**Application Of Neural Networks To Robot Animals**

**Final Project CS547**

Matthew A. Letter

&

Lin sun

University of New Mexico

Neural Networks 547

mletter1@unm.edu

sun@unm.edu

**Abstract: left blank intentionally**

Table of Contents

Abstract 3

1. Introduction 4

2. Approach 4

2.1 Architecture 0: No movement, measure lifetime 4

2.2 Architecture 1: Movement, measure lifetime as a function of speed without eating food 4

2.3 Architecture 2: Movement, measure lifetime as a function of speed with eating food 4

2.4 Architecture 3: Neuronal Classification of food using Food RGB Values 5

2.5 Architecture 4: Neuronal Movement towards brightest object 5

2.6 Architecture 5: Eat Objects based on Neuronal Classification using RGB Values 7

2.7 Architecture 5: Neuronal Classification of food using Food RGB Percentage 7

2.6 Architecture 5: Eat Objects based on Neuronal Classification using RGB Percentage 7

3. Results 7

3.1 Architecture 0: No movement, measure lifetime 7

3.2 Architecture 1: Movement, measure lifetime as a function of speed without eating food 8

3.3 Architecture 2: Movement, measure lifetime as a function of speed with eating food 9

3.6 Architecture 5: Eat objects based off neuronal classification 11

4. Discussion 11

5. Summary and Conclusion 11

6. Acknowledgements 11

7. References 11

8. Appendix A 11

# Abstract

Add abstract here, single space, 12pt intentionally blank

# 1. Introduction

This paper rigorously establishes that neural network design models can be used to manipulate learning on simulated biological system. All the research was done using professor Thomas Caudell’s animal robot environmental model. (Caudell) The starting point of the research involved 3 architectures, used to establish the parameters of life with respect to the simulated organism and its environment. These provide a basis for analyzing the implemented neural network algorithms, the goal of which is to create a “neuron” based brain for the robot organism to live as long as possible in its environment, and determine what neuronal design structures degrade or enhance this attribute.

# 2. Approach

### 2.1 Architecture 0: No movement, measure lifetime

For the first architecture, we will test the basic metabolic consumption for the robot. So we just set the robot’s speed as 0 and do nothing until it dies. Then we record down how many cycles the neuron could survive for.

### 2.2 Architecture 1: Movement, measure lifetime as a function of speed without eating food

This test will try to explore the effect of robot speed on robot’s lifespan. So that we could find a speed where the speed change will not affect the final result. Here we will let neuron start at the same position, but in different direction. To try test different direction, we will turn the neuron head to 5 degree counter-clockwise each time. When the neuron is moving, it will always head to the brightest object in the space, no matter if the food is poisonous or not. Each time after the neuron finish moving, it will eat nothing in the space, so that the only variable in this test is the neuron speed.

### 2.3 Architecture 2: Movement, measure lifetime as a function of speed with eating food

This test will try to explore the effect of neuron speed on robot’s lifetime. However, it is different from Architecture 2 in that neuron will eat food it means each time. What is more, we will not let neuron to classify object this time. Just let it eats whatever it contacts. Therefore, we could know more about speed effect based on eating. When the neuron is moving, it will always head to the brightest object in the space, no matter if the food is poisonous. To get the speed effect randomly, we will change the initial neuron head angle each time, but keep the initial position the same as origin point.

### 2.4 Architecture 3: Neuronal Classification of food using Food RGB Values

This test will try to classify food as either good or bad based on whether the food could increase or decrease energy of the robot. If the food could increase the energy for the robot, we say that the food is good. Otherwise, we say that the food is bad. To be simple, we will use LMS neuron to classify the food first. In order to classify the objects, we should know the weights for classification neuron. So in this architecture, we will try to train our neuron first using the objects eaten in the world.



Figure 1: Network diagram of the LMS neuron for classifying food

### 2.5 Architecture 4: Neuronal Movement towards brightest object

Now that the robot knows how to classify objects in the world in order to eat it properly. But there is another question, how to switch the direction of the robot each time the neuron makes a movement. To give the robot the ability to seek out objects one more neuron will be added to make direction choice.

