CS60050: Machine Learning Mini Project 2

Group - 7

[DSNN] 2D Self Driving Car Simulation Artificial Neural Network Model

Tanay Raghavendra (18EC10063), Ayan Chakraborty (18EC10075), Debjoy Saha (18EC30010)

Formulation of Problem Statement

1) To develop a neural network model which we need to train using an evolutionary algorithm in order to navigate the car on the given race tracks.

2) Observe the effects of variation of mutationParameter and chooseFittest parameters during evolution of the population of cars on the different race tracks.

Brief Theory and Methodology

A neural network is used to control the car. The car has 7 sensor inputs. Each sensor input gives the distance of the nearest obstacle in the direction of the sensor. These 7 sensor readings are fed as inputs to the neural network.

The car has four controls as follows: FA : Forward Acceleration, BA : Backward Acceleration, TL : Turn Left, TR : Turn Right. At any given time, the neural network can choose any combination of these 4 controls to apply.

Hence, the input layer of the neural network has 7 neurons (for 7 sensor readings) and, the output layer of the neural network has 4 neurons (for 4 control outputs). The binarization operation is applied to each neuron in the output layer. The binarize function has the following definition: $f(x) = 1$, if $x >= 0$ and $f(x) = 0$, if $x < 0$. The binarization output 0 indicates that control was not applied, and 1 indicates that control was applied.

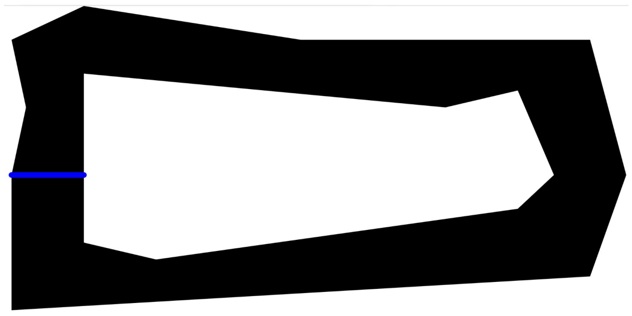
There are 3 hidden layers containing 10 neurons each. Each neuron performs the multiply and accumulate operation and then applies the ReLu Activation Function on the sum. ReLu Activation function is a popular activation function having the following definition: $f(x) = x$, if $x >= 0$ and $f(x) = 0$, if $x < 0$.

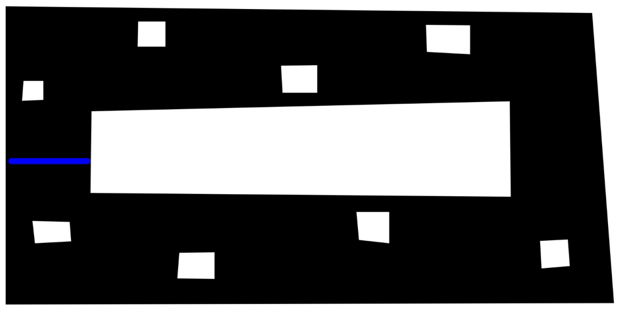
We are not training the neural network through back propagation, since it would require a huge amount of labelled training data for this application. Instead, we have adopted an evolutionary approach. There is a population of cars, with each car initialized by random weights. The weights of the best performing car are mutated slightly and then, passed onto the next generation, with some probability. This process repeats indefinitely. This ensures that each generation, we are choosing the best performing weights and then changing them slightly to obtain better weights, while also keeping other weights for better exploration of the hypothesis space.

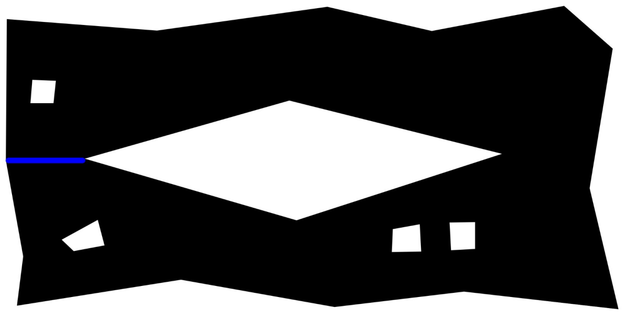
The scripts are written in Javascript. The training and the results are observed through a web browser.

