STUDIA INFORMATICA

Nr 1 (26) Systems and information technology 2022

Arkadiusz BOLESTA[1]
ORCID: 0000-0003-1719-1109
Jerzy TCHÓRZEWSKI[1]
ORCID: 0000-0003-2198-7185

[1] Siedlce University of Natural Sciences and Humanities
Faculty of Exact and Natural Sciences
Institute of Computer Science
ul. 3 Maja 54, 08-110 Siedlce, Poland

Neural model of the vehicle control system in a racing game. Part 2. Research experiments

DOI: 10.34739/si.2022.26.03

Abstract. This article, which is a continuation of the article under the same main title and subtitle: part 1 *Design and its implementation*, includes the obtained results of research experiments with the use of a designed and implemented racing game. It uses a neural model of the vehicle motion control system on the racetrack in the form of a Perceptron Artificial Neural Network (ANN). In designing the movement of vehicles on the racetrack, the following were used, inter alia, Godot Engine and MATLAB and Simulink programming environment. The numerical data (14 input quantities and two output quantities) for ANN training were prepared with the use of semi-automatic measurement of the race track control points. This article shows, among others, the results of 10 selected research experiments, testing and simulation, confirming the correct functioning of both the computer game and the model of the neural control system. As a result of simulation tests, it turned out that the longest lap of the track in the conducted experiments lasted 4 minutes and 55 seconds, and the shortest - 10.47 seconds. In five minutes, the highest number of laps was 34, while the lowest numbers of laps were 1 and 5. In the course of the experiments it was noticed that under the same conditions the ANN learning outcomes are sometimes different.

Keywords. Godot Engine, MATLAB and Simulink environment, Neural control system, Perceptron Artificial Neural Networks, Video games

1. Introduction

The project of a racing game with a car motion control system in the form of a neural model was implemented using mainly the Godot Engine game environment, the CLion environment, and the MATLAB and Simulink environment [8, 10-11, 24-25], which was described in detail in the works [6, 29]. It was shown, among others, that the design of the experiment was preceded by detailed research of various oriented algorithms and programming environments that can be used in designing a racing game [1-3, 5, 7-8, 10-11, 15, 17-19, 24-25 , 38], artificial intelligence algorithms such as expert systems [14, 34] and artificial neural networks [12-13, 16, 20-21, 23, 28, 30, 32-33, 36, 39], the detailed results of which are presented in in the works [6, 29].

A Perceptron Artificial Neural Network with one hidden layer was selected to control the movement of vehicles on the racetrack, assuming a different number of neurons (from 20 to 40) in the research. On the other hand, the ANN model of the neural control system was taught using the method of back propagation of errors and the data of the training file obtained as a result of semi-automatic registration of measurement points on two racing tracks, i.e. on the basis of manual control of the car and automatic registration of input and output data obtained on the racetrack [6, 29].

Cars moved on racing tracks had 14 input values, of which, in the optimal solution, 13 distance sensors directed at different angles in relation to the car's direction of travel and the current speed of the vehicle. Moreover, two output quantities were adopted, i.e. the turn factor and the acceleration factor. Each sensor returns the distance in pixels as a result of a collision with another object, including a racetrack wall. The movement of the vehicle takes into account, inter alia, friction, air resistance, length of the vehicle and at speeds higher than the prescribed limiting speed also skid. In total, 32 research experiments were carried out, of which the most interesting 10 experiments are presented in this paper, taking into account the conditions described in the works [6, 29].

During the teaching of the Perceptron Artificial Neural Network, a problem of disappearing gradients was encountered, because the derivatives were of very small values, and their multiple multiplication led to too small numbers and too small changes in the input layer of neurons. As a result of additional activities, including doubling the number of training pairs eliminated this problem and the ANN learned stably. In addition, the following possibilities were considered: the introduction of the neuron activation function on the hidden layer of the linear function, the so-called improved (ReLU) in place of the sigmoidal function, as well as the introduction of the so-called momentum, which usually reduces the ANN tendency to instability and avoids rapid fluctuations (e.g. weight change in a previous epoch or two eras), and even the so-called smoothing by introducing the coefficient β = 1-α, which on the other hand affects the deterioration of the convergence of the learning algorithm [6, 9, 12, 16, 23, 28, 39].

2. GUI of implemented racing game

The GUI of the implemented racing game is shown in Fig. 1, while on the left side there are buttons for selecting one of two tracks or a blank map painted in a checkerboard pattern so that it is possible to watch the vehicle's movement [6]. However, on the right side there are buttons for selecting the game mode in the options: manual control, ANN control, player and ANN races, and manual control with measurement recording [6].

If one of the versions of the neural networks is selected, the user goes to the settings screen and exits the game, while the settings screen shown in Fig. 2 allows you to save the paths to files with input data and output data used to train the ANN model of the neural vehicle motion control system. There is also a checkbox here, thanks to which you can enable normalization of the data used to train ANN by scaling them to the range <0; 1>. Once the racetrack is selected, the actual game screen is visible as shown in Fig. 3. While driving, the camera follows the car in the center of the screen. The vehicle state can be additionally recognized by its color, which is blue when controlled by the player or red otherwise.

