ANALYSIS OF TIME-DELAY ARTIFICIAL NEURAL NETWORKS IN BALL CATCHING TASK

An Undergraduate Research Scholars Thesis

by

CASSANDRA BUB

Submitted to the Undergraduate Research Scholars program at
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by Research Advisor: Dr. Yoonsuck Choe

May 2017

Major: Computer Science

TABLE OF CONTENTS

ABSTRACT

Analysis of Time-Delay Artificial Neural Networks in Ball Catching Task

Cassandra Bub
Department of Computer Science and Engineering
Texas A&M University

Research Advisor: Dr. Yoonsuck Choe
Department of Computer Science and Engineering
Texas A&M University

In this paper, we look at the performance of a time-delay neural network in a scenario requiring memory as well as reactivity. Utilizing a ball catching scenario where the agent will have to move to catch a falling ball, and then remembering where the second one was relative to its position in order to catch the second, we can determine how the time-delay neural networks perform in these tasks. For comparison to previous work with this scenario, we will compare the performance to a feed-forward network and a recurrent neural network.

CHAPTER I

INTRODUCTION

Neural networks have been a popular machine learning technique, but lately there has been an explosion of new methods that dramatically improve the performance, helping it to dominate the field [6]. Some of the ways that this has been studied is done by utilizing different constructs available to neural networks, including recurrent loops [2] and environmental markers [1]. Often, these additions proved to be more effective than a simple feed forward network. Yet another construct available to these networks is time delay lines [7] used to create a time-delay neural network, the network first introduced in regards to phoneme recognition [4].

In this thesis, we verify whether a time-delay artificial neural network can successfully learn and perform given in a task requiring memory. The results will be compared to those of recurrent neural networks and feed forward networks.

CHAPTER II

METHODS

To determine the level of success of the different networks tested they will be evaluated in a ball-catching scenario, the same one used in [1]. An agent acting to catch the balls is controlled by the artificial neural network, and is allowed to move horizontally based on two output neurons with a number of sensors, in our case five, projecting out to detect the falling balls. The agent is initially placed in the center of the area, and two balls are randomly placed at the top of the screen, which may or may not be sensed by the agent depending on their relative location. One ball, again randomly selected, will be falling at a velocity faster than the other, such that the ideal response of the agent is to determine which ball is falling faster and move to catch that one first, remember where the second slower ball was located and move back to catch it. The situation described is shown in a diagram from [1], replicated in Figure 1 below.

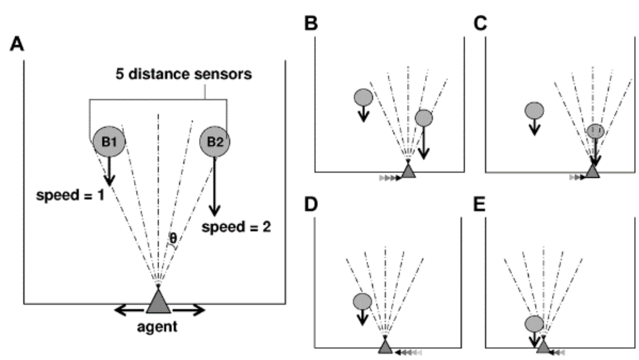

Figure 1. The ball catching scenario is shown above with the two balls and one agent. An ideal movement pattern is pictured, catching B2 first, then moving back to catch B1.

The three different types of neural networks that will be tested are (1) feedforward (FFNN), (2) recurrent (RNN), and a (3) time-delay neural network (TDNN). (1) The feedforward neural network, shown in Figure 2, is composed of five input nodes formed from the five sensors of the agent, a hidden layer of five nodes to compute the internal representation, and two output nodes that will tell the agent to move either left or right.

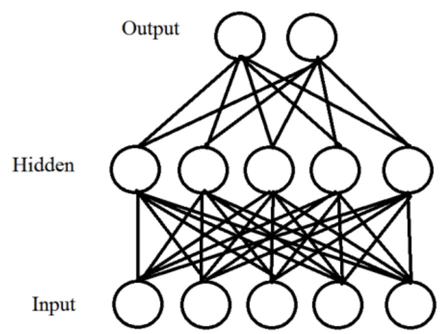

Figure 2. The internal structure of the feedforward neural network used.

(2) The recurrent network, seen in Figure 3, was implemented with a similar set up, however, there are also context nodes that allow the network to take into account past events and therefore have a “memory”. Specifically, we implemented Elman’s network [2], that adds the previous hidden layer activation to the next round’s input to allow past inputs to be taken into account.