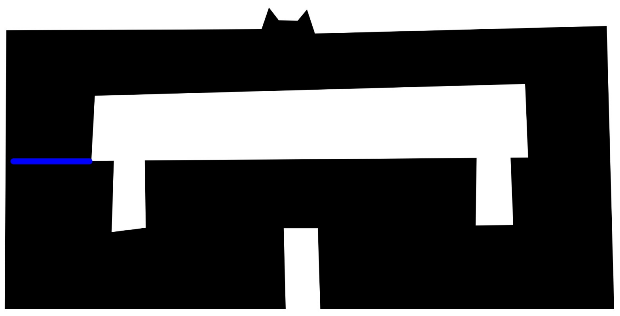
Plots of Results and Hyper-Parameter Tuning

Pictures of the different Race Tracks

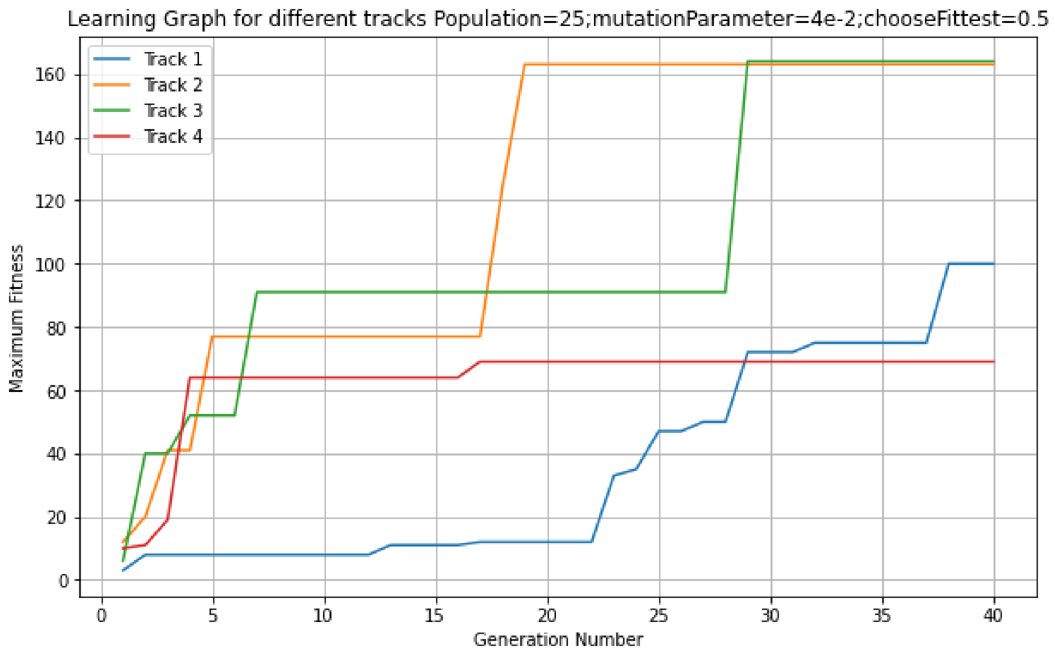

Race Track 1


Race Track 2

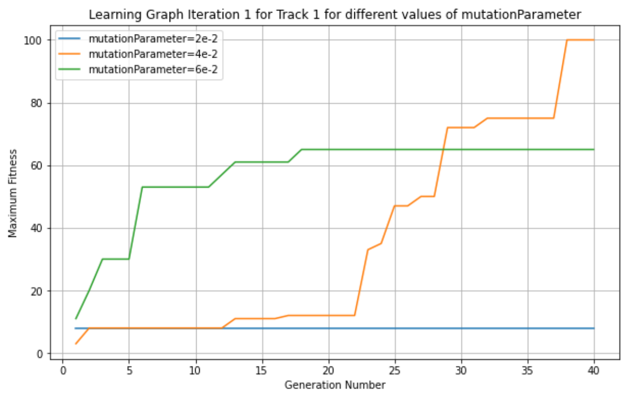

Race Track 3