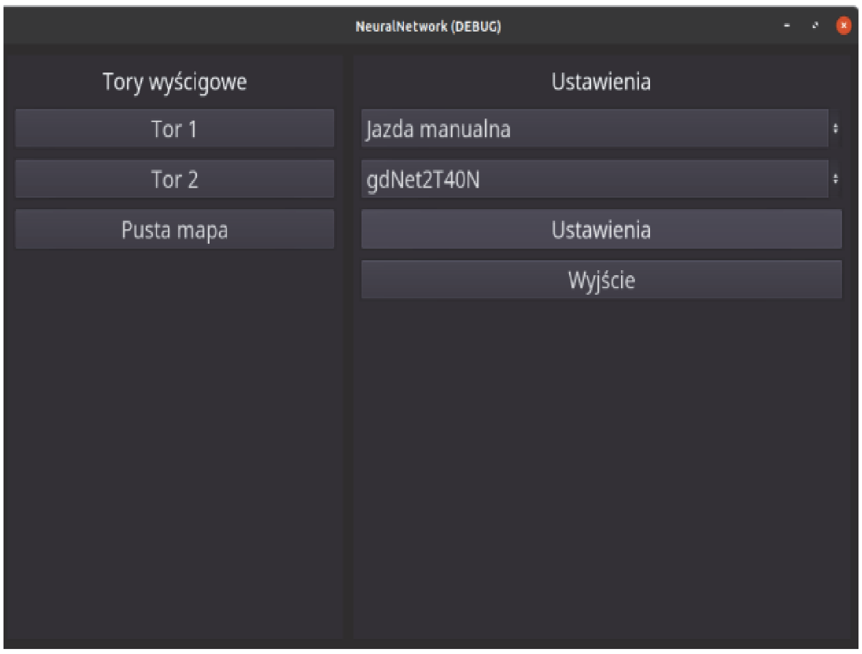
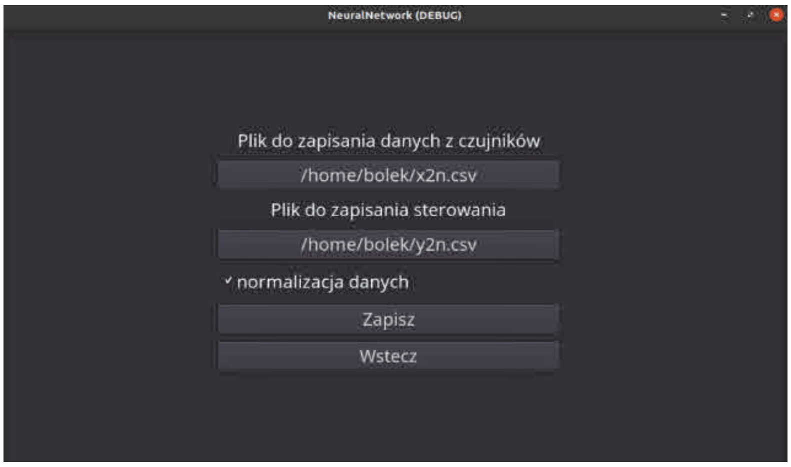


Figure 1. Game's main menu. Selected designation: Tory wyścigowe - Racing tracks (tor 1 – Track 1, Tor 2 – Track 2), Pusta mapa - A blank map, Ustawienia – Settings, jazda manualna - manual driving, wyjście – Exit. Source: [6].


NeuralNetwork (DEBUG)

Plik do zapisania danych z czujników

/home/bolek/x2n.csv

Plik do zapisania sterowania

/home/bolek/y2n.csv

✓ normalizacja danych

Zapisz

Wstecz

Figure 2. Game's settings. Selected designation: Plik do zapisania danych z czujników - File for saving sensor data, Plik do zapisania sterowania - File to save the control, Normalizacja danych - Data normalization, Zapisz - Save, Wstecz – Return. Source: [6].

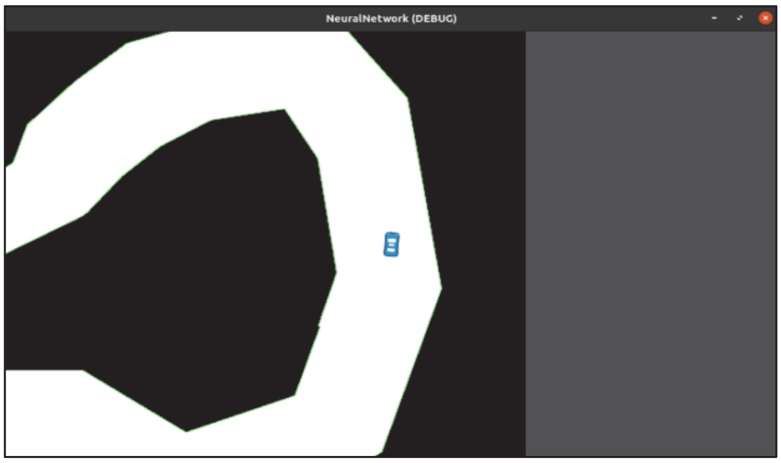


Figure 3. Game's window. Source: [6].

3. Research experiments

The analysis of the artificial neural network learning and its operation as a vehicle motion control system in a racing game was carried out on properly prepared 32 research experiments consisting in teaching and testing ANN with the use of appropriate training and testing files, and then its activation and testing the correctness of its behavior.

The results of the 10 most interesting experimental studies were published in, inter alia, in Tab. 1. A rich compendium of other interesting cases of the course of the game is included in [6]. The research experiments consisted in running the equipped cars with a different number of sensors and different architectures of the Perceptron Artificial Neural Network, which was a neural model of the car motion control system on the racetrack.

