\ldots,d_{max}\}$. In fact, the learned model was not performable on larger distances after trying to train this way.

On the other hand, the choice of d does not work well either according to the "actual" configuration allocation. Almost every time the d_{max} value is selected, and the model does not perform well in short-distance settings. Such a model could not be used in A^* as a solved cube is not recognized. The solution was to take a convex combination of the two distributions. Indicate by $\vec{\rho}$ the real distribution , and by $\vec{u}$ the uniform one, both over $\{0,1,\ldots,d_{max}\}$, at that point pick d from this set by $\alpha\vec{\rho}$ +(1- α) for $\alpha \in (0,1)$, which is clearly a substantial distribution. Finally chose $\alpha = 0.1$ which appeared to work best.

3.3.1 Admissible Heuristics

For this research paper, two admissible heuristic rules are used, Maximum Misplaced Edge Cubes in Face of Rubik's Cube and Minimum Misplaced Edge Cubes in Rubik's Cube Face.

3.3.1.1 Maximum Misplaced Edges Cubes in Rubik's Cube Face

Maximum Misplaced Edge Cubes in Rubik's Cube Face is one of the admissible heuristic rules. In this rule, which finding the maximum misplaced edges of the misplaced edges of cubes from the original centrepiece of the cube. This due the random scrambled may contain one of the permutations of misplaced edges of the cube.

Each face of Rubik's cube, more than one number of wrong edge cubes will take place. For example, the white face of Rubik's cube will have a maximum of 4 misplaced of yellow edges cube (*refer to Figure 3.4: Cube notations for Misplaced Edges Cube*).

This pattern is acceptable as an admissible rule because in a solved cube the edge cubes have to be the same colour as the centre cube and at least one action is needed to "fix" each one of them. This heuristic is larger than 0 and is limited to 4. In the mathematical notation of machine learning term:

$$h_1(c) = \frac{maximum}{f \in Faces\ (c)}\ \#misplaced - edges\ (f)$$

Equation 3.1: Maximum Misplaced Edge Cubes in Rubik's Cube Face.

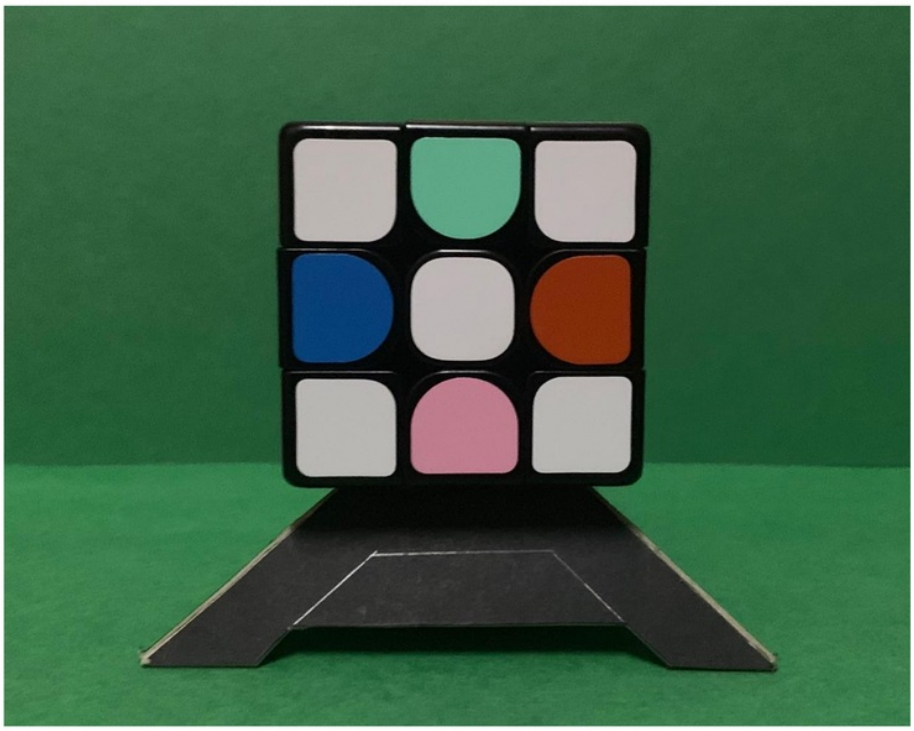

Figure 3.4: Cube notations for Misplaced Edges Cube.

3.3.1.2 Maximum Misplaced Corners Cubes in Rubik's Cube Face

Maximum Misplaced Corner Cubes in Rubik's Cube Face is one of the admissible heuristic rules. In this rule, which finding the maximum number misplaced corner of the misplaced corner of cubes from the original centrepiece of the cube. This due the random scrambled may contain one of the permutations of misplaced corners of the cube.

