Explorations into Algorithms

Submitted by

M Elamparithy
(Roll No: M20M19)

In partial fulfillment of the requirements for the award of Master of Science in Computer Science with Specialization in Machine Intelligence

of

Cochin University of Science and Technology, Kochi

Conducted by

Indian Institute of Information Technology and Management-Kerala
Technopark Campus
Thiruvanathapuram-695 581

April 2019

BONAFIDE CERTIFICATE

This is to certify that the project report entitled "Explorations into Algorithms" submitted by M Elamparithy (Roll No: M20M19) in partial fulfillment of the requirements for the award of Master of Science in Computer Science with Specialization in Machine Intelligence is a bonafide record of the work carried out at Indian Institute of Information Technology and Management under our supervision.

Supervisor	Course Coordinator
NAME DESIGNATION IIITM-K	NAME DESIGNATION IIITM-K

ORGANIZATION CERTIFICATE

To be Issued from company or institute on completion of project. (This is a dummy page)

.

DECLARATION

I, M Elamparithy, student of Masters of Science in Computer Science with specialization in Machine Intelligence, hereby declare that this report is substantially the result of my own work , except where explicitly indicated in the text, and has been carried out during the period March 2021 to June 2021.

Place: Thiruvananthapuram
Date: 4-7-2021

ACKNOWLEDGEMENT

I express my deep sense of gratitude to Dr. Alex P. James, Associate Dean (Academic), Professor for his guidance and support all through the project. His comments and feedback were immense and apt to improve my understanding and explore the area. I express my gratitude to the course coordinator, Dr. Asharaf S for his guidance. I thank Ms.Aswani A.R., Research Intern in the Department for organising the presentation meetings and coordinating our presentations. I also thank my parents and friends for their encouragement.

ABSTRACT

Algorithms are the processes through which problems are explored and solved. This project attempts to understand algorithms, particularly the Artificial Neural Network. Initially it explores the research in algorithms.Later complexity of the artificial neural network is derived.It also attempts to tinker with neural networks.The experiments conclude that the splitting of neural networks is much more effective in simple data sets when compared to one big neural network.

Table of Contents

List of Tables **viii**

List of Figures **ix**

1 Introduction **1**

2 Literature Review **2**
2.1 Reinforcement Learning 2
2.1.1 Objective 2
2.1.2 Findings 2
2.2 Comparing AlphaZero and Stockfish 4
2.2.1 Objective 4
2.2.2 Findings 4
2.3 Understanding How AlphaZero Works 5
2.3.1 Objective 5
2.3.2 Findings 5
2.4 Complexity of Artificial Neural Networks 7
2.4.1 Objective 7
2.4.2 Findings 7

3 Derivations and experiments **8**
3.1 Deriving the complexity of a neural network 8
3.1.1 Objective 8
3.1.2 Findings 9
3.2 Deriving the complexity of the parallel implementation of a neural network 10
3.2.1 Objective 10
3.2.2 Findings 11
3.3 Splitting Neural Networks 12
3.3.1 Objective 12
3.3.2 Findings 12

4 Conclusion **15**

REFERENCES **16**

List of Tables

3.1 results 14

List of Figures

2.1 chess engine 4
2.2 self play 5
2.3 Value Network 5
2.4 Policy Network 6
2.5 Architecture 6

3.1 General ANN 8
3.2 Matrix Multiplication during forward pass 9
3.3 Parallel implementation of ANN with m processors 10
3.4 A single processor 10
3.5 MNIST Datapoint 12
3.6 horizontal split 13
3.7 block split 13
3.8 splitting the dataset 14
3.9 feeding the split datasets into smaller networks 14

Chapter 1

Introduction

The main objective of this mini project was to get acquainted with research processes involved in understanding and creating algorithms. It has been carried out through a series of presentations, discussions and feedback on the presentations with the supervisor, Dr. Alex P. James.The feedback received from the previous presentations were incorporated in the subsequent presentations.The following algorithms have been explored in the presentations:

1. Forward and Backward pass in Artificial Neural Networks
2. Monte Carlo Tree Search
3. Minimax Algorithms
4. Reinforcement learning algorithms

This report presents all the eight presentations of which first four are explore the existing research in the field through which an understanding has been developed(by the student).The next four present the derivations and experiments carried out based on the learning and feedback from the supervisor.The eighth presentation includes codes developed by the student.