The changes in the Artificial Neural Network architecture mainly concerned the number of neurons in the hidden layer, which were changed from 20 neurons to 40 neurons. Moreover, the ANN learning results were influenced by the learning parameters by the back propagation error method, including the learning speed and learning inertia. The research experiments started with the use of five distance sensors in the car. Unfortunately, it turned out that then the effects of learning and testing ANN as a model of the neural vehicle motion control system were unsatisfactory. Ultimately, the car was equipped with 13 sensors.

In the case of 5 sensors, a not very flexible possibility of controlling the movement of the vehicle was observed, which repeatedly hit the wall or, on the contrary, did not even try to get close to it. Moreover, on sharp turns (of 180^o and more), the car did not receive enough information about the surroundings, which also caused the car to hit the wall. In order to increase the ability to control the movement of the vehicle, the number of sensors was increased to 13, which turned out to be sufficient for its flexible control.

Initially, no attention was paid to the number of hidden layer neurons, but as the simple possibilities of changing the flexibility of the vehicle motion exhausted, it was decided to learn and test several artificial neural networks with a different number of neurons in the hidden layer. Due to the relatively small number of entries and exits from the ANN, the ANN behavior for more than one hidden layer was not investigated.

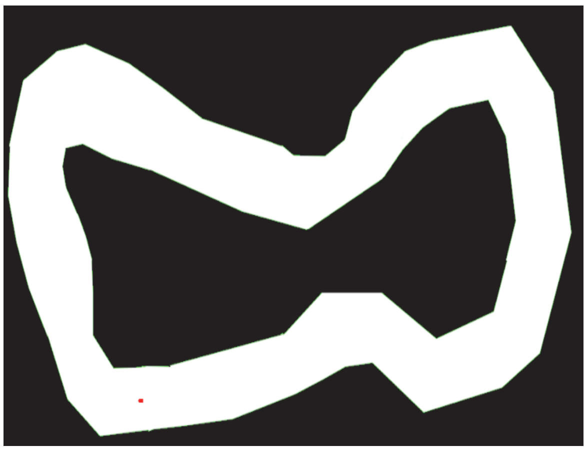
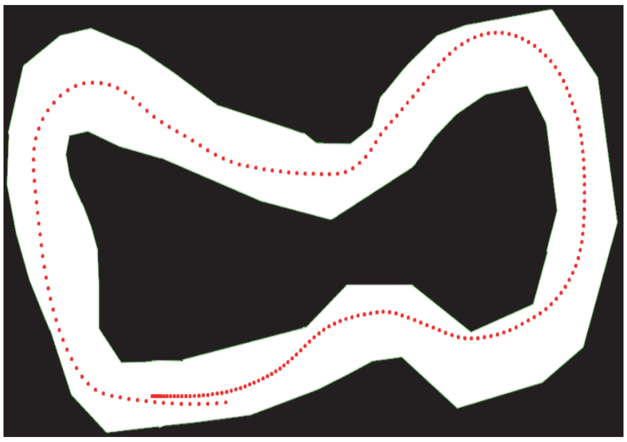
In addition, the number of measurements was also increased, i.e. more training and testing pairs were created by measuring control points. In this way, three training files were obtained, that is: a file with data from measurements on the first track (so-called easy track), a file with data from measurements on the second track (so-called complex track) and a file obtained by mixing measured data on the first and second racetracks.

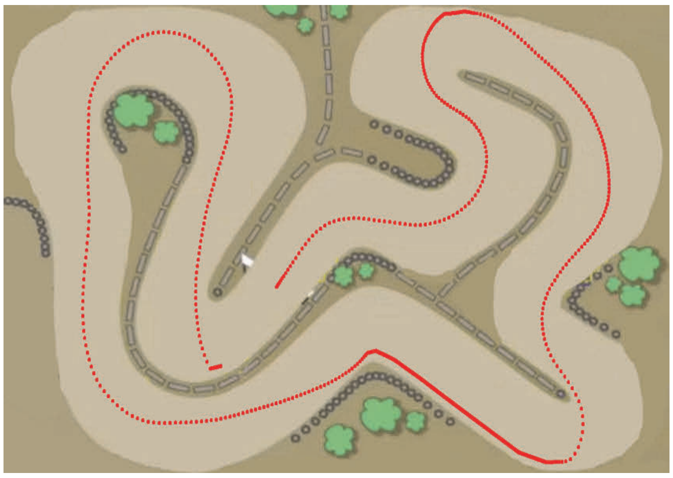
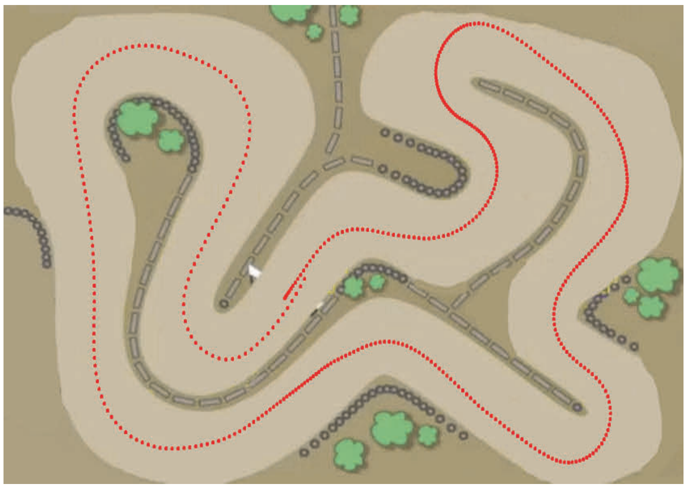
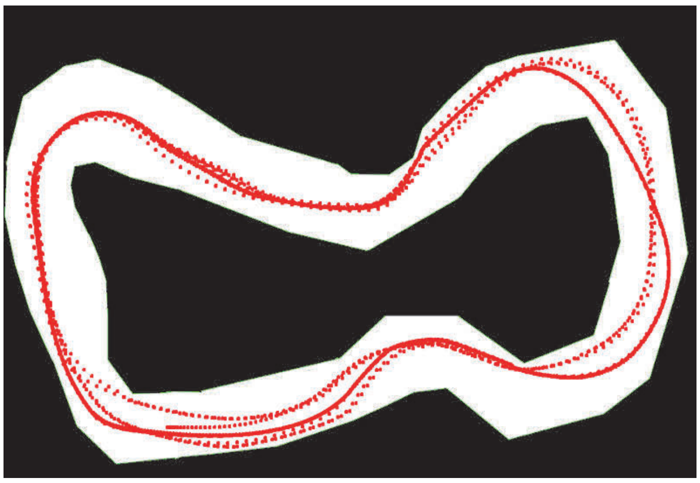
Moreover, the measurement points obtained in the case of the first track were placed in evenly marked places, i.e. at points that are often the least collision-free, which allowed for a relatively simple construction of the experiment, including the neural model. It turned out, however, that the measurement data obtained in this way was insufficient for the correct teaching and testing of the ANN, which, as a neural model of the system controlling the movement of the vehicle, led to a collision with the walls of the racetrack.