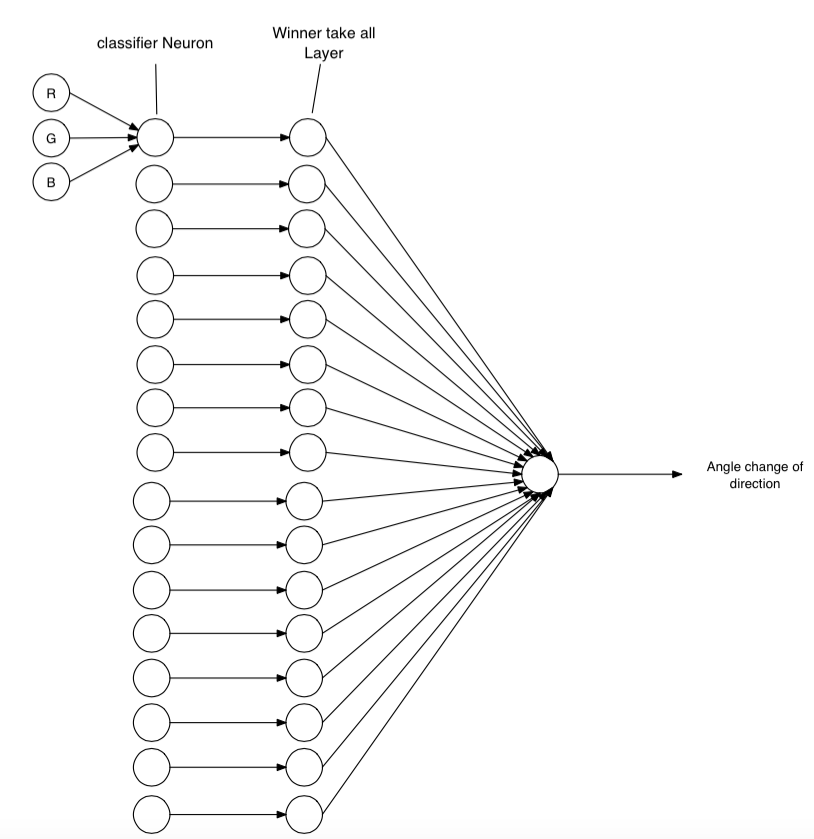
In this project, the neuron will choose the direction based on the reflection light intensity. It is known that the light intensity is inversely related to the distance between the robot and the object. It means that objects having higher light intensity are closer to the robot than other objects. In this simulation world, the robot wants to survive, therefore, it should try to grab the objects in the world closer to it. Eq 1.1 shows the equation to decide the direction. So based on the theory above, we design a neuron like Fig 3. The inputs for the input are RGB lights from 31 direction, while the output is the angle of the lightest object with the horizontal line. So the direction neuron will change the direction of the robot automatically based on light intensity.

d = argmax(sum(r^2 + g^2 + b^2)) (eq 1.1)



Figure 2: This is a three layer neural network where the RGB circle denotes the intensity signal bands being received by the first layer. The second layer is a winner-take all network which selects the brightest cone. The third layer is a neuron, which computes the angle change for movement

### 2.6 Architecture 5: Eat Objects based on Neuron Classification using RGB Values



### 2.7 Architecture 6: Neuronal Classification of food using Food RGB Percentage

In architecture 3, we use RGB values to classify objects. However, if robot is far away from the object, the light value is decreased for each single bandwidth. Therefore, it is not trustable to classify objects far away from the robot. So we want to see if normalized RGB values could generate the same result like original RGB values. That is what we will do in architecture 6.

### 2.8 Architecture 7: Eat Objects based on Neuron Classification using RGB Percentage

In architecture 6, we use normalized RGB value to classify the object. So we will choose the direction based on both intensity and poisonous. So we will choose the most intense direction, which has nonpoisonous objects.