Each face of Rubik's cube, more than one number of wrong edge cubes will take place. For example, the yellow face of Rubik's cube will have a maximum of 4 misplaced of corners cube (*refer to Figure 3.5: Cube notations for Misplaced Corners Cube*).

This pattern is acceptable as an admissible rule because in a solved cube the edge cubes have to be the same colour as the centre cube and at least one action is needed to "fix" each one of them. This heuristic is larger than 0 and is limited to 4. In the mathematical notation of machine learning term:

$$h_2(c) = \frac{maximum}{f \in Faces\ (c)}\ \#misplaced - corners\ (f)$$

Equation 3.2: Maximum Misplaced Corners Cubes in Rubik's Cube Face.

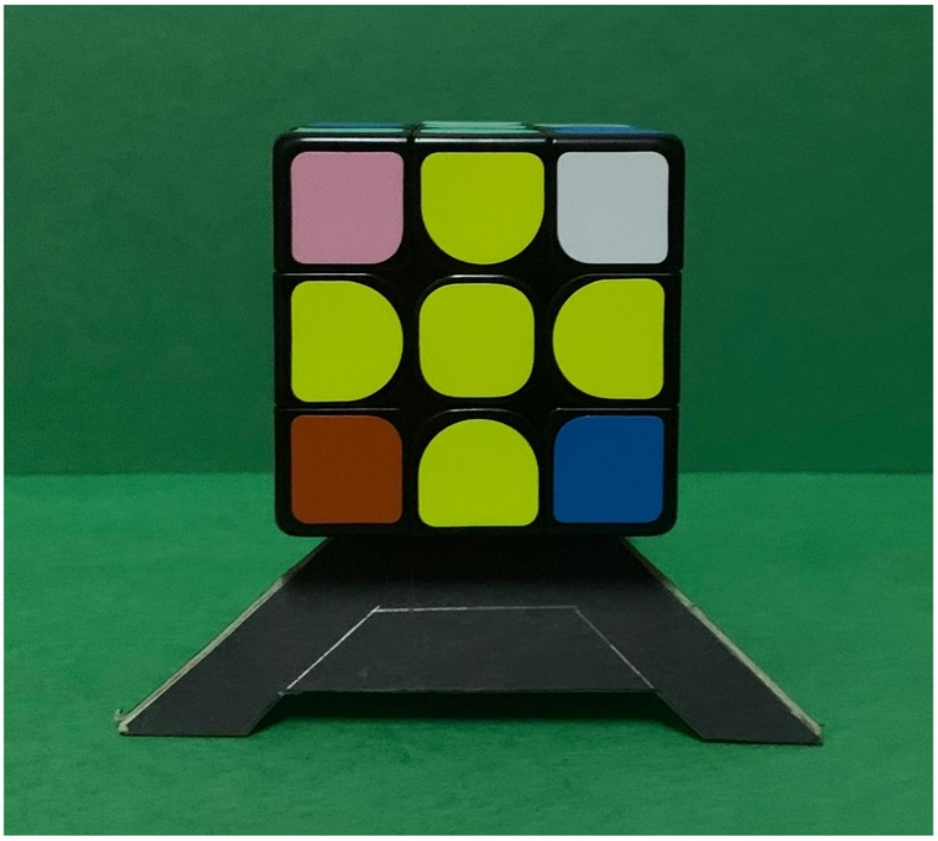

Figure 3.5: Cube notations for Misplaced Corners Cube in Rubik's Cube Face.

3.3.2 Inadmissible Heuristics

For this research paper three admissible heuristic rules are used by finding the non-optimal path, Maximum Misplaced Edges and Corner Cubes in Face of Rubik's Cube, Maximum and Minimum Misplaced Corner and Edge Cubes in Face of Rubik's Cube and Maximum Misplaced Corner and Maximum Misplaced Edge Cube in Face of Rubik's Cube.

3.3.2.1 Maximum Misplaced Edges and Corner Cubes in Rubik's Cube Face

Maximum Misplaced Edges and Corner Cubes in Rubik's Cube Face is one of the inadmissible heuristic rules since this rule is randomly estimating the misplaced edges and corner. In this

rule, which finding the maximum misplaced corner of the misplaced edges and corner of cubes from the original centrepiece of the cube.

Each face of Rubik's cube, more than one number of the wrong edge and corner cubes will take place. For example, the red face of Rubik's cube, will have greater than 0 for an unsolved cube piece and limited to 8.

This pattern is acceptable as an inadmissible rule because, in a solved cube, the edge and corner cubes have to be the same colour as the centre cube to 'fix', in the non-optimal path. In the mathematical notation of machine learning term:

$$\bar{h}_3(c) = \frac{maximum}{f \in Faces\ (c)} [\#misplaced - edges\ (f) + \#misplaced - corners\ (f)]$$

Equation 3.3: Maximum Misplaced Edge and Corner Cubes in Rubik's Cube Face.