Chapter 2

Literature Review

2.1 Reinforcement Learning

2.1.1 Objective

To learn and present the fundamental concepts of Reinforcement Learning.

2.1.2 Findings

Reinforcement Learning involves a class of problems and algorithms.The algorithms map situations to actions.*Good* actions are reinforced.*Bad* actions are discouraged with punishments.After many iterations, the agent learns optimal actions.
The fundamental concepts of reinforcement learning involve-

1. Terminology: Agent, Environment, Rewards, State
2. Feedback: Rewards and Punishment
3. Formulating the problem: Markov Decision Processes

In a reinforcement learning problem an *agent* takes actions in an *environment* which emits an observation in return.The *reward* is given based on the observation. An action is performed based on some representation of the environment i.e *state*.
The job of a Reinforcement Learning Algorithm is to reward the agent or punish the agent based of it actions to achieve a particular state in the environment.

Markov decision processes formally describe an environment for reinforcement learning. A Markov Decision Process is a tuple $\langle S, A, P, R, \gamma \rangle$

1. S is a finite set of states.
2. A is a finite set of actions
3. P is a state transition probability matrix

$$P^a_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a] \tag{2.1}$$

4. R is a reward function

$$R^a_s = \mathbb{P}[R_{t+1} | S_t = s, A_t = a] \tag{2.2}$$

5. γ is a discount factor, $\gamma \in [0, 1]$.

AlphaZero is a chess engine which uses deep reinforcement learning to play chess and has been consistently better than the classical chess engine like stockfish.
[1] was used to understand and develop a basic understanding in reinforcement learning.

2.2 Comparing AlphaZero and Stockfish

2.2.1 Objective

To find out the differences in the implementation of the chess engines AlphaZero and Stockfish.

2.2.2 Findings

In a given chess position, a chess engine searches for the possible moves and evaluates the position.

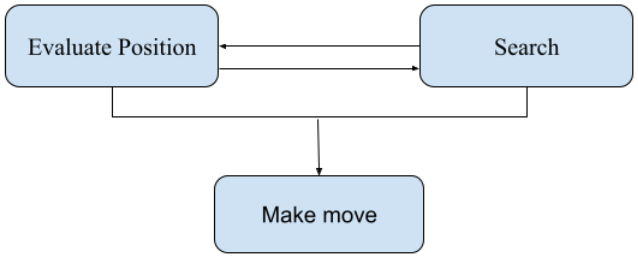

Figure 2.1: chess engine

Both Stockfish and AlphaZero search for moves and evaluate the possible postions, but they do it it in a completely different way.
Stockfish

1. Uses a fixed depth *Minimax* Algorithm with *Alpha-beta pruning* [2].
2. Uses a predefined evaluation function which takes in inputs like Material, Piece-Square Tables, Pawn Structure etc [2].

AlphaZero

1. Uses a neural network for evaluation function and possible moves [3].
2. Monte Carlo Tree Search for cross checking the prediction of the neural network [3].

2.3 Understanding How AlphaZero Works

2.3.1 Objective

To understand in detail how AlphaZero uses deep reinforcement learning to predict chess moves.

2.3.2 Findings

AlphaZero has 2 components-

Value Network

After a round of self play, AlphaZero labels the each move with the winning players label.

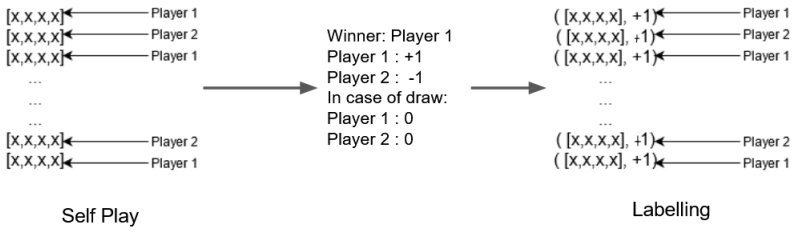

Figure 2.2: self play

The value network is trained by the labeled moves obtained from self play. The network makes a prediction on which player is going to win for each move played during self play.And this prediction is tested against the actual results obtained.