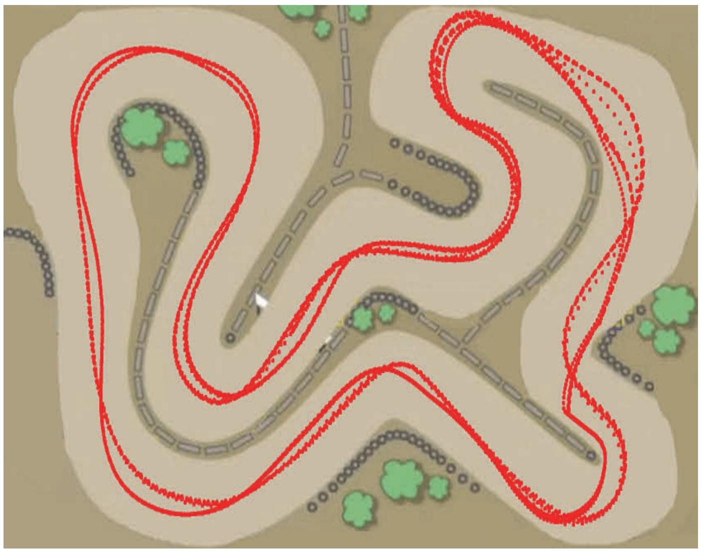
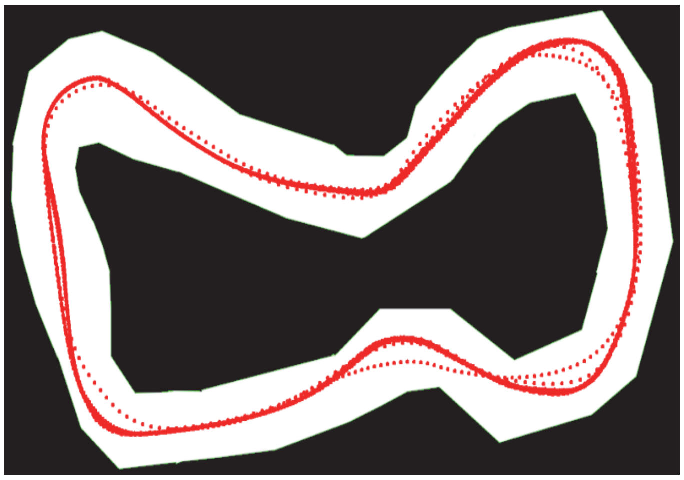
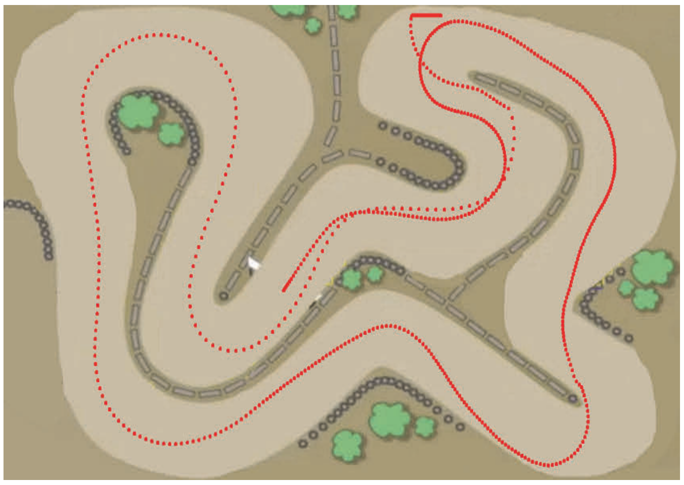
Moreover, the ANN learned to control the movement of the car on the first track could not control the movement of the car on the second, more complex track, where the collisions occurred more frequently and in a shorter period of time. The analysis of the conducted research experiments, of which the results of 4 experiments are shown in Tab. 1, showed, among others, that the initialization of the weights played an important role in the Perceptron ANN initialization due to the randomness of their selection.