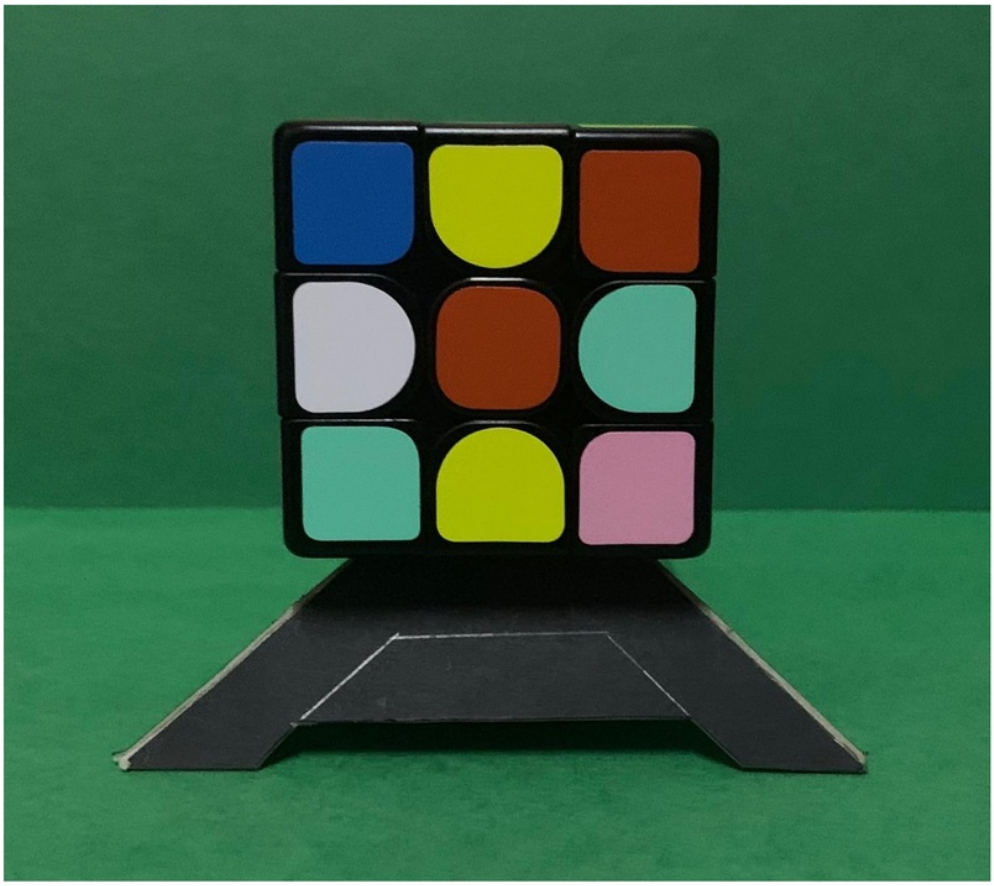

Figure 3.6: Cube notations for Maximum Misplaced Edge and Corner Cubes in Rubik's Cube Face.

3.3.2.2 Maximum and Minimum Misplaced Corner and Edge Cubes in Rubik's Cube Face

Maximum and Minimum Misplaced Edges and Corner Cubes in Rubik's Cube Face is one of the inadmissible heuristic rules since this rule is randomly estimating the misplaced edges and corner. In this rule, which finding the maximum and minimum misplaced corner of the misplaced edges and corner of cubes from the original centrepiece of the cube.

Each face of Rubik's cube, more than one number of the wrong edge and corner cubes will take place. For example, the red face of Rubik's cube, will have greater than 0 for an unsolved cube piece and limited to 16.

This pattern is acceptable as an inadmissible rule because, in a solved cube, the edge and corner cubes have to be the same colour as the centre cube to 'fix', in the non-optimal path. In the mathematical notation of machine learning term:

$$\bar{h}_4(c) = \frac{maximum}{f \in Faces\ (c)}\ g(f) + \frac{minimum}{f \in Faces\ (c)}\ g(f)$$

Equation 3.4: Maximum and Minimum Corner and Edge Cubes in Face.