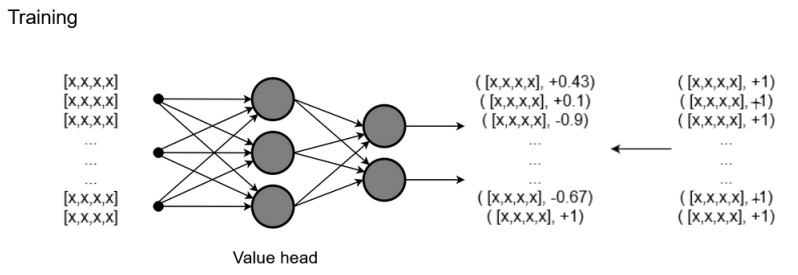

Figure 2.3: Value Network

Policy Network

The policy network predicts possible moves which are worth playing

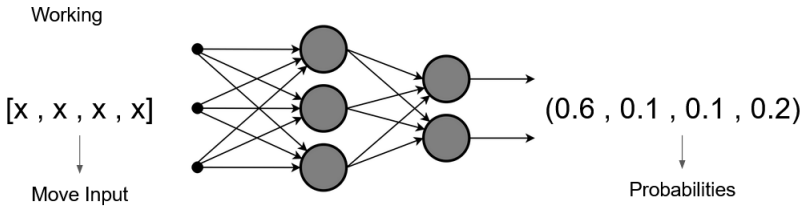

Figure 2.4: Policy Network

Monte Carlo Tree Search (MCTS) is used to roll out the predictions made by the policy network [3]. Using MCTS helps improve the predictions made by the policy network.
The value and policy networks are fed the simplified input of the game state. Given below is the architecture of the Neural Network in AlphaZero [3].

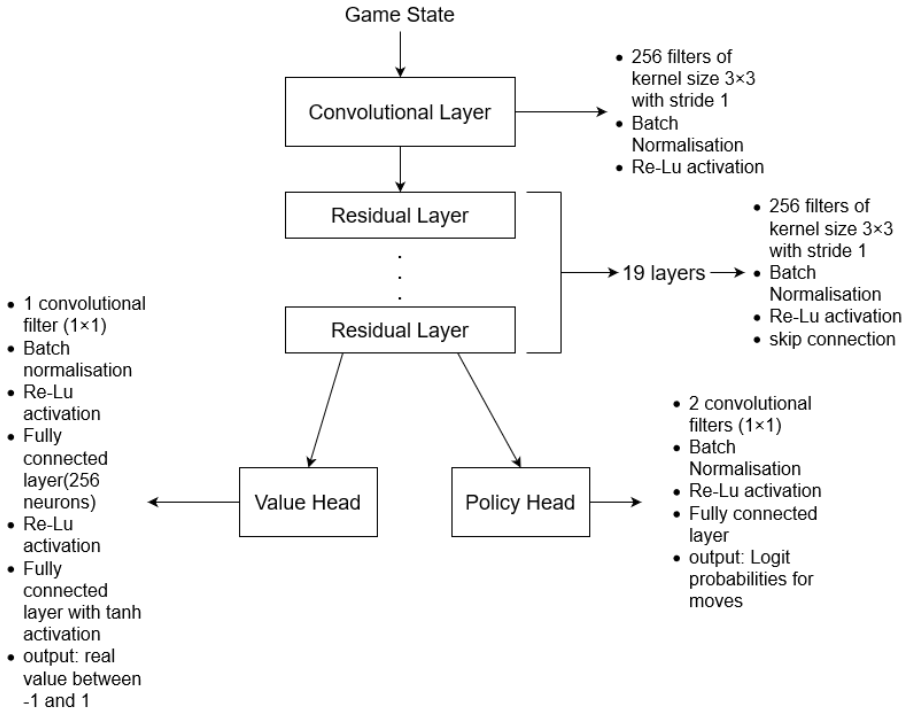

Figure 2.5: Architecture

2.4 Complexity of Artificial Neural Networks

2.4.1 Objective

To find the time complexity of AlphaZero.

