Requirements : Pygame and numpy libraries

Code tricks.....

- you can control the frame rate by up and down arrows.
- After completion of every generation, the population was stored in the backup file(‘Snake game by GA.NPZ’), so when you rerun the code, it will run from the last generation.
- If you want see the final snake having maximum score, download the ‘Snake game by GA.NPZ’ file along with code which has last generated population.
- To start the generation from beginning, delete the ‘Snake game by GA.NPZ’ file

AI learns to play Snake using Genetic Algorithm

Hear, we will see the how Genetic algorithm in Neural Network can play the Snake game

- Alone Neural network can also be used to play the Snake but Neural Network needs a lots of training data.
- In Neural network weights are adjusted by backpropagation technique, wherein Genetic algorithm weights are adjusted manually.

Let’s see how we will use Genetic Algorithm in Neural Network

- 1st we will decide the Network network architecture,
- Then we create the fitness function to evaluate the snake,
- Then a random initial population was created,
- Then we play a game for each individual in the population and sort out each individual in the population based on the fitness function score,
- Then we find out the parents based on the top fitness from the population,
- Then we generate the remaining population (or) chromosomes from the parents using Crossover and Mutation,
- Then we combine the parents and chromosomes to form a new population and we will play the game for new population like in previous step and we will repeat the process until the Snake gets maximum score.

Deciding Neural network architecture

- Neural network consist of Input, Hidden and Output layers and the Activation functions and we have to find the neurons in each layer which will helps the snake in getting high score.

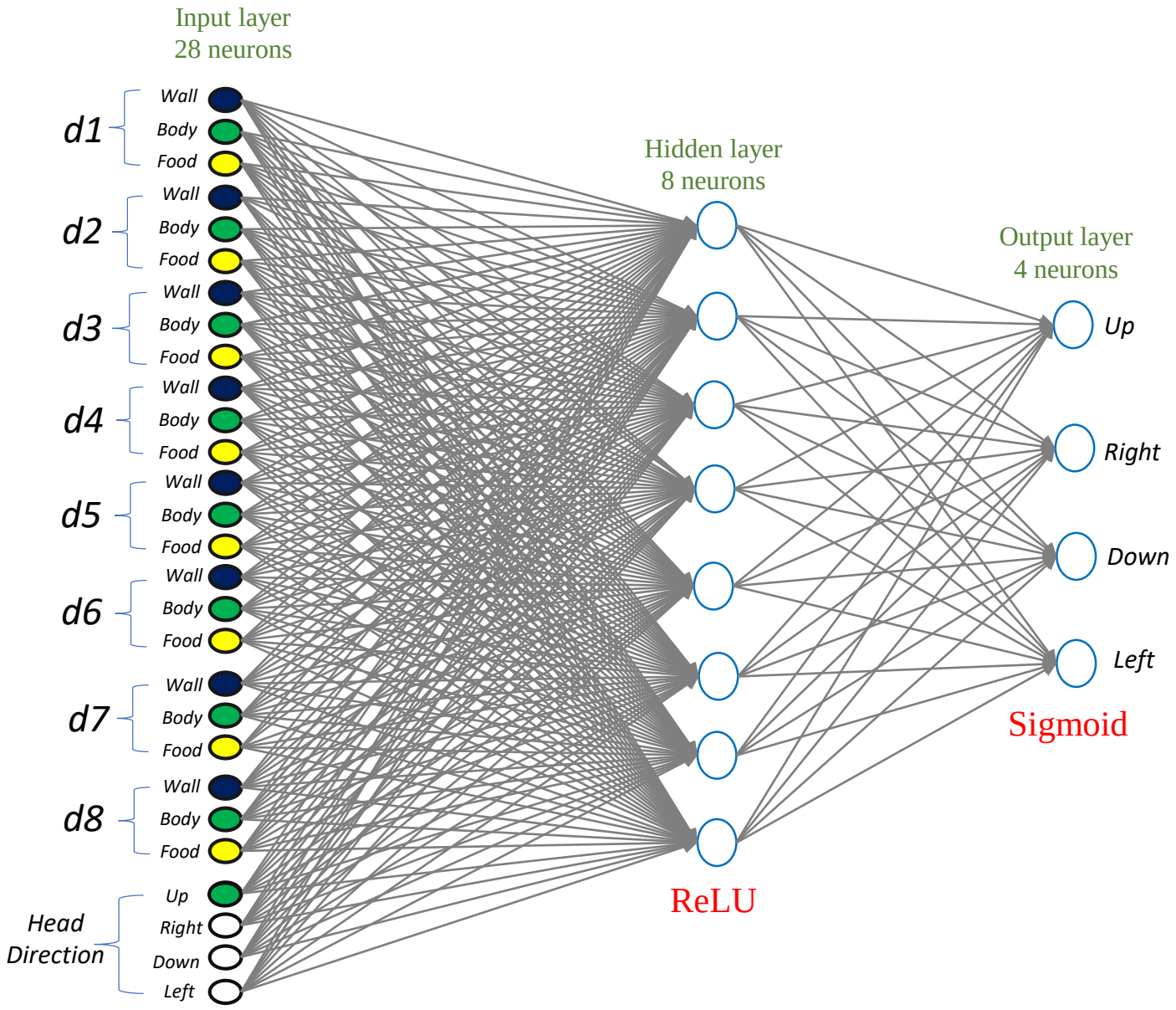
1st we will see Activation function, Hidden and Output layers and then we go for Input layer