### 2.9 Architecture 8: Classify objects based on visual lights and acoustic sounds:

# 3. Results

### 3.1 Architecture 0: No movement, measure lifetime

Table 1: Over 200 runs, every run of a stationary robot had a lifespan of 5001 cycles

|  |  |  |
| --- | --- | --- |
| Number of Runs | Lifespan of Robot | Standard Deviation |
| 200 | 5001 cycles | 0 |

From Table 1, we could see that the robot life is 5001 cycles if it does not move and does not eat any food in the space. From settings in the system, we could see that the total initial charge of 1. The robot needs one cycle to jump out the loop. So totally there should 5000 cycles to consume energy in the system. Therefore, we could see that the basic metabolic rate should be 0.0002 unit/cycle.

### 3.2 Architecture 1: Movement, measure lifetime as a function of speed without eating food



Figure 1: Shows the max speed of the robot to be 0.1



Figure 2: Linear function of the life cycles with respect to the Robots speed.

From Fig1, we could see that life time is related with robot speed. When the speed is low, the lifetime has a nearly linear function with speed. As speed increases, the lifetime is decreased. But when the speed is over 0.1, the lifetime is stabilized. So the speed will not change even though the speed is increasing. Besides, even though we change the head angle from the same origin, the lifetime for the same speed is still the same. Therefore, the standard deviation for lifetime at the same speed is the 0.

### 3.3 Architecture 2: Movement, measure lifetime as a function of speed with eating food



Figure 3: Error bar graph of varying speeds and directions

| Table 2 Robot Lifetime Mean and Standard Deviation for Different Speed Rate | | |
| --- | --- | --- |
| Speed Rate | Lifetime Mean(Cycles) | Lifetime Standard Deviation |
| 0.02 | 4541 | 884 |
| 0.04 | 4233 | 973 |
| 0.06 | 3517 | 849 |
| 0.08 | 3093 | 765 |
| 0.10 | 2861 | 832 |
| 0.12 | 2858 | 827 |
| 0.14 | 2858 | 827 |

Fig 3 shows the final result of lifetime vs speed while robot could eat food in the word. Like Architecture 2, the lifetime is relevant with speed when the speed is low. As speed increases, the lifetime is decreased. But when robot speed goes over 0.1, the lifetime is stabilized even though there is still standard deviation for the same speed with different initial head angle.

3.4 Architecture 3: Neuronal Classification of food

Fig 4 RMS vs lifetime training result

| Table … Classification Neuron Weights after training | | | | |
| --- | --- | --- | --- | --- |
| Weight Index | 0 | 1 | 2 | 3 |
| Value | -0.0593627 | -0.0574903 | 0.643285 | -0.0570887 |

Fig 4 shows training result for the classification neuron. We could see that at the beginning the RMS drops very quickly and dramatically. However, after a few cycles, the RMS values is stabilized but still tremble within 0.2 and 0.35. So we could see that the neuron is trained well after that. Table … shows the weight for the final trained neuron using RGB values. Therefore, in the following experience, we could use these weight to classify the objects robot meets and decide if robot should eat the objects or not.

3.5 Architecture 4: Neuronal Movement towards brightest object

In 2.5, we have shown the theory about how to make decision on direction for the neuron. However, we need to verify if the neuron algorithm we implement is correct or not. Fortunately, there is an internal implemented function called intensity\_winner\_takes\_all which returns the index of the receptor which got the most intensity. Then we will compare the result from our function and the internal function to see if they are the same. Fig … shows the result for both functions. We can see that the neuron implemented here generates the same result as the internal functions. Therefore, our function is right.

Need plot the data

Fig … Direction Decision Neuron Checking

### 3.6 Architecture 5: Eat objects based on neuronal classification

In this architecture, we will let neuron

# 4. Discussion

# 5. Summary and Conclusion

# 6. Acknowledgements

# 7. References

1. Caudell, Thomas. "Flatworld."
2. Haykin, Simon S. *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ: Prentice Hall, 1999. Print

# 8. Appendix A