2.4.2 Findings

AlphaZero uses a convolutional neural network and Monte Carlo Tree Search to make its moves.

Convoultion

For a M×N input matrix and m×n filter,the output matrix will have O(mn) computations for each entry.So the 2D convolution would have an approximate complexity of O(MNmn)

Monte Carlo Tree Search

The runtime of the algorithm can be simply be computed as *O(mkl)*, where m is the number of random children to consider per search k is the number of parallel searches l is the number of iterations.

Forward and Backward pass

Discussed in the section.

Chapter 3

Derivations and experiments

3.1 Deriving the complexity of a neural network

3.1.1 Objective

To calculate the number of computations performed in the forward and backward pass of the neural network given below.

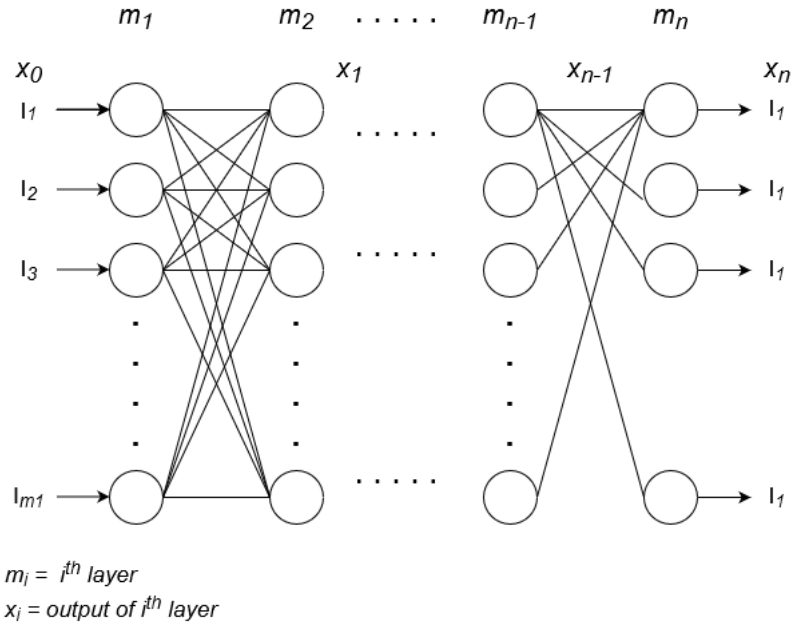

Figure 3.1: General ANN

3.1.2 Findings

Forward pass between two layers can be represented as the product of weight matrix and the input of the next layer.

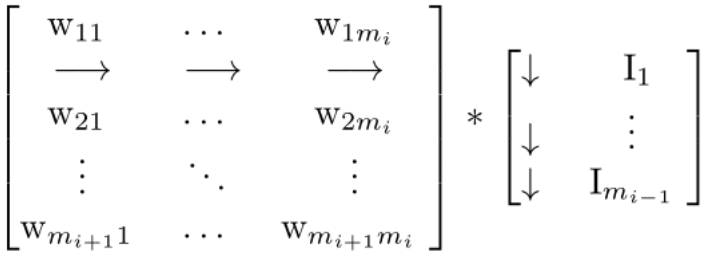

Figure 3.2: Matrix Multiplication during forward pass

So, in total, the number of operations $= T = m_1m_2 + m_1m_2 + \ldots + m_{n-1}m_n$. This means $T \leq N^3$, where $N = Max(m_i, n)$ So, the upper limit of the number of computations in forward pass is n^3.

Backward Pass consists of back propagation and updating weights. Back propagation involves matrix multiplication followed by a component wise multiplication:

$$W^{l+1} \times \delta^{l+1} \times \sigma'(z^l) \tag{3.1}$$

where W^{l+1} is the weight matrix, δ^{l+1} is the error of layer $l+1$ and $\sigma'(z^l)$ is the derivative of the output of layer l.
This gives us the upper limit of the number of computations in backward pass as n^4.

3.2 Deriving the complexity of the parallel implementation of a neural network

3.2.1 Objective

To calculate the number of computations performed in the forward and backward pass of the parallel implementation of the neural network given below [4].