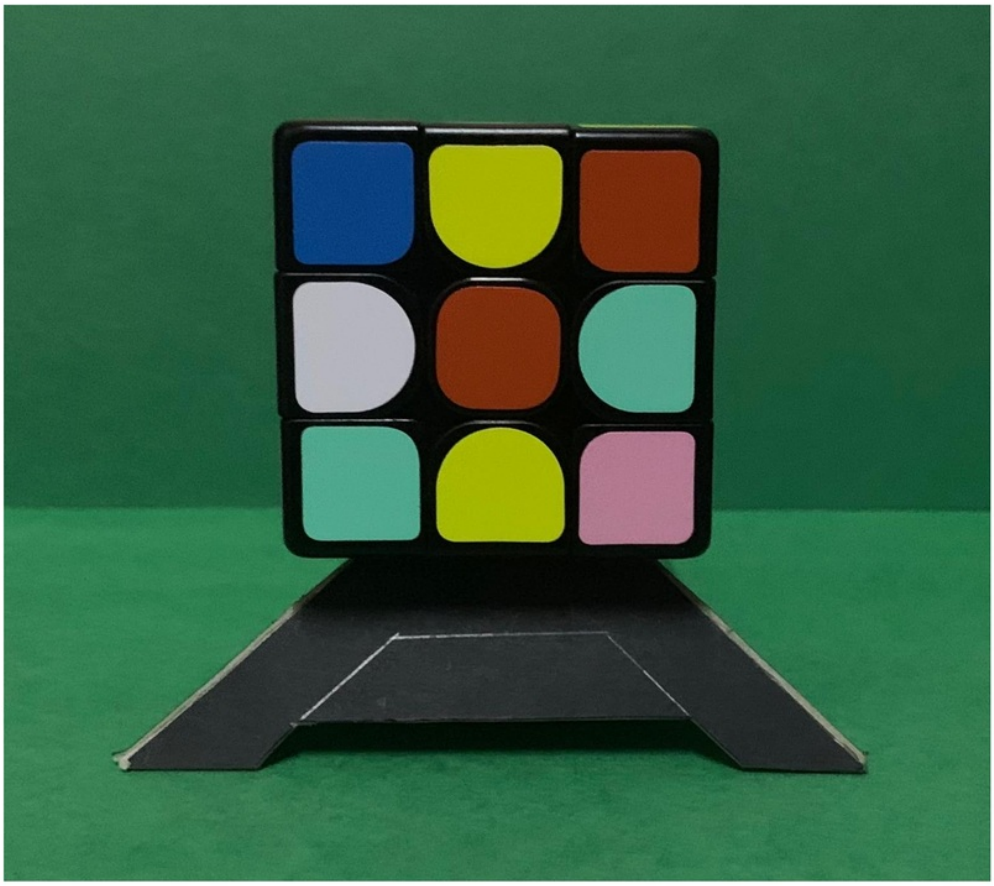

Figure 3.7: Cube notations for Maximum and Minimum Corner and Edge Cubes in Face.

3.3.2.3 Maximum Misplaced Corner and Maximum Misplaced Edge Cube in Face

Maximum Misplaced Corner and Maximum Misplaced Edges Cubes in Rubik's Cube Face is one of the inadmissible heuristic rules since this rule is randomly estimating the misplaced edges and corner. In this rule, which finding the maximum misplaced corner of the misplaced edges and corner of cubes from the original centrepiece of the cube.

Each face of Rubik's cube, more than one number of the wrong edge and corner cubes will take place. For example, the red face of Rubik's cube, will have greater than 0 for an unsolved cube piece and limited to 8.

This pattern is acceptable as an inadmissible rule because, in a solved cube, the edge and corner cubes have to be the same colour as the centre cube to 'fix', in the non-optimal path. In the mathematical notation of machine learning term:

$$\bar{h}_5(c) = h_1(c) + h_2(c)$$

Equation 3.5: Maximum Misplaced Corner and Maximum Misplaced Cube Cubes in Face.