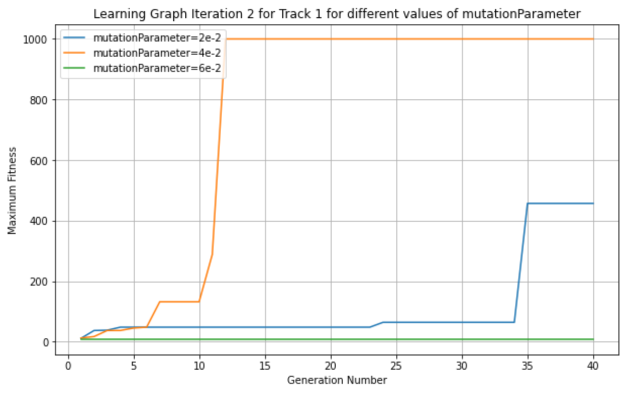

Race Track 4

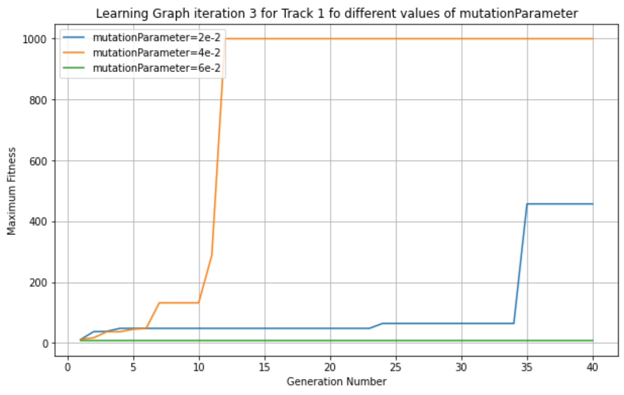
Plot of Fitness scores across different Race Tracks for default hyperparameter



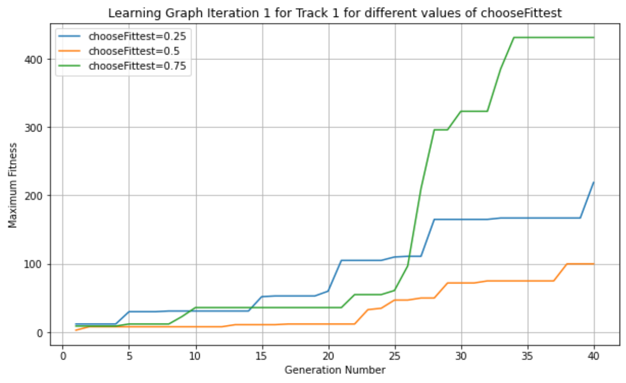
Plot of Fitness Scores by varying the mutationParameter on Race Track 1