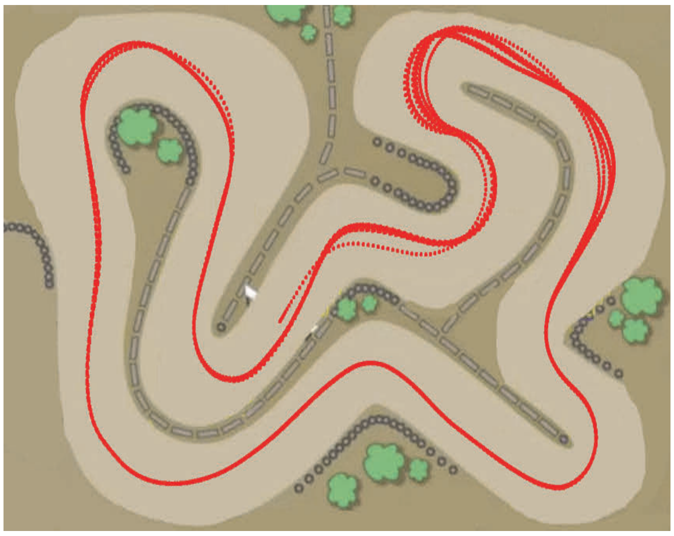
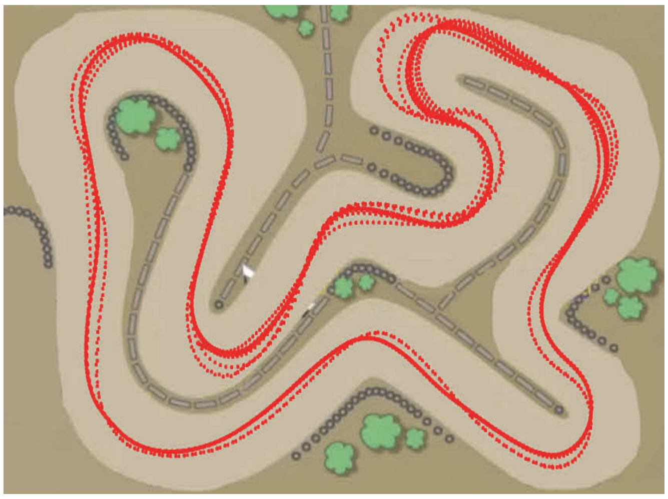
It turned out that two artificial neural networks with the same architecture and identical parameters, learned on the same measurement data of track checkpoints, achieved different results, which was obtained as a result of comparing the first two experiments described in detail in [6]. Well, one of them caused the car to not move, while the other was able to circle the entire track. Another interesting case was in experiment 6, in which cars, after lapping several times, have different speeds, which in turn affects other possibilities of ANN control of the vehicle movement, which takes place on a completely new, less collision trajectory.

Table 1. Results of 10 experiments on car movement control on a racetrack using a neural model in the form of an Artificial Neural Network. Source: [6].

Experiment 1. ANN tested on easy track, vehicle start on a easy track section	
Architecture: Perceptron ANN with 14 input neurons, 2 output neurons, with one hidden layer with 30 neurons, with tansig activation function for the first layer and purelin for the second layer of neurons. **Input and testing data**: the data for learning were obtained as a result of measurements carried out on a complex track. **Experiment results**: an attempt was made to measure the time of one lap, which was not achieved because the car did not go forward, but was moving backwards at a very low speed.	
Experiment 2. ANN tested on a easy track, vehicle start on a easy track section	
Architecture: Perceptron ANN with 14 input neurons, 2 output neurons, with one hidden layer with 30 neurons, with tansig activation function for the first layer and purelin for the second layer of neurons. **Input and testing data**: the data for learning were obtained as a result of measurements carried out on a complex track. **Experiment results**: one lap time was measured, which was 11.46 s.	

Experiment 3. ANN tested on a complex track, vehicle start on a easy track section	
Architecture: Perceptron ANN with 14 input neurons, 2 output neurons, with one hidden layer with 30 neurons, with tansig activation function for the first layer and purelin for the second layer of neurons. **Input and testing data**: the learning data was obtained from measurements carried out on both racetracks. **Experiment results**: an attempt was made to measure the time of one lap which the car failed to complete by hitting a wall at high speed.	
Experiment 4. ANN tested on a complex track, vehicle start on a easy track section	
Architecture: Perceptron ANN with 14 input neurons, 2 output neurons, with one hidden layer with 40 neurons, with tansig activation function for the first layer and purelin for the second layer of neurons. **Input and testing data**: the learning data was obtained from measurements carried out on both racetracks. **Experiment results**: the time of one lap was measured, which was 23.55 s.	
Experiment 5. ANN tested on a easy track, start of the vehicle on a easy track section	
Architecture: Perceptron ANN with 14 input neurons, 2 output neurons, with one hidden layer with 30 neurons, with tansig activation function for the first layer and purelin for the second layer of neurons. **Input and testing data**: the learning data was obtained from measurements carried out on a complex racetrack. **Experiment results**: The number of laps of the race track in 5 minutes was 32 times, the car collided with the wall, and after a few laps it changed its trajectory, where there were fewer collisions with the wall.	

