Controlling LEGO Linefollower Vehicle with Neural Networks

Jakub Maćkowiak[a], Sandra Świeczak[a] and Bartosz Wanot[a]

[a]Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44100 Gliwice, POLAND

Abstract

In the article we present our idea for learning system made for a LEGO robot. We built a line following robot which is using four sensors to trace the line. Track trace logic is based on our original implementation written in Python language for LEGO Mindstorm and on a neural network implemented using Python as well. As a result of our work we see that using neural network might be an effective way of teaching a robot to follow a line, but not in case of LEGO EV3.

Keywords

Artificial neural network, Lego blocks, line follower robot

1. Introduction

The fast pace of enrolling technology into our lives makes it much easier and more available to implement robotics for house needs. Especially those are a great opportunity for young engineers to learn programming and develop skills needed for creating advanced machines. There are many companies, that have tried to capitalize on that exact idea - teaching the youth about future technologies. Lego has also taken part in that competition, with their LEGO Mindstorms. The concept of combining robotics and LEGO bricks have revolutionise the market, making it extremely easy to produce any type of robot, limited only by the creators imagination. On company's official EV3 site there are many tutorials on how to create template robots and in that group line-following robot can be found. The idea behind the vehicle created and programmed for the article, was to check and improve the formula of self-driving robot, compiling our own code which includes sophisticated AI algorithm based on neural network trained with backpropagation.

There are many interesting applications in which neural networks work with images as classifiers. In [1] was presented special manager in federated learning to improve image processing. The model presented in [2] show how to work with neural networks to improve resolution of images and can be used for vritual immersion also [3]. There are also many important aspects of training algorithms. In [4, 5, 6, 7, 8, 9]. was presented how to select the best training algorithm to fit the data input describing objects.

ICYRIME 2021 @ International Conference of Yearly Reports on Informatics Mathematics and Engineering, online, July 9, 2021

jakumac927@student.polsl.pl (J. Maćkowiak);
sandswi038@student.polsl.pl (S. Świeczak);
bartwan560@student.polsl.pl (B. Wanot)

2. Related works

This article won't be revolutionary technologically-wise, since there have been many attempts of implementing a bit more complex algorithms into EV3 controlled LEGO robotics like fuzzy set rules [10, 11, 12, 13, 14, 15]. The use of additional code has in fact improved robots abilities in driving.

Many of the studies our research group has came through, are treating about a use of LEGO EV3 based machines in teaching robotics [16, 17, 18]. The ones in our reference section are slightly old, but the conclusion, that one can learn artificial intelligence implementation into robotics, using the same tools our group did, is still relevant, and it supports our statements from the introduction.

For the purpose of the article we have implemented feed-forward neural network [19, 20] trained by backpropagation algorithm [21].

3. Assumptions of the project

The assumption of the project is to build a vehicle controlled by an artificial intelligence system based on neural network. AI system must be able to set appropriate motoric power, dependent on position of a black line on a track, in each of four engines. The vehicle must not go off the track, and move as slightly as possible.

4. Robot construction

We have built a robot (Fig. 1) using LEGO Mindstorm elements. A vehicle has an EV3 Intelligent Brick, which has:

- a six-button interface
- a black and white display

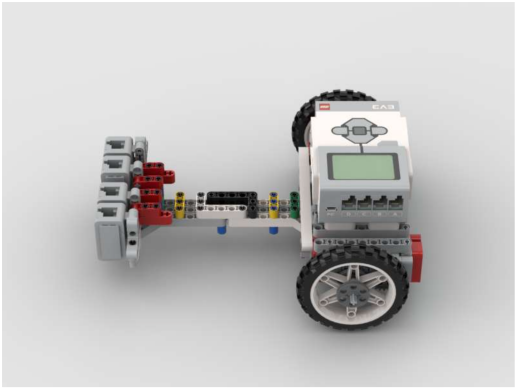

Figure 1: Linefollower vehicle.

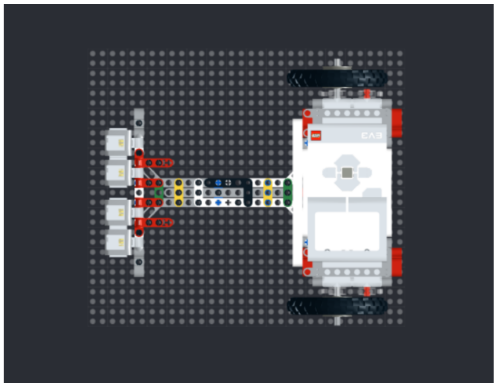

Figure 3: Top view of the vehicle.