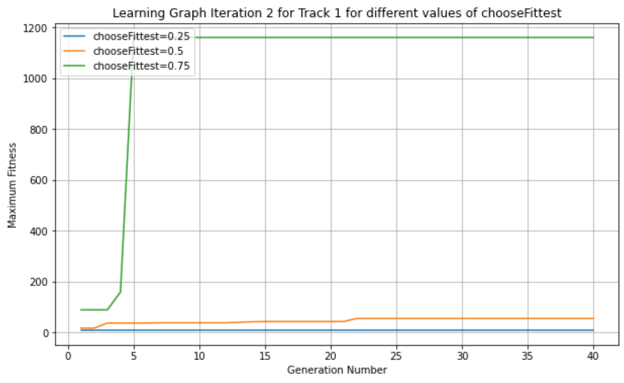


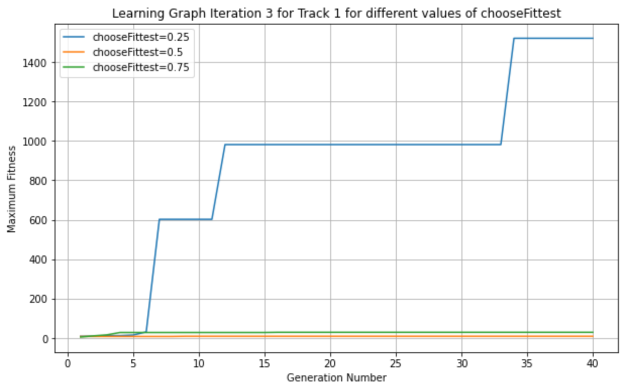




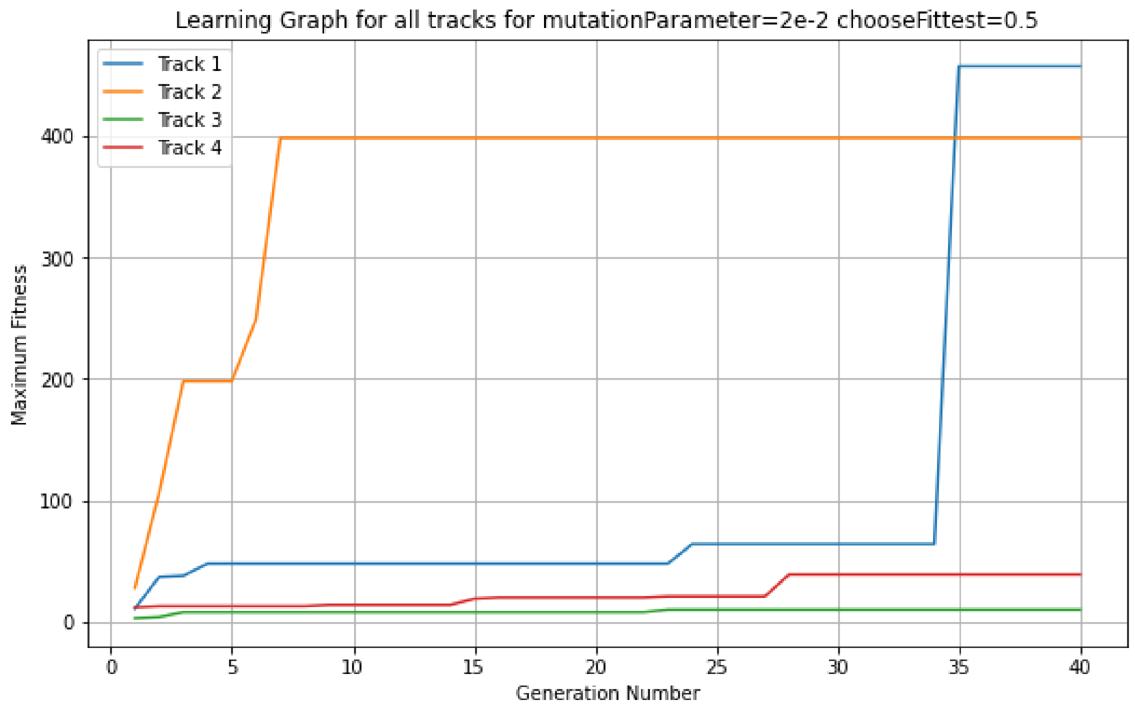
Plot of Fitness Scores by varying the chooseFittest parameter on Race Track 1