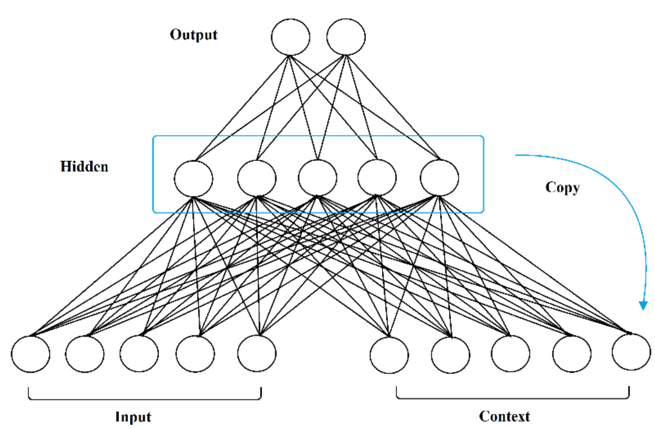

Figure 3. An illustration of Elman’s recurrent neural network.

(3) The time-delay neural network, seen in Figure 4, approaches memory differently from the recurrent network by feeding past inputs to the hidden layer, also known as delay lines [7]. It will map a finite sequence of these inputs into the hidden layer in addition to the current input from the sensor, and each input will be measured from various points in time and allow the network to compare previous inputs to current, all within a feedforward network.

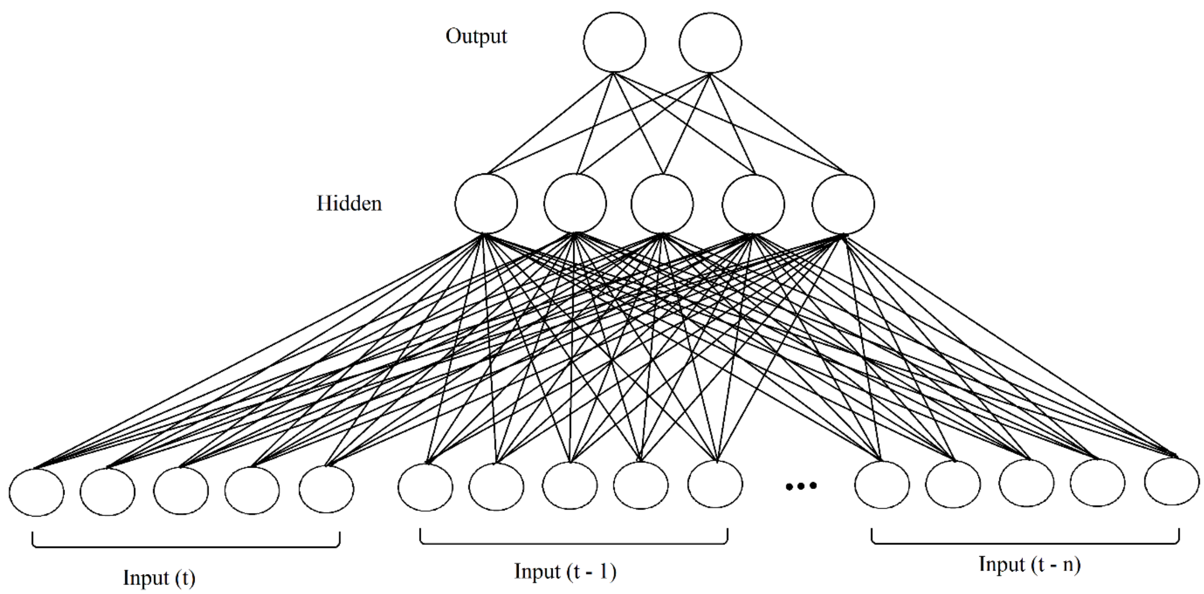

Figure 4. An illustration of a time-delay neural network with the delayed inputs fed into the hidden layer.

These three networks were trained using a genetic algorithm, with a one-point crossover with probability 0.9 and a 0.015 mutation rate. For each of the 50 generations there were 20 individuals each given 30 trials of the ball catching scenario, where the top 4 out of these 20 individuals was used in the reproduction for the next generation. The fitness of these individuals during the generation was based on the number of times it successfully caught both balls in each trial, with the final fitness being evaluated by the sum of all 30 trials. The halting criterion was catching both balls in every trial, which was never reached. The performance reported seen in the results section are based on being able to catch both balls in the scenario.