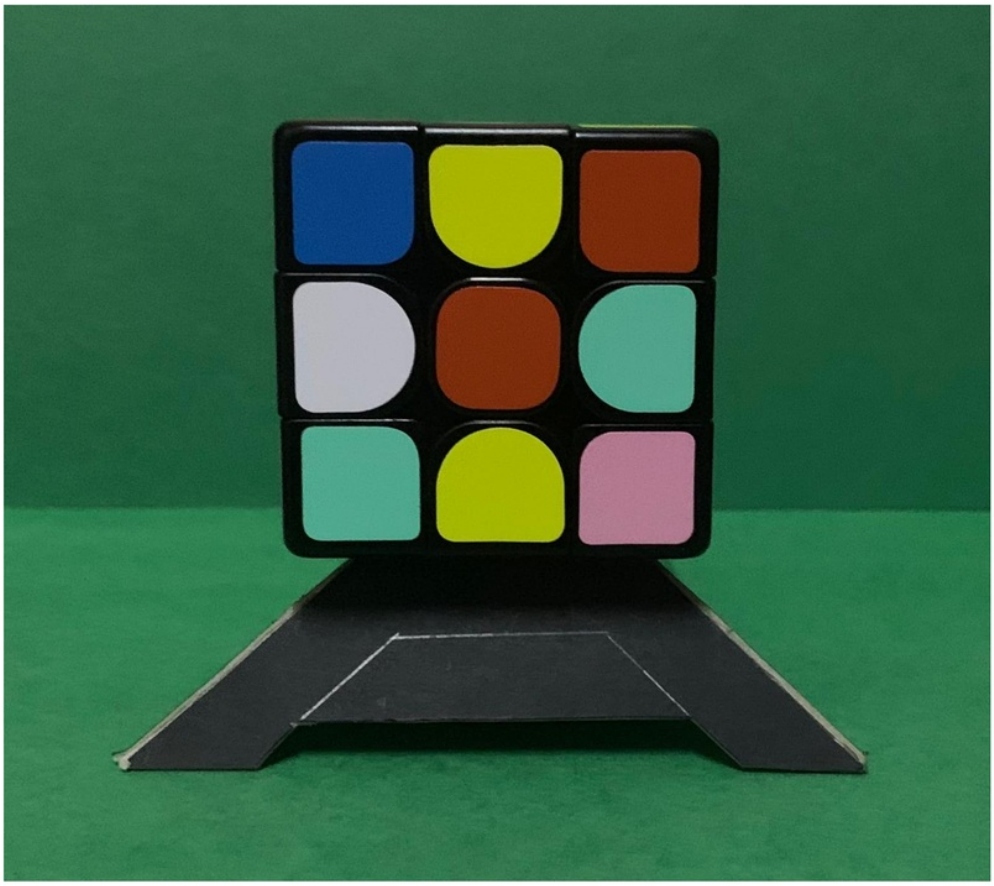

Figure 3.8: Cube notations for Maximum Misplaced Corner and Maximum Misplaced Cube Cubes in Face.

CHAPTER 4: RESULT AND DISCUSSION

In this research paper, several tests are run to compare different heuristics. Firstly, compare the regular heuristics (human-made heuristic, the trained model using heuristic rules), with each other. Then, compare the learned heuristics (the learned model from regular heuristic), with each other. Finally, compare the best heuristic from the first group to the respective one from the second group. The results are analyzed as a number of nodes expanded to include the A^* algorithm during operation is reasonable in the performance of various heuristics.

4.1 Python Scripts

The scripts are split into two main parts, which are provided by the author of "*Solving the Rubik's Cube Without Human Knowledge, 2018*", for benchmarking. (Forest Agostinelli, 2018).

1. **The cube module** – Consists of a simple class representing a cube, the Rubik's Cube problem written as an A^* Search Algorithms and some other useful functions.

   PyPI package link: https://pypi.org/project/rubikai/

2. **Jupyter Notebooks 3** – Virtually hosted on the Google Collaboratory, which contains two parts:

   I. Model training:
   https://drive.google.com/open?id=1fDSMxdsxiGHj7tAJlTtCsfPYl4PZP5p7

   II. Main notebook:
   https://drive.google.com/open?id=1pVlrv83zAFDMYCok2cHIKIJbf9kCS6XT

4.2 Results

4.2.1 Trained Model Using Deep Neural Network

Three models are trained for a DNN regression with the different layer and neuron numbers, over 25 scrambles:

h_1: has three layers, of 70, 60 and layer of 50 neurons, respectively.

h_2: has four layers and 50 neuron levels each layer.

h_3: has five layers, with neurons: 50, 40, 30, 20 and 20.

All models have been trained on 100,000 labelled cube examples, with about the same loss and training curves (*refer to Figure 4.1: Learning Curves of Training Data for Plateau from 100 to 200 Epoch*.). Based on the trained data with about the same loss and training curves. Learning curves are broadly utilized in machine learning for algorithms hat adapt gradually after some time. During the training of a machine learning model, the current state of the model at each step of the training algorithm can be evaluated. It can be evaluated on the training dataset to give an idea of how well the model is "*learning*". It seems as if the learning curve reaches a plateau and stopped training because the assessment loss is similar (*refer to Figure 4.2: Model Evaluation of Average over 1600 examples.),* the evaluation loss was compared to the heuristics (*refer to Figure 4.3: Result of Heuristics Comparison Just Learned of Average over 10 iterations*.).

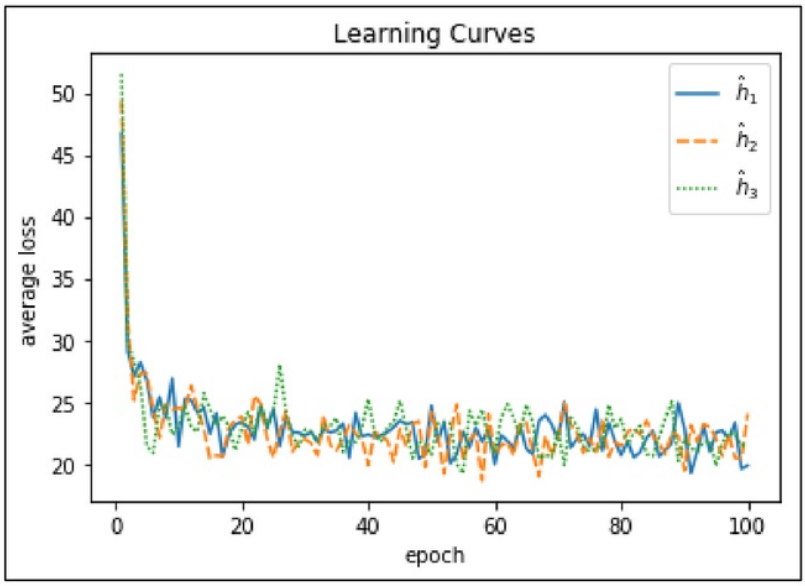

Figure 4.1: Learning Curves of Training Data for Plateau from 0 to 100 Epoch.