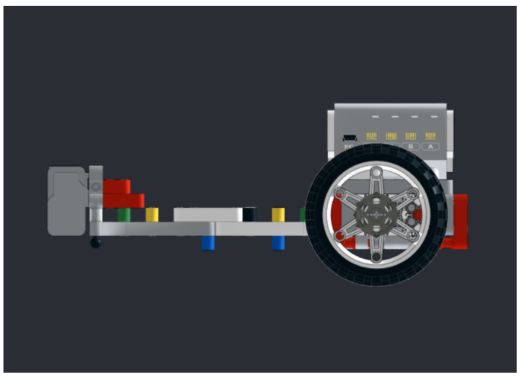

Figure 2: Side view of the vehicle.

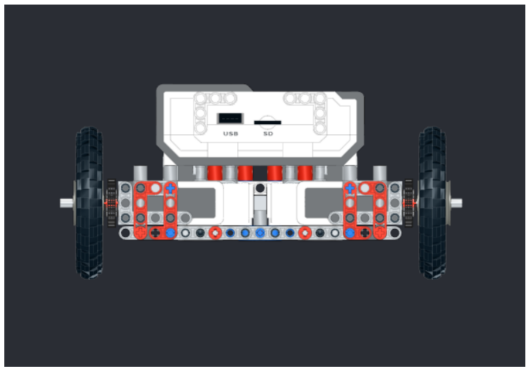

Figure 4: .Back view of the vehicle

- USB port
- a mini SD card reader
- a speaker
- four input ports and four output ports

It includes four EV3 Color Sensors which tell the difference between black and white and four EV3 Medium Servo Motors which allow robot to drive. A side plan of the vehicle is shown at Fig. 2, top view at Fig. 3 and back view at Fig. 4.

5. Mathematical Model

Our code for the machine is modular: we have started the code from a basic line follower with 4 engines (2 for each side) and 4 color detectors used as light sensors, also 2 for each side - their placement determines decisions the vehicle should make. We are going to present only right turns, as the left side related movements are just a mirror copy of the previously mentioned ones.

5.1. Normalization

The sensors are calibrated with a number from 0 to 100, where 0 is no light reflection (black) and 100 is the maximum value of reflected light. Due to the wear of the sensors, the data had to be normal- ized. The formula for the sensor 1 (see eq. 1) is similar for all sensors. For the right side, a sign of the equation changes.

$$SKR_1 = \frac{SKR_{MAX} \cdot (sensor1.reflection() - max_{cL})}{min_{c1} - max_{cL}} \quad (1)$$

SKR_{MAX} is maximum turn of the vehicle, *sensor.reflection()* is a value sent by the sensor and max_{cL} is a sum of maximum values of left-side sensors after calibration process (2). Analogue equation is for right-side sensors.

$$max_{cL} = max_{c1} + max_{c3} \quad (2)$$

Depending on the mode selected by the program, values of SKR_1, SKR_2, SKR_3 and SKR_4 are calculated or set to 0. Then *SKR* is a sum of all of them (eq. 3).

$$SKR = SKR_1 + SKR_2 + SKR_3 + SKR_4 \quad (3)$$

A final engine power is determined by an equation below (eq. 4). This equation is for left first engine, the rest of equations are similar with the change of a sign for right engines.

$$motorL1.run = MOC + SKR \quad (4)$$

5.2. Basis

Depending on the values read from the sensors, the vehicle can choose one of 6 options:

- Case1 (and-1)

  If one of the internal sensors notices less than 50% of possible light reflection, external sensor is starting to be taken into account. If it notice the black line, the vehicle turns until the other internal sensor sees the line. If sensor 2 has seen the line, sensor 4 is activated and if it has seen the line, the vehicle turns right until the sensor 1 sees the black line. Analogous situation is for left turn.

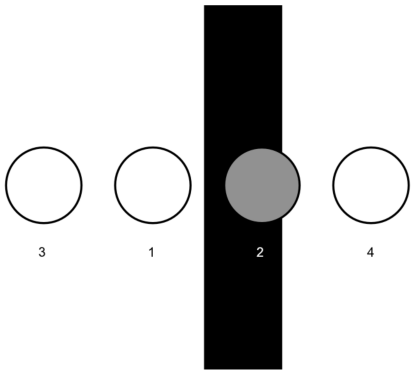

Figure 5: Start of the mode 1.

- Case2 (and-2)

  If three of the sensors see the black line, a vehicle has to go beyond a line and then turn toward sensors that have seen the line until an internal sensor on the other side see the line again. If sensors 1,2 and 4 have seen the line, the vehicle turns right until the sensor 1 sees the black line. Analogous situation is for left turn.
- Default

  The vehicle is going straight, the black line is between internal sensors.