CHAPTER III

RESULTS

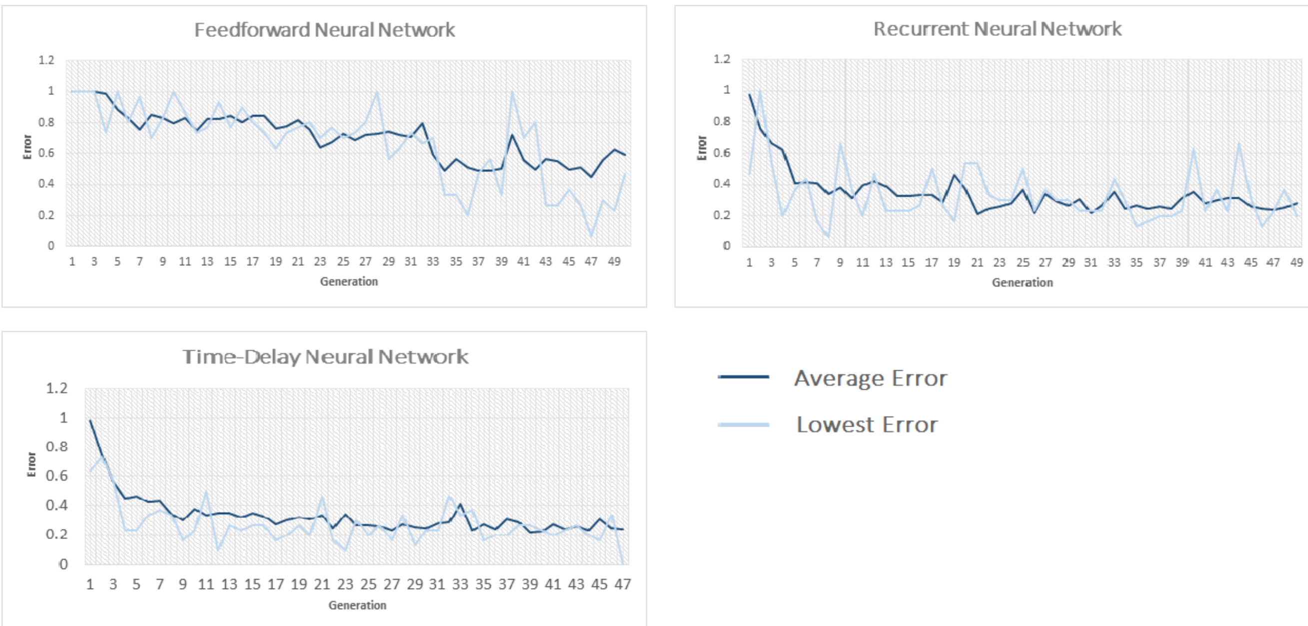

Figure 5. The progression of the networks during training.

While the feedforward neural network never got to an error rate of less than 50% and averaged out at 56% error, the time-delay and recurrent networks were able to have much lower error rates. This was expected behavior for the feedforward network as it had no internal representation for past inputs, no mechanism for memory. Therefore, the chances that it would move to catch the correct ball depended on whether it began to move in the correct direction, out of the two choices left or right, and whether or not it stayed moving in that direction until the sensors began to pick up on the ball again.

The time-delay and recurrent networks performed considerably better and were almost equal to each other with the recurrent network performing slightly better. These increasingly better rates were due to the fact that these two networks had a way of analyzing past inputs. Even though they both used different forms of memory, both were able to show competence at the ball catching scenario, utilizing their constructs for analyzing past inputs.