Experiment 6. ANN tested on a complex track, the start of the vehicle on the easy sections	
Architecture: Perceptron ANN with 13 input neurons, 2 output neurons, with one hidden layer with 30 neurons, with tansig activation function for the first layer and purelin for the second layer of neurons. **Input and testing data**: the learning data was obtained from measurements carried out on a complex racetrack. **Experiment results**: The car circled the track 16 times, but collided with the walls. However, after a few laps, he changed the trajectory to a completely new one, on which he rode with much less collisions with walls.	
Experiment 7. ANN tested on a easy track, the start of the vehicle on the easy sections	
Architecture: Perceptron ANN with 13 input neurons, 2 output neurons, with one hidden layer with 30 neurons, with tansig activation function for the first layer and purelin for the second layer of neurons. **Input and testing data**: the learning data was obtained from measurements carried out on both racetracks. **Experiment results:** the car circled the track 33 times. During the movement, it basically kept a constant trace of the trajectory almost all the time.	
Experiment 8. ANN tested on a complex track, vehicle start on a easy track section	
Architecture: Perceptron ANN with 14 input neurons, 2 output neurons, with one hidden layer with 30 neurons, with tansig activation function for the first layer and purelin for the second layer of neurons. **Input and testing data**: the learning data was obtained from measurements carried out on both racetracks. **Experiment results**: The car circled the track once, then picked up high speed and did not make a sharp turn.	

Experiment 9. ANN tested on a complex track, vehicle start on a easy track section	
Architecture: Perceptron ANN with 13 input neurons, 2 output neurons, with one hidden layer with 40 neurons, with tansig activation function for the first layer and purelin for the second layer of neurons. **Input and testing data**: the learning data was obtained from measurements carried out on both racetracks. **Experiment results:** the car circled the track 12 times. During the movement, it basically kept a constant trace of the trajectory almost all the time.	
Experiment 10. ANN tested on a complex track, vehicle start at a curve in the track	
Architecture: Perceptron ANN with 13 input neurons, 2 output neurons, with one hidden layer with 40 neurons, with tansig activation function for the first layer and purelin for the second layer of neurons. **Input and testing data**: the learning data was obtained from measurements carried out on both racetracks. **Experiment results**: The car circled the track 13 times. Despite the conditions similar to experiment 9, the car's trajectory turned out to be more variable.	

4. Conclusions

In the ten presented research experiments, for the purpose of checking the correctness of motion, a straight track was selected four times and a complex track six times. Nine times the start was from the track on a relatively straight road and once around a curve. On the other hand, in six cases, the data for the experiment was measured on both tracks and in four cases - on a complex track (data obtained only on a straight track were not used). In seven cases, ANN in the hidden layer had 30 neurons, and in three cases, 40 neurons.

As a result of the conducted research experiments (Ei, i - number of the next experiment), the following results were obtained:

– time for one lap of a racetrack in an experiment: E2 – 11,46 s, E4 - 23,55 s,
– the number of complete laps of the track: E5 - 32, E9 – 12, E10 – 13, E6 – 16, E7 – 33,
– collisions with the track wall: E3 - before the first lap of the track, E8 - one lap as a result of taking a sharp turn, E5 - continuous impacts, but driving after the impact, E6 - collisions and then a change in the trajectory of movement along the racetrack and fewer collisions, E9 - no collisions with the wall,
– change of trajectory on the racetrack: E5 - after three laps on the track with fewer collisions with the wall, E6 - after five laps on the track with fewer collisions with the wall, E9 - approximately constant driving trajectory during 12 laps of the racetrack, E10 - approximately constant trajectory of the car's driving during 13 laps of the racetrack (but worse results than for E9, despite similar conditions of participation of the car in motion on the track),
– backward movement: E1 - slow backward movement of the car on an easy race track, despite the use of data obtained for learning ANN on a compound track.

Thus, as a result of the conducted design, implementation and investigated research experiments (Tab. 2), it was shown that the Perceptron Artificial Neural Network can be used as a neural model of vehicle motion control on the racetrack. For the purpose of obtaining a training and testing file, a research experiment was designed to enable measurements of the appropriate number of points on the racetrack. The learning pairs obtained were used in teaching and testing the Artificial Neural Network of the neural model of the car control system along a given race track.

Table 2. Summary of selected research results of 10 experiments on car motion control on a racetrack using a neural model in the form of a Perceptron Artificial Neural Network. Denotation: ST – Simple Track, CT – Compex Track, SS – Straight Section of Track, CS – Courve Section. Source: own elaboration.

Condition					Results			
No	**Type of track**	**Take-off place**	**Place of measurement data**	**Number of hidden layer neurons**	**Time for one lap [s]**	**Number of complete laps**	**Description of a collision with a wall**	**Movement**
E1	ST	SS	CT	30	x	0	no collisions	reverse driving
E2	ST	SS	CT, ST	30	11.46	1	on a straight stretch	x
E3	CT	SS	CT, ST	30	x	x	before the first lap	x
E4	CT	SS	CT	30	23.55	1	on a straight stretch	x
E5	ST	SS	CT	30	300	32	many collisions, but keep going	change of trajectory after several collisions
E6	CT	SS	CT	30	x	16	collisions, keep going	after a few laps change of

								trajectory with fewer collisions
E7	ST	SS	CT, ST	30	x	33	no collisions	essentially constant trajectory of motion
E8	CT	SS	CT, ST	30	x	x	collision in a bend after one lap	no further driving
E9	CT	SS	CT, ST	30	x	12	no collisions	permanent trace of trajectory
E10	CT	CS	CT, ST	30	x	13	no collisions	movement on a very variable trajectory

In order to meet the requirements resulting from the assumed conditions of research experiments, it was necessary to design an appropriate ANN architecture and select appropriate parameters for it, and above all the number of hidden layers. It turned out that one hidden layer with the number of neurons from 20 to 40 is sufficient, depending on the conducted research experiment. The tansig function for the hidden layer and the purelin function for the output layer of neurons were used as the function of neuron activation.