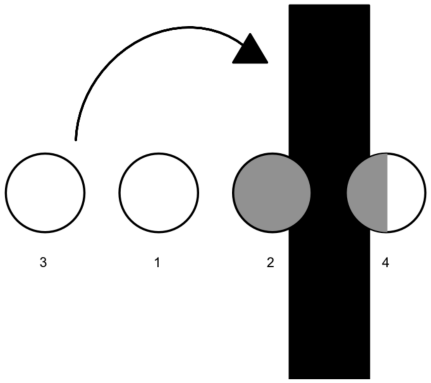

Figure 6: Turning moment in mode 1.

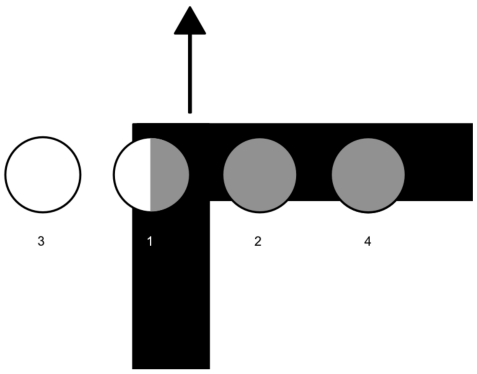

Figure 7: Start of the mode 2.

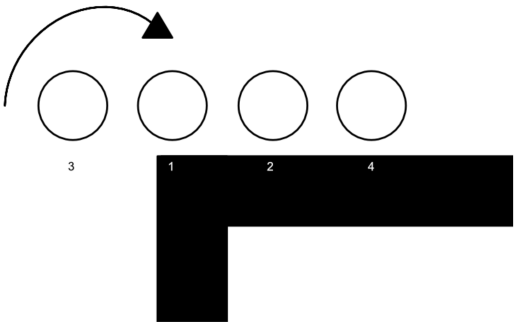

Figure 8: Turning moment in mode 2.

5.3. Neural Network

The assumption of the project was to create a neural network and train it so that the vehicle would run smoother than using the basic program. Our neural network is trained by backward propagation of errors which calculates the gradient of the error function and allow to

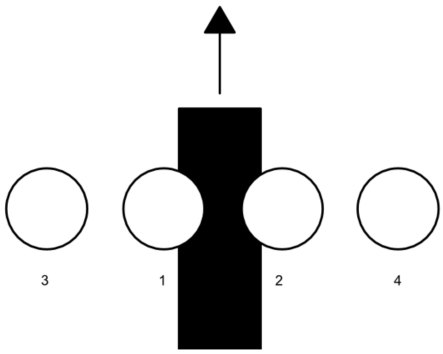

Figure 9: Default mode of the vehicle.

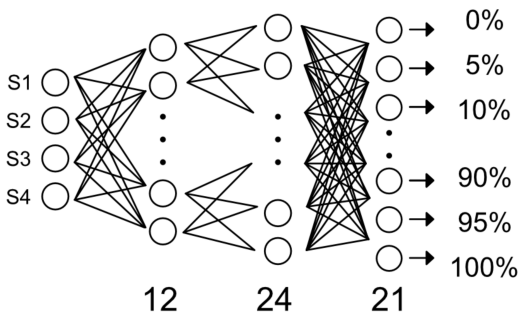

Figure 10: Architecture of the neural network.

change weights on nodes. The network (Fig. 10) has 4 neurons in an input layer, one for each of sensors and two hidden layers with 12 and 24 neurons. 21 neurons in an output layer represents percentages from 0 to 100%, every 5 percent. We create two neural networks, one for the left, and one for the right engines. A decision of neural network, a value of one of output

neurons is used in an equation below (eq. 5) to calculate percentage of maximum power of the engines which will change into appropriate turn. The equation is given for the left side.

$$MOCL = MAXMOC \cdot \frac{stronaL}{mianownik} \tag{5}$$

Where *StronaL* is an index of chosen output neuron multiplied by 5 to get a percentage value. *mianownik* depends on which neural network made the decision on the higher index.