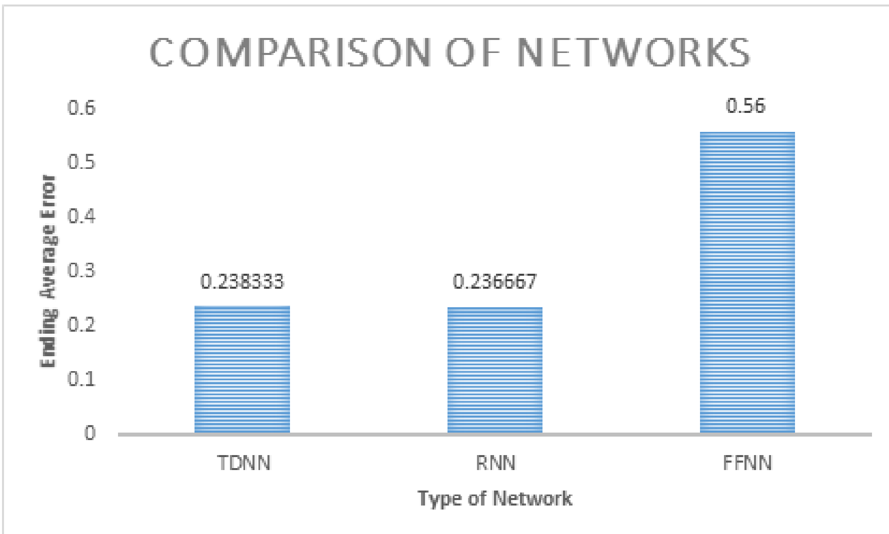

Figure 6. Chart showing a comparison of performance in the average error of the different types of networks tested.

However, it was interesting to note that in the time-delay neural network the more delay lines, or rather the number of delayed inputs, the worse the performance was, with the best performance with two delay lines, as seen in Figure 7. This network with two delay lines was the time-delay network used in comparison against the others, and the one shown in Figure 5 and 6. A hypothesis on why more delay lines created worse performance could be that the network would pay more attention to inputs that were no longer relevant, but this would require further testing to verify.

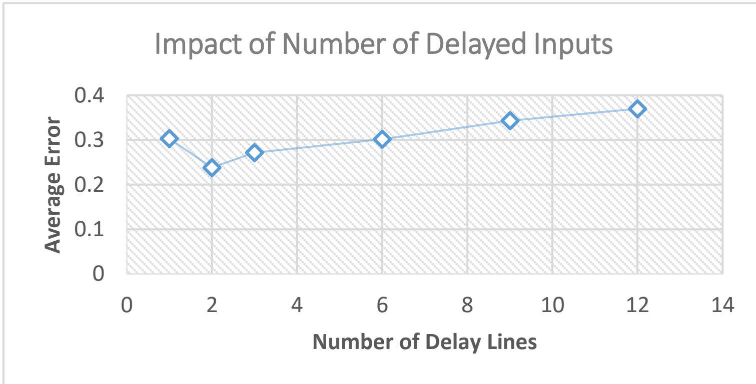

Figure 7. Chart illustrating the impact that the number of delay lines had on performance in the time-delay neural network (TDNN).

CHAPTER IV

CONCLUSION

The memory capabilities of the time-delay neural network with delay lines seem to be able to reach equal performance measures as Elman's recurrent neural network. Specifically, in this ball-catching scenario the recurrent neural network performed slightly better. This goes to show that even with different constructions of memory in artificial neural networks we are able to see comparable performance, and that even simple feedforward networks, if inputs are taken from a short time window in the past, can perform well in such memory tasks.

The more delay lines in the time-delay neural network the worse the performance was. Surprisingly, the most effective network had only two previous inputs fed into it, and after this the performance declined in a linear fashion. Further testing could help us understand why this is the case. Is the network putting too much emphasis on past input unnecessarily? Would more time in training give a different result? For now, we know that more delay lines does not mean better.

REFERENCES

[1] Ji Ryang Chung and Yoonsuck Choe. *Emergence of memory in reactive agents equipped with environmental markers.* IEEE Transactions on Autonomous Mental Development, 3:257-271, 2011.

[2] J. L. Elman, “*Finding structure in time,*” Cogn. Sci., vol. 14, pp. 179–211, 1990.

[3] D. Svozil, V. KvasniEka, J. Pospichal, *Tutorial introduction to multi-layer feedforward neural networks,* Chemometr. Intell. Lab. Syst. (1997) 43–62.

[4] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, “*Phoneme recognition using time-delay neural networks,*” IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 37, no. 3, pp. 328–339, Mar. 1989.

[5] Caamaño, P., Salgado, R., Bellas, F. et al. *Introducing Synaptic Delays in the NEAT Algorithm to Improve Modelling in Cognitive Robotics*, Neural Process Lett (2016) 43: 479. doi:10.1007/s11063-015-9426-5

[6] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press, 2016.

[7] T. Liu, S. Bounmaiza, F. M. Ghannouchi, "*Dynamic Behavioral Modeling of 3G Power Amplifiers Using Real-Valued Time-Delay Neural Networks*", IEEE Transactions on Microwave Theory and Techniques, vol. 52, no. 3, pp. 1025-1033, 2004.