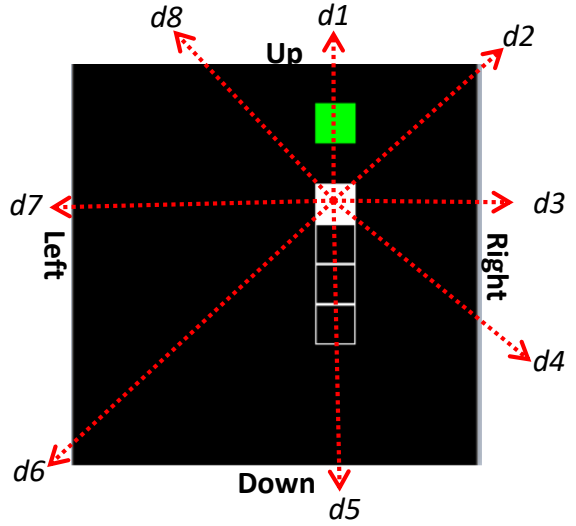
- Activation function can be chosen what ever we want, hear we used Rectified Linear Unit (ReLU) function for the hidden layer and Sigmoid function for the output layer.
- Number of neurons in Hidden layer and number of Hidden layers can be chosen what ever we want, for fast processing we chosen one Hidden layer with 8 Neurons.
- Output Layer has 4 neurones which will decide the direction of Snake move.

Deciding input layer

- In order to select the input layer, we have to find the attributes that helps the snake in getting maximum score.
- Here we selected the Vison to the food, body and wall around snake head in 8 direction from the top in the clockwise direction as 1st attribute and Head direction as 2nd Attribute.
- Head direction of a snake is chosen because, the perfect snake takes maximum number steps in head direction.

Neural Network Architecture





Input layer values

- Input layer values are given in between 0 to 1, so that snake will understand the attributes in the right way

- Based on number of blocks present in between Snake head and Wall/Body/Food, Input Neuron values are decided from 0 to 1.

- It is good for the snake to move in the direction that have a value One (or) nearer to One and It is bad for the snake to move in the direction that have a value Zero (or) nearer to Zero.