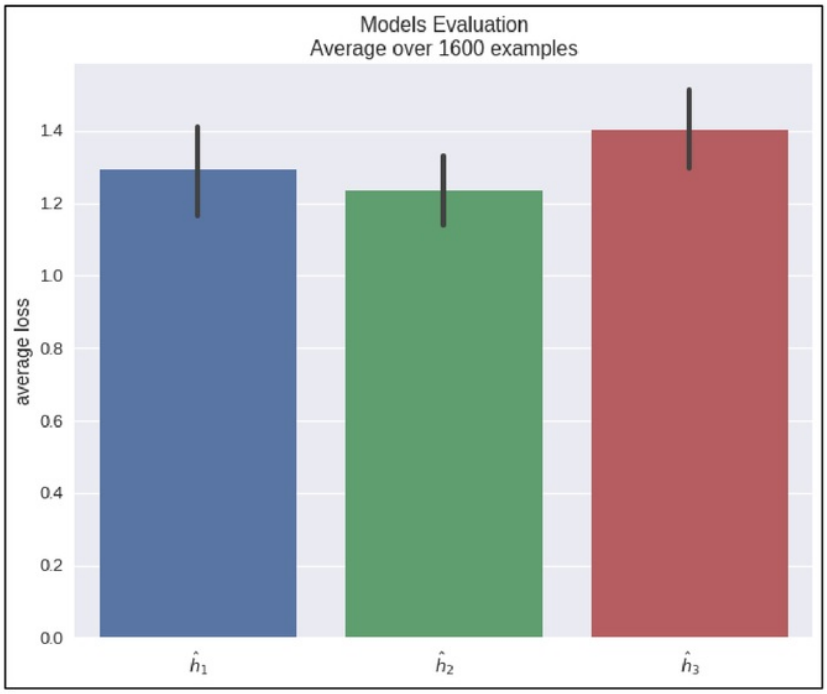

Figure 4.2: Model Evaluation of Average over 1600 examples.

4.2.2 Admissible Over Inadmissible.

As anticipated, the heuristics allowed for $d > 7$ (*refer to Figure 4.3: Result of Heuristics Comparison Just Regulars of Average over 10 iterations.*) were too inefficient, so only used the inadmissible values for larger scrambles (d) values. These actions were successful in solving up to $d = 25$ moves.

The heuristic $\bar{h}_4$ has been especially successful with averages of 1, 500 nodes for 25 scramble movements from the originally trained model (*refer to Figure 4.3: Result Heuristics Comparison Just Regulars of Average over 10 iterations*).

Furthermore, inadmissible heuristics did not significantly affect the optimal solution. To test this, checked whether the returned length of the solution exceeds the number of scramble moves for this example. The results are shown in *Table 3.1: Average solution length, taken over 100 examples for each d.*, where for scramble = 1 to 6, the solutions returned have always been optimum.

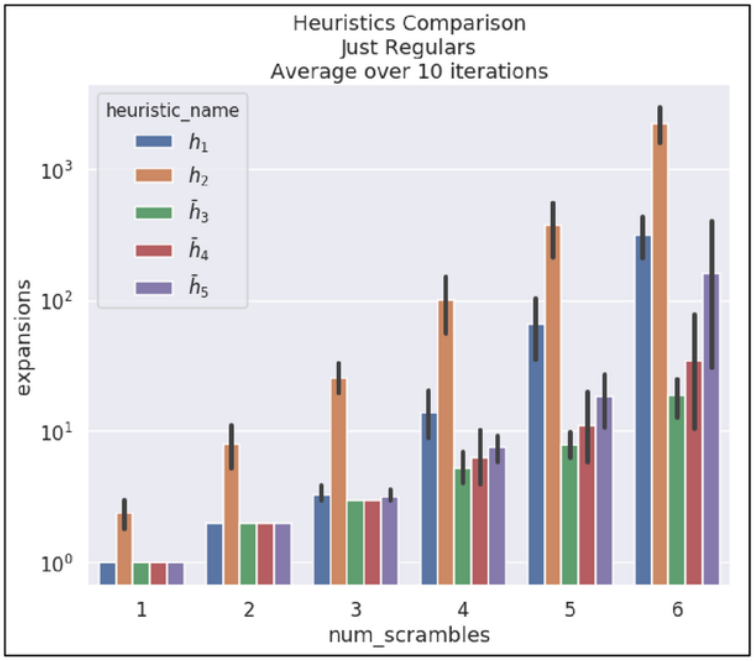

Figure 4.3: Result of Heuristics Comparison Just Regulars of Average over 10 iterations