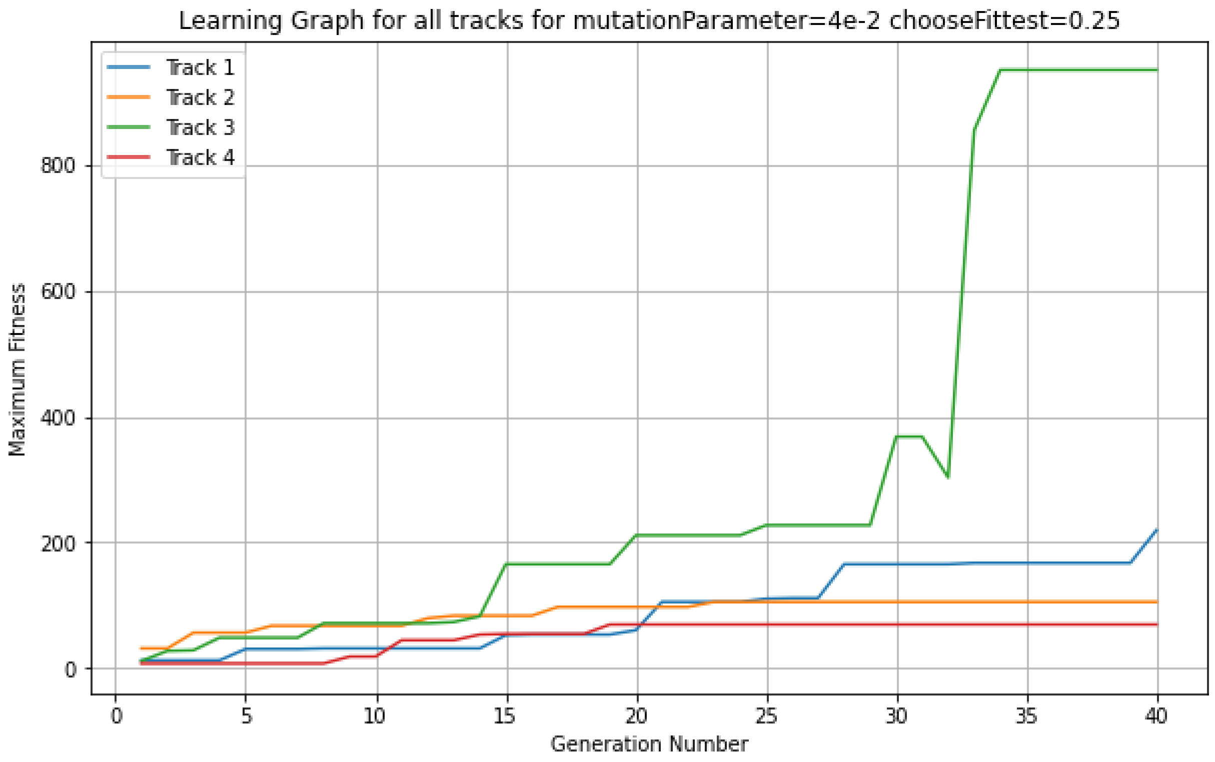




Plot of Fitness Scores for different mutationParameter on all Race Tracks



Plot of Fitness Scores for different chooseFittest parameter on all Race Tracks



Discussions and Conclusions

[Note: Hyperparameter variation results are taken 3 times each to lessen variances in results caused due to probability]

Effect due to the variation of:

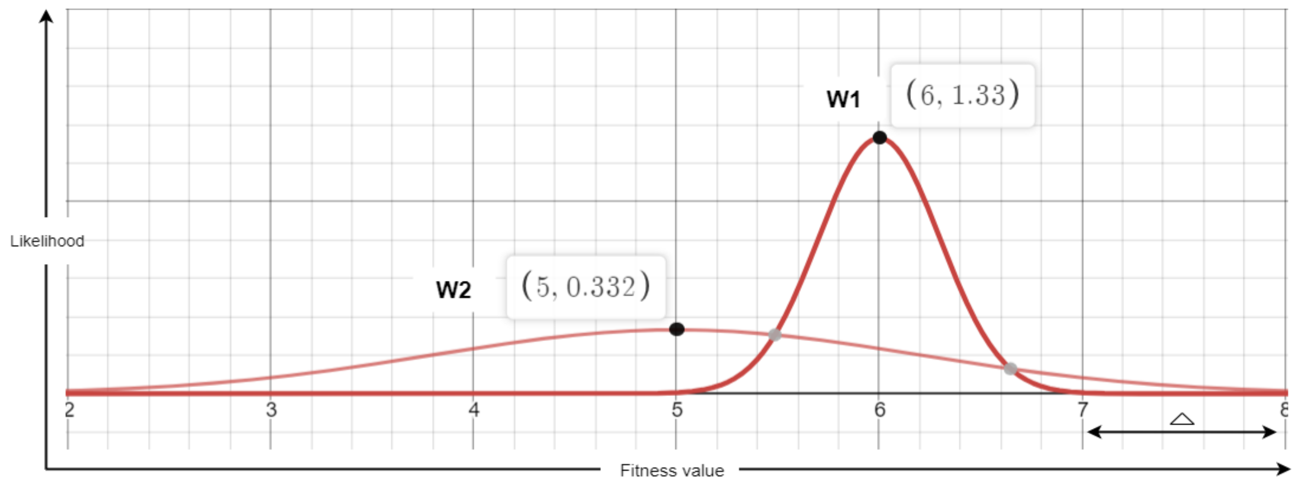
Track index - Intuitively, it would appear that track-1 is the easiest to traverse due to lesser amount of obstacles as compared to other paths. However, track-2 and 3 have an advantage over tracks-1 and 4, and that is the ability to form much smoother paths. In this respect, track-4 appears the hardest, since it has sharper turns as well as obstacles. We find our belief being supplemented by the learning plots obtained during training with varying track indexes. Performance on track-4 was consistently low across all training runs. The performance on track-1,2,3 was largely variable across different runs, largely dependent on the values of hyperparameters chooseFittest and mutationParameter, however on average performance varied as - Track 2 > Track 1 ≈ Track 3, which confirms our intuitions.