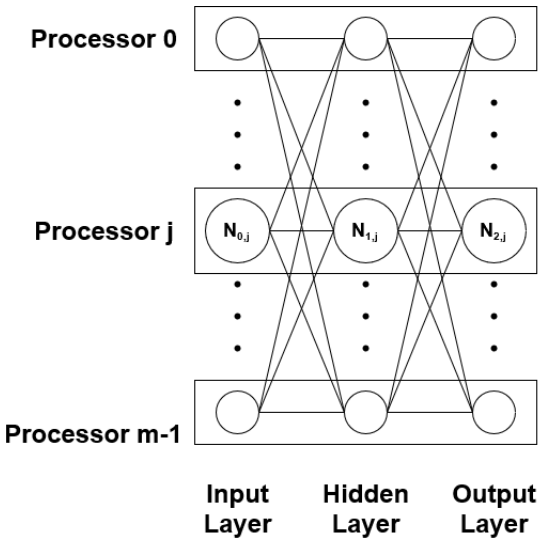

Figure 3.3: Parallel implementation of ANN with m processors

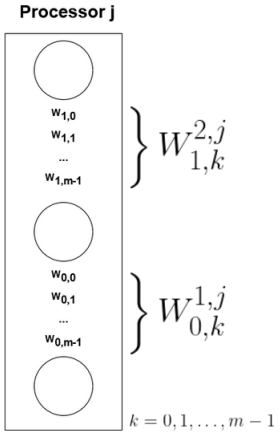

Figure 3.4: A single processor

3.2.2 Findings

Forward Pass

1. 'm' multiplications are happening at the same time
2. 'm' rotation happen in total
3. Thus, it takes 'm' steps for a forward pass from one layer to another [4].

Back propagation

1. Weight is multiplied with the error from the node
2. Rotation of error and repeat
3. Thus, it takes 'm' steps for one layer [4].

Updating weights

1. Rotate input, compute weight change
2. add weight change and Repeat
3. Thus, 'm' steps per layer [4].

3.3 Splitting Neural Networks

3.3.1 Objective

To use multiple smaller neural networks instead of one big network to make predictions.

3.3.2 Findings

The following needs to be done-

1. Splitting each datapoint the dataset.
2. Creating smaller networks so that each network gets on part of the split datapoint.
3. combining the output of smaller networks to make prediction.

The MNIST dataset has been used here. A single data point looks like-

```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   3  18  18  18 126 136 175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253 225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251  93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  35 241 225 160 108   1   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119  25   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253 150  27   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252 253 187   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249 253 249  64   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253 253 207   2   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253 250 182   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201  78   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 136 253 253 253 212 135 132  16   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]]
```

Figure 3.5: MNIST Datapoint

We split the datapoint in two ways-

```
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 3 18 18 18 126 136 175 26 166 255 247 127 0 0 0 0]
 [0 0 0 0 0 0 0 0 30 36 94 154 170 253 253 253 253 253 225 172 253 242 195 64 0 0 0 0]

 [0 0 0 0 0 0 0 49 238 253 253 253 253 253 253 253 253 251 93 82 82 56 39 0 0 0 0 0]
 [0 0 0 0 0 0 0 18 219 253 253 253 253 253 198 182 247 241 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 80 156 107 253 253 205 11 0 43 154 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 14 1 154 253 90 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 139 253 190 2 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 11 190 253 70 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 35 241 225 160 108 1 0 0 0 0 0 0 0 0 0 0]

 [0 0 0 0 0 0 0 0 0 0 0 0 0 81 240 253 253 119 25 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 45 186 253 253 150 27 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 16 93 252 253 187 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 249 253 249 64 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 46 130 183 253 253 207 2 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 39 148 229 253 253 253 250 182 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 24 114 221 253 253 253 253 201 78 0 0 0 0 0 0 0 0 0]

 [0 0 0 0 0 0 0 0 23 66 213 253 253 253 253 198 81 2 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 18 171 219 253 253 253 253 195 80 9 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 55 172 226 253 253 253 253 244 133 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 136 253 253 253 212 135 132 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
```