4.2.3 Learned Over Human-Made Heuristics

The learned heuristic, $\bar{h}_2$, is marginally better than the best human-made, $\bar{h}_4$ as shown in the results (*refer to Figure 4.4: Result of Heuristics Comparison Regular Versus Learned of Average over 5 iterations.*). The following plot the iteration decreases to 5 iterations since only comparing learned heuristic, $\bar{h}_2$ against the human-made, $\bar{h}_4$, which already plotted in*to Figure 4.3: Result Heuristics Comparison Just Regulars of Average over 10 iterations*.

During plotting the comparison, $\bar{h}_2$, it was relatively accurate and expected it to perform better.

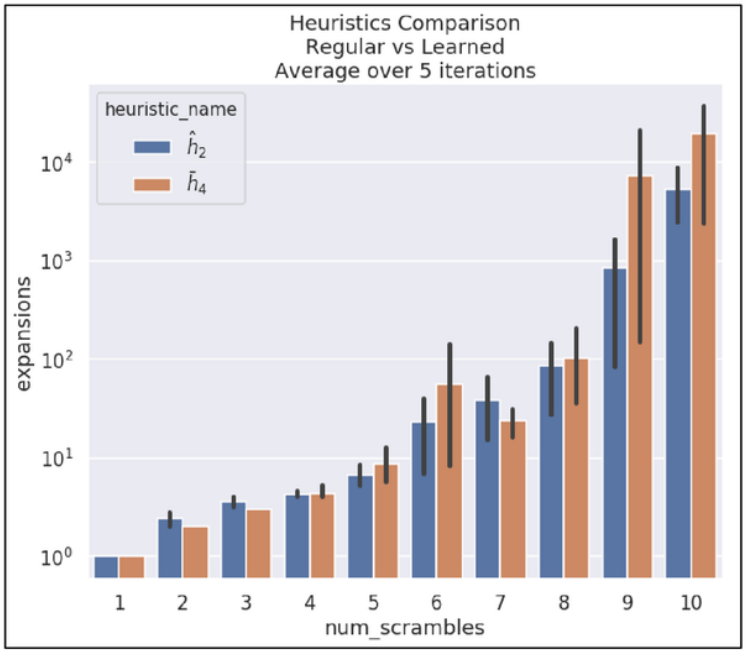

Figure 4.4: Result of Heuristics Comparison Regular Versus Learned of Average over 5 iterations.

CHAPTER 5: CONCLUSION AND FUTURE RESEARCH

5.1 Conclusion

The learned heuristics were generally perfect compared to those produced by humans. In particular, as the learning process has become limited, there have been relatively few parameters tweaked, and not more than three hours of training time per model, which is significantly less than other Rubik's Cube solvers. (Stephen McAleer, 2018)

Neural networks with more than one hidden layer tend to perform better than shallow ones. Consequently, this research considered several network architectures and parameters, all with more than 3 fully-connected hidden layers. For the estimator training, it is important to generate examples with a given distance or the scrambled number from the objective state— data examples of mixed cubes with distance is required to be generated. In order to do this, scrambled it back randomly from an objective state (solved state of Rubik's Cube). That provides a cube arrangement that is long, as it cannot be guaranteed that there are no shorter walks that will produce the same result. With the distance of finding the solution, it is compulsory for the number of configurations to increase exponentially with a given distance so that automatically the probability of a configuration d should be high.

Finally, as expected, the heuristics allowed for $d > 7$ were too inefficient, so only used the inadmissible values for larger scrambles (d) values. These actions were successful in solving up to d = 25 moves. The heuristic $\bar{h}_4$ has been successful especially with averages of 1, 500 nodes for 25 scramble movements from the originally trained model. Furthermore, inadmissible heuristics did not significantly affect the optimal solution. To test this, checked whether the returned length of the solution exceeds the number of scramble moves for this example. The learned heuristic, $\bar{h}_2$, is marginally better than the best person made, $\bar{h}_4$ as shown in the results. During plotting the comparison, it was relatively accurate and expected it to

perform better. However, the results satisfied with the results given the limited training time and data.

5.2 Limitation

The obstacle faced in this research project is mainly required more time to train the bigger size of the epoch, and why using Google Collaboration, the GPU usage is close to limit, for executing the heuristic models.