6. Implementation

Program implementation is based on two parts:

- Basic program

  It is used for collecting data and creating a database. The program is also responsible for keeping the vehicle on the trace when it is not well trained yet.
  It contains initial values of calibrated sensors, maximum power engine and maximum turn rate. It also initializes all motors and sensors of the vehicle (Fig. 11).

```
SKRMAX = 100
MOC = 100

motorP1 = Motor(Port.A)
motorP2 = Motor(Port.B, Direction.COUNTERCLOCKWISE)
motorL1 = Motor(Port.C)
motorL2 = Motor(Port.D, Direction.COUNTERCLOCKWISE)
sensor3 = ColorSensor(Port.S3)
sensor1 = ColorSensor(Port.S1)
sensor2 = ColorSensor(Port.S2)
sensor4 = ColorSensor(Port.S4)
```

Figure 11: Default mode of the vehicle.

- Neural network program

  Neural network is used to compute motor speeds basing on sensor values (Fig. 12). It returnes table of results form output layout. The index of neuron witch returned the highest value is converted to percent of MAXMOC (Fig. 10).

```
def compute(self, s1, s2, s3, s4):
    tab = [s1, s2, s3, s4]
    tmp = []
    for nrWarstwy, warstwa in enumerate(self.ukryte):
        for nrNeurona, neuron in enumerate(self.ukryte[warstwa]):
            n = self.ukryte[warstwa][neuron]
            if nrWarstwy == 0:
                tmp.append(n.activate([tab[nrNeurona]]))
            else:
                tmp.append(n.activate(tab))
        tab = tmp
        tmp = []
    return (tab)
```

Figure 12: Compute function.

However, before riding the route becomes possible, network have to be well trained. To do this, we used backpropagation algorithm. Training takes place outside the robot, using database created by basic program (Fig. 13). After training, weights are exported from simulation to robot.

7. Experiments

7.1. Preparing

A test route for creating a database of sensors values and for training the vehicle with neural network was

```
def backErroring(self, expected):
    for nrWarstwy, warstwa in enumerate(reversed(self.ukryte)):
        #print (num, warstwa)
        a = list(self.ukryte.keys())

        for nrWagi, i in enumerate(self.ukryte[warstwa]):
            neuron = self.ukryte[warstwa][i]

            if nrWarstwy == 0:   ## Warstwa wyjścia
                neuron.errorList.append (expected[nrWagi] - neuron.active)

            else:
                err = 0
                nextLayer = a[a.index(warstwa)+1]
                for  j in self.ukryte[nextLayer]:
                    nextNeuron = self.ukryte[nextLayer][j]
                    err += nextNeuron.wagi[nrWagi] * nextNeuron.error
                neuron.errorList.append(err)

def weightsChange(self, learningRate):
    for nrWarstwy, warstwa in enumerate(self.ukryte):
            #print (num, warstwa)
            a = list(self.ukryte.keys())
            for nrWagi, i in enumerate(self.ukryte[warstwa]):
                neuron = self.ukryte[warstwa][i]
                if nrWarstwy != 0:
                    for indexWagi, waga in enumerate(neuron.wagi):
                        neuron.wagi[indexWagi] += (learningRate *
                            sum(neuron.errorList)/len(neuron.errorList) * neuron.i
                    neuron.errorList = []

def training(self, trainSet, side, repeats, learningRate):
    for _ in range (0, repeats):
        for el in trainSet.iloc:
            self.compute(el["s1"], el["s2"], el["s3"], el["s4"])
            wzor = []
            for i in range (0,21):
                wzor.append(0)

            y = ceil(el[side]/0.05)
            wzor[y] = 1

            self.backErroring(wzor)
        self.weightsChange(learningRate)
```

Figure 13: Training functions.

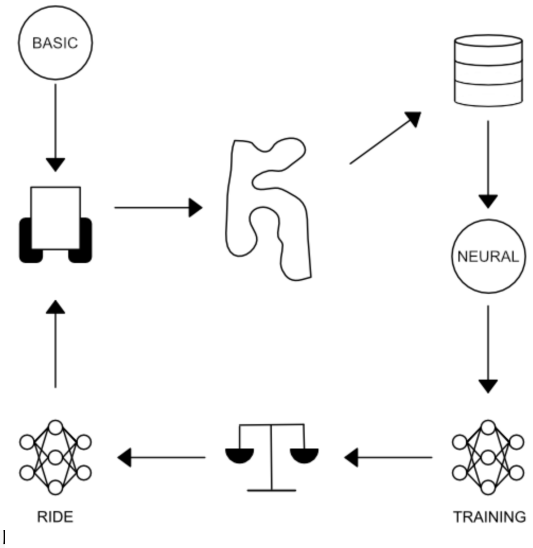

Figure 14: Flow chart of the algorithm and flow of data.