Let's see how we give input values for Wall

- One is given when the number of blocks between Head and Wall are maximum and Zero is given when no blocks are present in between Head and Wall.

$$\text{Wall} = \frac{number \text{ of blockes } between\ Head\ and\ Wall}{total\ number\ of\ blockes - 1}$$

Value for Body

- One is given when no body is present in that direction and Zero is given when no blocks are present in between Head and Body.

$$\text{Body} = \frac{number \text{ of blockes } between\ Head\ and\ Body}{total\ number\ of\ blockes - 1}$$

$$= 1 \textit{ (when body is not present in that direction)}$$

Value for Food

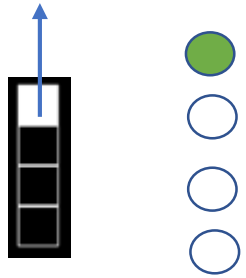
- It is good for snake to move in the direction of food, so One is given when no blocks are present in between Head and Food and Zero is given when no food is present in that direction.

$$\text{Food} = \frac{total\ number\ of\ blockes - number \text{of blockes } between\ Head\ and\ Food - 1}{total\ number of\ blockes - 1}$$

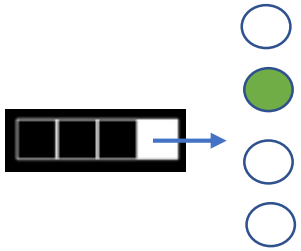
$$= 0 \textit{ (when Food is not present in that direction)}$$

Value for Head direction

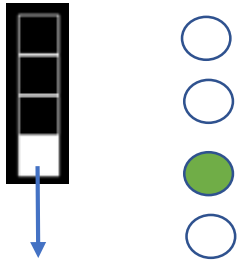
- In most of cases, it is good for the snake to move in the Head direction, so one is given in the direction of Head and zero is given for remaining 3 directions



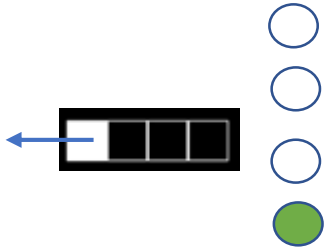
UP : [1,0,0,0]



RIGHT : [0,1,0,0]