Figure 3.6: horizontal split

```
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0        0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0        0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0        0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0        0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0        0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 3 18        18 18 126 136 175 26 166 255 247 127 0 0 0 0]
 [0 0 0 0 0 0 0 0 30 36 94 154 170 253        253 253 253 253 225 172 253 242 195 64 0 0 0 0]
 [0 0 0 0 0 0 0 49 238 253 253 253 253 253        253 253 253 251 93 82 82 56 39 0 0 0 0 0]
 [0 0 0 0 0 0 0 18 219 253 253 253 253 253        198 182 247 241 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 80 156 107 253 253 205        11 0 43 154 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 14 1 154 253 90        0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 139 253 190        2 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 11 190 253        70 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 35 241        225 160 108 1 0 0 0 0 0 0 0 0 0 0]

 [0 0 0 0 0 0 0 0 0 0 0 0 0 81        240 253 253 119 25 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0        45 186 253 253 150 27 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0        0 16 93 252 253 187 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0        0 0 0 249 253 249 64 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0        46 130 183 253 253 207 2 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 39 148        229 253 253 253 250 182 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 24 114 221 253        253 253 253 201 78 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 23 66 213 253 253 253        253 198 81 2 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 18 171 219 253 253 253 253 195        80 9 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 55 172 226 253 253 253 253 244 133 11        0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 136 253 253 253 212 135 132 16 0 0        0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0        0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0        0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0        0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
```

Figure 3.7: block split

For The neural network, we have a single template so that we can create multiple networks with different hyper parameters.

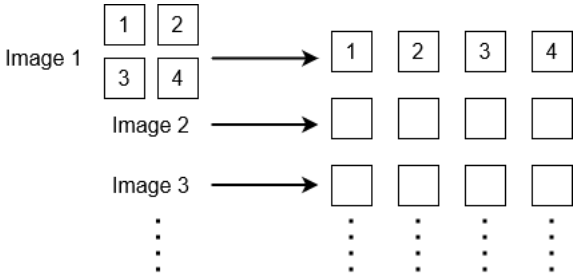

Figure 3.8: splitting the dataset

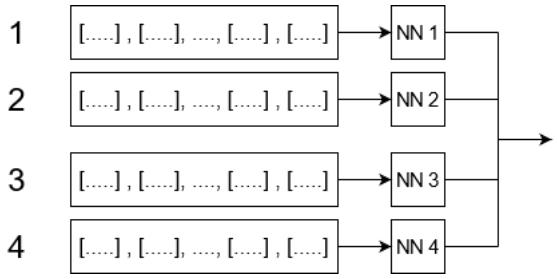

Figure 3.9: feeding the split datasets into smaller networks

The outputs of the four networks are combined by taking a vote.

Results

Info	Time taken (for 10 epochs) (s)	Accuracy (%)
Single Network	31	97.9
Smaller Network (horizontal split)	2 (for one network)	97.5
Smaller Network (block split)	2 (for one network)	98.3

Table 3.1: results

Chapter 4

Conclusion

This research began with the aim to understand algorithms through the process of review of research and explorations into some of the algorithms *viz.* AlphaZero, Back propagation. Further, time complexity of the artificial neural network and its parallel implementation was explored.This was followed by experimenting the splitting of neural networks as shown in the last presentation. The learning, in sum, is the splitting of neural networks gives accurate results in much less time for simple data sets when compared to one big neural network. It would be appropriate and the interest of the student to further work on splitting of neural networks with complex data sets and splitting the data in complex ways.

REFERENCES

[1] Reinforcement Learning Course by David Silver, Deep Mind, 2015 `https://www.davidsilver.uk/teaching/`

[2] Tord Romstad, Marco Costalba, Joona Kiiski, et al. Stockfish: A strong open source chessengine. `https://stockfishchess.org/`

[3] David Silver,Thomas Hubert,Julian Schrittwieser. *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm.* arXiv:1712.01815 [cs.AI], 2017.

[4] Xiru Zhang, Michael Mckenna, Jill P. Mesirov, David L. Waltz *An Efficient Implementation of the Back-propagation Algorithm.* NIPS, 1989.