5.3 Future Research

5.3.1 Improve Learning Model

This project was very timely and computational and was the first real experience with the use of neural web frameworks. Finally, believe that further architecture and tweaking of hyperparameters will improve the performance of models.

This research method for generating examples is over-simplistic. There is no guarantee that a cube that was scrambled with, say, 25 moves is not solvable in less than that. A more intricate solution to this would be to store look-up tables and replace long sub-sequences of moves with shorter equivalent ones, if possible. As it is now, the cube's orientation is fixed. It is possible to augment the generated data by first permuting the cube, and then scrambling it. This process may help the learner extract some valuable features.

5.3.2 Benchmark on other Heuristic Models

The heuristic that has compared to those in this research. There are well-researched heuristics that can compete more effectively with the improved learning heuristics Korf's Algorithm (Harpreet Kaur, 2015), which can be used to improve the learned heuristic.

Kociemba's algorithm is an improved version of Thistlethwaite's algorithm that works by isolating the problem into only two subproblems. This method applies a propelled execution of IDA* search to be able to calculate, expel symmetries from the inquiry tree, and tackle the block near the briefest number of potential outcomes

NEURAL RUBIK's -THESIS FINAL

ORIGINALITY REPORT

18%	12%	4%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	pergamos.lib.uoa.gr Internet Source	2%
2	www.cse.iitk.ac.in Internet Source	2%
3	www.hpcwire.com Internet Source	2%
4	citeseerx.ist.psu.edu Internet Source	2%
5	Submitted to Mahindra United World College of India Student Paper	1%
6	H. Gunes, M. Piccardi. "Fusing face and body gesture for machine recognition of emotions", ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005., 2005 Publication	1%
7	machinelearningmastery.com Internet Source	1%

8	towardsdatascience.com Internet Source	1%
9	Submitted to UI, Springfield Student Paper	1%
10	Submitted to Arab Open University Student Paper	1%
11	en.wikipedia.org Internet Source	1%
12	Submitted to University of East London Student Paper	<1%
13	Submitted to Ridgeview High School Student Paper	<1%
14	vitalas.ercim.eu Internet Source	<1%
15	"Hybrid Metaheuristic Algorithms in Geotechnical Engineering", Modeling and Optimization in Science and Technologies, 2016. Publication	<1%
16	Submitted to South Dakota Board of Regents Student Paper	<1%
17	Submitted to Sim University Student Paper	<1%
	www.beingdexter.com	

18	Internet Source	<1%
19	scholar.uwindsor.ca Internet Source	<1%
20	Marco Fanfani, Fabio Bellavia, Carlo Colombo. "Accurate keyframe selection and keypoint tracking for robust visual odometry", Machine Vision and Applications, 2016 Publication	<1%
21	S Hidayat, Tulus, P Sirait. "Weighting Optimization of Decision Matrix in Fuzzy TOPSIS Using SMARTER Method", Journal of Physics: Conference Series, 2019 Publication	<1%
22	Submitted to iGroup Student Paper	<1%
23	Submitted to University of Strathclyde Student Paper	<1%
24	www.ukessays.com Internet Source	<1%
25	kb.psu.ac.th Internet Source	<1%
26	brage.bibsys.no Internet Source	<1%
	Cheng, . "Location of critical failure surface,	

27	convergence and advanced formulations", Slope Stability Analysis and Stabilization, 2014. Publication	<1%
28	Submitted to Xavier School Student Paper	<1%
29	Submitted to Northport High School Student Paper	<1%
30	Abhay Kumar Singh, Rajendra Sahu, Shalini Bharadwaj. "Portfolio evaluation using OWA-heuristic algorithm and data envelopment analysis", The Journal of Risk Finance, 2010 Publication	<1%
31	www.advancedlab.pe.kr Internet Source	<1%
32	Submitted to Saint Joseph's Institution, Singapore Student Paper	<1%
33	soar.wichita.edu Internet Source	<1%
34	Submitted to University of Sheffield Student Paper	<1%
35	Submitted to University of Warwick Student Paper	<1%
36	www.cse.buffalo.edu Internet Source	<1%

37	Saiyma Fatima Raza, Vishal Satpute. "A novel bit permutation-based image encryption algorithm", Nonlinear Dynamics, 2018 Publication	<1%
38	Submitted to Nazarbayev Intellectual School Student Paper	<1%
39	Submitted to University of Wales Swansea Student Paper	<1%
40	Submitted to Southridge High School Student Paper	<1%
41	Submitted to Australian International School Student Paper	<1%
42	Submitted to Queen Mary and Westfield College Student Paper	<1%
43	Submitted to Newbury Park High School Student Paper	<1%
44	Submitted to Island School Student Paper	<1%
45	Aleš Prokop, Seth Michelson. "Systems Biology in Biotech & Pharma", Springer Nature, 2012 Publication	<1%

Exclude quotes	Off	Exclude matches	Off

Exclude bibliography Off