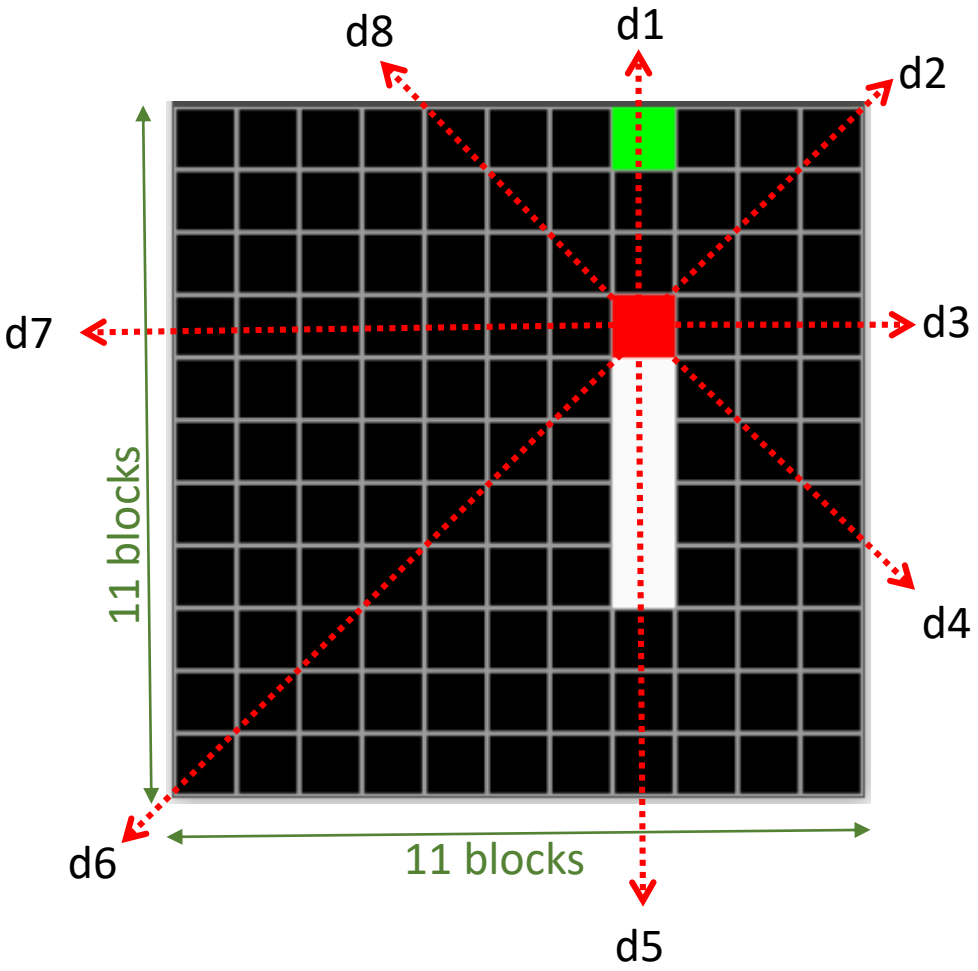


DOWN : [0,0,1,0]



LEFT : [0,0,0,1]

Let’s see the input value for an example of 11x11 blocks !!!!!

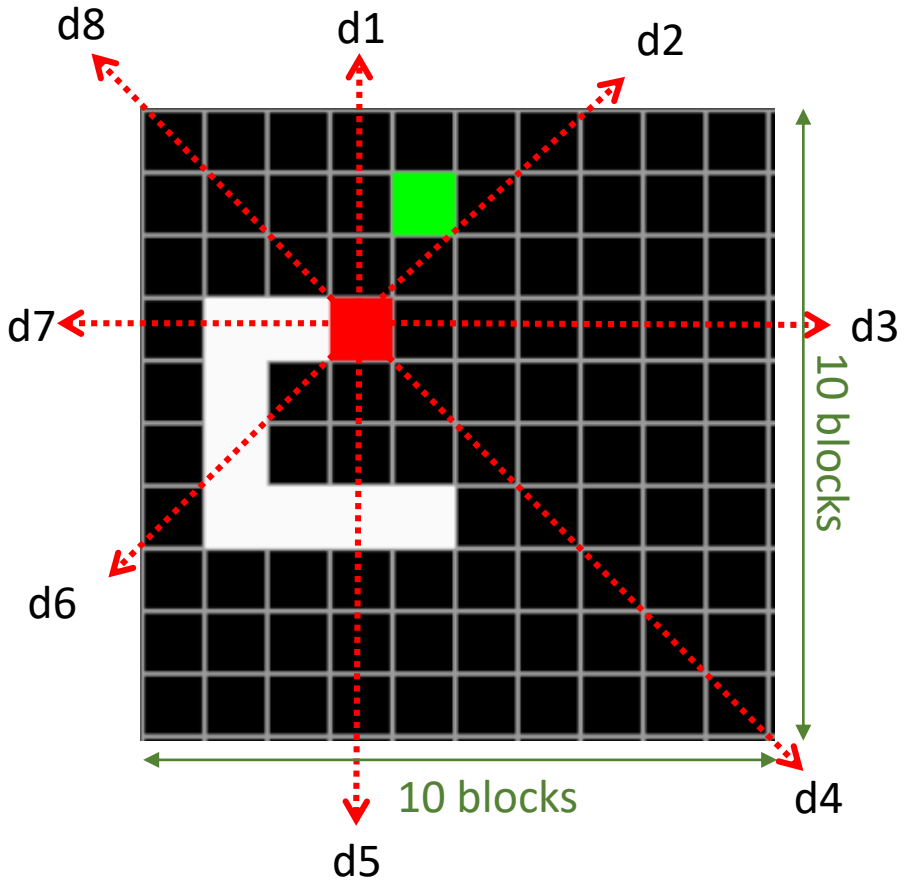


	d1	d2	d3	d4	d5	d6	d7	d8
Wall	3/10	3/10	3/10	3/10	7/10	7/10	7/10	3/10
Body	1	1	1	1	0	1	1	1
Food	8/10	0	0	0	0	0	0	0

Head direction : [1,0,0,0]