In experimental studies, the key issues turned out to be, among others: the number of training pairs, i.e. measuring points on the racetrack, the way of initiating neuron weights, the results of the analysis of testing and simulations checking the movement on the racetrack with the use of, as well as the interpretation of the obtained results and their translation to improve the parameters of the experiment and the artificial neural networks used in it.

During the construction of the ANN, 14 input quantities and two output quantities were used. The inputs were the distances returned by 13 sensors placed on the car and the current speed of the car, and the outputs were the acceleration coefficient and the steering coefficient of the vehicle. The development environments consisted of such components as: CLion environment, Godot game engine and MATLAB computing environment. The aforementioned tools made it possible to efficiently perform, that is, to design and implement research experiments, as well as to test them later and conduct simulation studies.

As could be expected, the first experiments were burdened with certain failures, consisting in the collisions of the car with the walls of the racetrack. As a result of testing the learned ANN model of the neural vehicle motion control system, that the main cause of the collision and the lack of complete influence on the car control was too small number of training and testing pairs. It turned out that after introducing a vastly large number of training pairs, the designed and implemented ANN met the requirements for flexible control of vehicle movement on the racetrack.

As a result of simulation tests, it turned out that the longest lap of the track in the conducted experiments lasted 4 minutes and 55 seconds, and the shortest - 10.47 seconds. In five minutes, the highest number of laps was 34, while the lowest numbers were 1 and 5. In experiment 5, the car circled the race track 32 times in 5 minutes. During the experiments, shortcomings of the neural network were noticed, which consisted in the fact that under the same conditions, the learning outcomes may be different.

In continued research, both the ANN learning method itself can be further improved, and a way to increase the accuracy of Perceptron ANN learning by other artificial intelligence methods, such as evolutionary algorithms [4, 14, 21, 31], or instead of the Artificial Neural Network to vehicle traffic control, for example, use ant algorithms [22, 26, 35], firefly algorithms [27], or a systemic immunological algorithm to increase the ANN resistance to wall collisions [37].

In further research, it may also be worth considering how to use the obtained results in this research in the design of a real racing vehicle on a track close to the real track.

References

1. Abdal M.: Artificial Intelligence in Racing Games. University of Birmingham. https://www.cs.bham.ac.uk/~ddp/AIP/RacingGames.pdf [access: 2021-04-16].
2. Algorytm BFS. https://eduinf.waw.pl/inf/alg/001_search/0126.php [access: 2021-04-16].
3. Algorytm Dijkstry. http://www.algorytm.org/algorytmy-grafowe/algorytm-dijkstry.html [access: 2021-04-16].
4. Arabas J.: Wykłady z algorytmów ewolucyjnych (Eng. Lectures on evolutionary algorithms). WNT. Warszawa 2003, pages 303.
5. Barczak A. and Woźniak H.: Comparative Study on Game Engines. Studia Informatica. Systems and Information Technology 1-2(23)2019.
6. Bolesta A.: Artificial Neural Networks as vehicle control systems in racing games. Master's thesis written under the supervision of dr hab. inż. Jerzy Tchórzewski, prof. uczelni w Instytucie Informatyki, na kierunku informatyka na Wydziale Nauk Ścisłych i Przyrodniczych, UPH w Siedlcach, Siedlce 2021, pages 82.
7. Cui X. and Shi H.: Direction Oriented Pathfinding In Video Games, International Journal of Artificial Intelligence & Applications, 2 Oct. 2011.
8. Dokumentacja silnika Godot (Eng. Documentation of the Godot engine), https://docs.godotengine.org/pl/latest/ [access: 2021-04-16].

9. Flasiński M.: Wstęp do sztucznej inteligencji (Eng. Introduction to Artificial Intelligence). WN PWN, Warszawa 2011.
10. Godot na platformie Steam. URL: https://store.steampowered.com/app/404790/Godot%5C_Engine/ [access: 2021-04-16].
11. Graham R., McCabe H., and Sheridan S.: Neural Networks for Real-time Pathfinding in Computer Games. Jan. 2004. Środowisko CLion. https://www.jetbrains.com/clion/?gclid=CjwKCAjw8cCGBhB6EiwAgOReyyEyTBD5gq4mvxeo4vMWqwDXjk0wpBxnIIVewVGuPop667dqTgDLRoC3P4QAvD_BwE&gclsrc=aw.ds. [access: 2021-04-16].
12. Horzyk A.: Metody Inżynierii Wiedzy – Uczenie głębokie i głębokie sieci neuronowe (Eng. Knowledge Engineering Methods - Deep learning and deep neural networks), AGH w Krakowie, Warszawa 2019, http://home.agh.edu.pl/~horzyk/lectures/miw/MIW-DL.pdf [access: 2021-04-16].
13. Kłopotek M., Tchórzewski J.: The concept of discoveries in evolving neural net, Advances in Soft Computing, IPI PAN, No. 17, Warszawa 2002, pp. 165-174.
14. Mulawka J.: Systemy ekspertowe (Eng. Expert systems), WNT, Warszawa 1996, pages 235.
15. Oliveira M. M., Chan M. T., Chan C. W., and Gelowitz C.: Development of a Car Racing Simulator Game Using Artificial Intelligence Techniques, International Journal of Computer Games Technology (Nov.), p. 839721, 2015 [access: 2021-04-16].
16. Osowski S.: Sieci neuronowe do przetwarzania informacji (Eng. Neural networks for information processing). OW PW, Warszawa, pages 422, 2013.
17. Patel A.: Introduction to A*, http://theory.stanford.edu/~amitp/Game Programming/AStarComparison.html [access: 2021-04-16].
18. Przychodzki M: Neuro-evolution of artificial neural networks based on the NEAT algorithm. Master's thesis written under the supervision of dr Artur Niewiadomski, kierunek informatyka, Wydział Nauk Ścisłych i Przyrodniczych, Uniwersytet Przyrodniczo-Humanistyczny w Siedlcach, Siedlce 2021.
19. Robbins M.: Neural Networks in Supreme Commander 2, https://ubm- twvideo01s3.amazonaws.com/o1/vault/gdc2012/slides/Summit_AI/Robbins_Michael_Off%20the%20Beaten.pdf. [access: 2021-04-16].