mutationParameter - This parameter defines how much mutation of the best performing weights is done while passing it to the next generation. If it is too high, then the weights are changed more rapidly. This leads to the possibility of instability and higher variance of fitness scores across generations. On the other hand, if the mutationParamater is too low, then the weights are updated very slowly, thus needing much more generations in order to achieve better fitness scores. This is validated by the plots. The mutationParameter 4e-2, which is the middle value consistently achieves the best performance across all runs. The mutationParameter 6e-2, shows variability: either achieving a good fitness score or a very bad fitness score. This variability is due to the instability discussed earlier. The mutationParameter 2e-2 (the lowest), updates much slowly across generations, compared to the other values. This again matches with our theoretical predictions.

chooseFittest - ChooseFittest parameter modifies the exploration ability of the evolutionary algorithm, higher value of choosefittest parameter results in a higher number of cars in the next generation being mutated from the best performing car from the current generation. This has the following consequences -

(a) Higher value of chooseFittest offers greater stability, more and more cars having neural network weights closer to the best performing car will mean that those cars on average will perform closer to the best performance of the current generation.

(b) However, a disadvantage can be getting stuck in local maximas. Higher value of chooseFittest can mean lesser exploration of the weight-space. Thus, we can miss out on a scenario, in which the current weights w_i might be performing poorly, but after some perturbation $w_{i+1} = w_i$ + delta, performance might surpass the current best. This trade-off is demonstrated in the following diagram -



In the diagram, we see a possible scenario, where choosing mutating less optimal weights can be ideal. On x-axis, we have the average fitness value that can be obtained by mutating the weights for each set, with a likelihood of that fitness value given by the y-axis value.

We observe, the weight set W1 performs better than W2 in the current generation, hence the higher mean of the Gaussian distribution. But there is a possibility that if we mutate the weights a little for W2, we might achieve performance that is impossible to achieve using W1 only(region marked with Δ). Very high value of chooseFittest will consistently mutate W1 only, and that can mean, we can have a higher fitness value on average but we might settle on a sub-optimal final fitness value. This presents a tradeoff.

We find the best performance two times with chooseFittest value of 0.75, and once with chooseFittest value of 0.25. chooseFittest parameter value of 0.5 performed worst consistently across all trials. We get the highest average maximum fitness value with 0.25, which, however is accompanied with a higher standard deviation as well, which is consistent with our theoretical explanation. However, these results depend a lot on the random perturbation to the weights and a lot of experiments must be performed to get reliable information.

Appendix

Table of all data used for Plot Generation

mutationParameter	chooseFittest	Track Number	Max Fitness average	Max Fitness std_dev
2e-2	0.5	1	156.67	260.1
4e-2	0.5	1	388	530
6e-2	0.5	1	41.33	29.7

mutationParameter	chooseFittest	Track Number	Max Fitness average	Max Fitness std_dev
4e-2	0.25	1	582.3	818.67
4e-2	0.5	1	54.3	45.5
4e-2	0.75	1	540	573.3

mutationParameter	chooseFittest	Track Number	Max Fitness
4e-2	0.5	1	100
4e-2	0.5	2	163
4e-2	0.5	3	164
4e-2	0.5	4	69

mutationParameter	chooseFittest	Track Number	Max Fitness
2e-2	0.5	1	457
2e-2	0.5	2	398
2e-2	0.5	3	10
2e-2	0.5	4	39

mutationParameter	chooseFittest	Track Number	Max Fitness
4e-2	0.25	1	219
4e-2	0.25	2	105
4e-2	0.25	3	951
4e-2	0.25	4	69