INPUT LAYER : [0.3,1,0.8,0.3,1,0,0.3,1,0,0.3,1,0,0.7,0,0,0.7,1,0,0.7,1,0,0.7,1,0,1,0,0,0]

Let’s see the input value for another example of 10x10 blocks !!!!!



	d1	d2	d3	d4	d5	d6	d7	d8
Wall	3/9	3/9	6/9	6/9	6/9	3/9	3/9	3/9
Body	1	1	1	1	2/9	1/9	0	1
Food	0	0	0	0	0	0	0	0

Head direction : [0,1,0,0]

INPUT LAYER : [0.333,1,0, 0.333,1,0, 0.666,1,0, 0.666,1,0, 0.666,0.222,0, 0.333,0.111,0, 0.333,0,0, 0.333,1,0,0,1,0,0]

Creating Fitness function

- We have to differentiate the great Snake from weak snake is based on its score and number of steps it taken to achieve that score, so we created a function by using Steps and Score as variables which will helps the Snake in getting maximum score in less steps.
- A function was created from which we will differentiate the snakes based on steps and it gives a values between -1 to 1

$$\text{Fitness} = \frac{Steps - \frac{Steps}{\text{Score}}}{Steps + \frac{Steps}{\text{Score}}}$$

- when the fitness value is nearer to 1 means the score was achieved in less steps.
- when the fitness value is nearer to -1 means the score was achieved in high steps.

- To avoid zero division error we added one to score variable in fitness function

$$\text{Fitness} = \frac{Steps - \frac{Steps}{\text{Score}+1}}{Steps + \frac{Steps}{\text{Score}+1}}$$

Creating Fitness function

- For avoiding negative values, we compressed the output value of Fitness function from [-1,1] to [0,1] as shown below

$$\text{Fitness} = 0.5 + (0.5 * \frac{Steps - \frac{Steps}{\text{Score}+1}}{Steps + \frac{Steps}{\text{Score}+1}})$$

- Till now, we differentiate the snakes based on steps, now we will add the Score to the fitness function, so that we will get the snake with high score in less steps from Fitness function

$$\text{Fitness} = Score + 0.5 + (0.5 * \frac{Steps - \frac{Steps}{\text{Score}+1}}{Steps + \frac{Steps}{\text{Score}+1}})$$

- In order to give importance to the decimal values we can multiple the whole fitness function by any value, where we used 10,0000

Let’s see how Fitness function will work !!!!

- Hear we take 3 snakes having different score in different steps, then we compare there fitness value from our fitness function.

Snake1 takes 180 steps to get 23 score
Snake2 takes 120 steps to get 23 score
Snake3 takes 500 steps to get 24 score

$$\text{Fitness} = Score + 0.5 + \left(0.5 * \frac{Steps - \frac{Steps}{Score+1}}{Steps + \frac{Steps}{Score+1}} \right)$$

S1 fitness = 23.7541
S2 fitness = 23.8214
S3 fitness = 24.5454

- S2 fitness > S1 fitness (because Snake2 got same score in less steps)
- S3 fitness > S2 fitness (because Snake3 got high score)

- From fitness values of three snakes, we observed that fitness function is giving 1st priority to snake with high score and then the snake with less steps

Crossover and Mutation

- After we play for each individual in the population, we will select the parents from Roulette wheel method and new chromosomes are generated from Random cross over method.
- After generating new Chromosomes, 5 percentage of each chromosome was muted about -0.5 to 0.5.

Let’s come to Inputs required for the code

1. Population size (500)
2. Parent size (50)
3. Number of blocks in the snake game (10 X 10)

- Size of population should not be more and less, if it is more it will take more time to complete the generation and if it less, then the snake have very less chinses to learn, so we take 500 as population size and 5 percentage of population as parent size that means 50

- To get the good snake within the less time, we used 10 X 10 blocks for the snake game and even we can change the block sizes after getting good population (i.e high avg score) because input values will be same for any block size

Other Code inputs

1. NN: Neural network layers
2. AF: Activation functions
3. pause_time: Adjust the beginning frame rate
4. key_sensitive: sensitive of up and down arrows