20. Ruciński D.: The Influence of the Artificial Neural Network type on the quality of learning on the Day-Ahead Market model at Polish Electricity Exchange join-stock company. Studia Informatica. Systems and Information Technology. Vol. 1-2(23)2019, pp.77-94.

21. Rutkowski L.: Metody i techniki sztucznej inteligencji (Eng. Methods and techniques of artificial intelligence). WN PWN, Warszawa 2017.

22. Sitkiewicz T., Tchórzewski J.: Wykorzystanie algorytmów mrówkowych do poprawy funkcjonowania algorytmu ewolucyjnego dla zagadnień transportowych (Eng. The use of ant algorithms to improve the functioning of the evolutionary algorithm for transport issues), Zeszyty Naukowe AMW, Nr 169 K/1, pp. 349-362, 2007.

23. Tadeusiewicz R. and Szaleniec M.: Leksykon sieci neuronowych (Eng. Lexicon on Neural Networks), Wydawca Projekt Nauka, pages 134, Jan. 2015.

24. Strona główna silnika Godot (Eng. The home of the Godot engine), https://godotengine.org/ [access: 2021-04-16].

25. Strona główna MATLAB & Simulink, https://www.mathworks.com/products/matlab. html [access: 2021-04-16].

26. Szewczak M., and Trojanowski K.: Wirtualne laboratoria optymalizacji heurystycznej: wykorzystanie algorytmów mrówkowych (Eng. Virtual heuristic optimization laboratories: the use of ant algorithms). Studia Informatica. Systems and Information Technology 1-2(11)2003, pp. 87–100.

27. Świtalski P., Bolesta A.: Firefly algorithm applied to the job-shop scheduling problem. Studia Informatica. Systems and Information Technology. Vol. 1-2(25)2021, pp.87-100.

28. Tadeusiewicz R.: Elementarne wprowadzenie do techniki sieci neuronowych z przykładowymi programami (Eng. Elementary introduction to the technique of neural networks with sample programs). Problemy Współczesnej Nauki: Informatyka. AOW, 1998, https://books.google. pl /books?id=SjgzygAACAAJ [access: 2021-04-16].

29. Tchórzewski J.: Metody sztucznej inteligencji i informatyki kwantowej w ujęciu teorii sterowania i systemów (Eng. Methods of artificial intelligence and quantum computing in terms of control theory and systems), Wydawnictwo UPH w Siedlcach, Siedlce 2021, pages 343.

30. Tchórzewski J.: Systemowy Algorytm Ewolucyjny SAE (Eng. Systemic Evolutionary Algorithm), Bio-Algorithms and Med-Systems, Vol. 1, No. 1/2, 2005, pp. 61-64.

31. Tchórzewski J., Kłopotek M.: A Case Study in Neural Network Evolution, Prace Naukowe Instytutu Podstaw Informatyki PAN, Nr. 943, IPI PAN, Warszawa 2002.
32. Tchórzewski J., Kłopotek M.: The Concept of Making Discoveries in Evolving Neural Net, Intelligent Information Systems 2002, Physica-Verlag HD, pp. 165-174.
33. Tchórzewski J.: Systemy ekspertowe (Eng. Expert Systems), [w:] Użytkowanie mikrokomputerów IBM PC. Część II. Podstawowe oprogramowanie, [pod red. Tchórzewski J., Barczak A., Barański M., Rozwadowski L.], Wydawnictwo WSR-P w Siedlcach, Siedlce 1993, pp. 131-177.
34. Trojanowski K.: Metaheurystki. Materiały pomocnicze do przedmiotu "Metaheurystyki – laboratorium" (Eng. Metaheurists. Auxiliary materials for the subject "Metaheuristics - laboratory"). Wyd. WSISiZ, Warszawa 2003, pages 80.
35. Wardziński K.: Przegląd algorytmów sztucznej inteligencji stosowanych w grach komputerowych (Eng. Review of artificial intelligence algorithms used in computer games), Homo communicativus. Filozofia – komunikacja– język – kultura (3 May): Kulturotwórcza funkcja gier. Cywilizacja zabawy czy zabawy cywilizacji? Rola gier we współczesności, pp. 249–263, 2008.
36. Wierzchoń S.: Sztuczne systemy immunologiczne. Teoria i zastosowania (Eng. Artificial immune systems. Theory and Applications). AOW EXIT, Warszawa 2001, pages 282.
37. Yannakakis G. N. and Togelius J.: Artificial Intelligence and Games. Springer, 2018.
38. Żurada J., Barski M., and Jędruch W.: Sztuczne sieci neuronowe: podstawy teorii i zastosowania (Eng. Artificial neural networks: basic theory and applications). WN PWN, Warszawa 1996, pages 375.