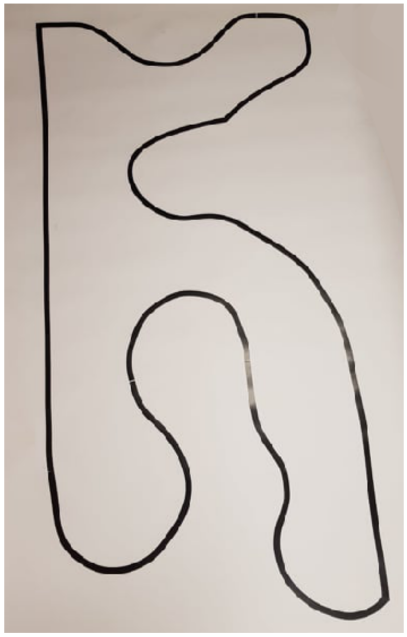

Figure 15: A test route for the vehicle.

made with white photographic paper and a black tape. The trace is shown at Fig. 15 below. First we needed a database with sensors values and turns calculated by the basic program. The vehicle collected data (shown at Fig. 16) which was sent to the program with neural network. Neural network has been correcting weights and learning to make appropriate decisions on the trace. The vehicle went through the trace twice and gave us the database containing almost 10000 records (Fig. 17). All of the records were normalized.

7.2. Neural network

First our neural network was supposed to look different which is shown at Fig. 18. We wanted to create only one network with smaller hidden layers. Output layers was supposed to contain only two neurons, one for each wheel of the vehicle, which were supposed to return velocity of a motors. This method was not optimal and has been given inappropriate values. We change a concept to build a much larger neural network. The new concept is given much better results. After giving random values in nodes weights, before training the network, a sum of errors was enormous, but after training with backpropagation, errors were almost completely reduced. Below you can see the sum of errors of all neurons before and after training the network (Fig. 19 and Fig. 20). Neural network with backpropagation algorithm allowed to get really satisfying results, at least in the simulation. But calculations are too slow and the vehicle is not keeping up with making decisions. There are huge delays so that the vehicle runs mainly under the control of the basic

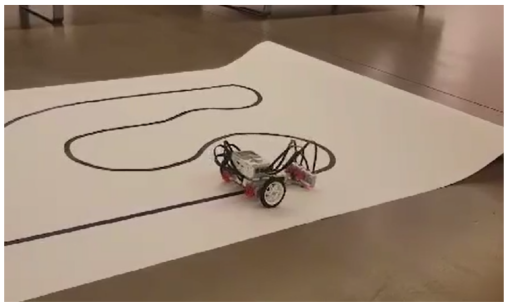

Figure 16: The vehicle in the process of creating database.

	s1	s2	s3	s4	L	P
0	0.968750	0.966292	1.000000	0.977011	0.000000	0.000000
1	0.968750	0.966292	1.000000	0.977011	0.000000	0.000000
2	0.958333	0.966292	0.986842	0.977011	0.010345	0.011952
3	0.968750	0.966292	1.000000	0.977011	0.010345	0.027888
4	0.968750	0.977528	1.000000	0.977011	0.179310	0.561753
...	...	...	...	...	...	...
9702	0.927083	0.921348	0.855263	0.862069	0.413793	0.454183
9703	0.916667	0.910112	0.815789	0.873563	0.337931	0.358566
9704	0.947917	0.921348	0.815789	0.873563	0.375862	0.382470
9705	0.947917	0.921348	0.868421	0.885057	0.331034	0.454183
9706	0.937500	0.921348	0.868421	0.885057	0.279310	0.442231

Figure 17: Fragment of the database.

program, which helps to go back to the line, after getting off the track.

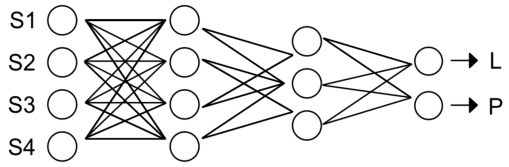

Figure 18: First concept of neural network.

8. Conclusions

Despite the very promising simulations, because of the delays and hardware limitations we cannot say that our robot passed the route using artificial neural network. Program requires optimization and some changes. What is more, an EV3 system would need more RAM memory and better CPU to work properly.

Fortunately, not all of our work was in vain. During the experiments we have found out that neural network

```
xL = Siec(struktura)
xL.erroring(bazaPrep, "L")
```

```
97066.61954397813
```

```
xP = Siec(struktura)
xP.erroring(bazaPrep, "P")
```

Figure 19: Errors before training the network.

```
xL.erroring(bazaPrep, "L")
```

```
6.999996465916024
```

```
xP.erroring(bazaPrep, "P")
```

```
2.99999238384742
```

Figure 20: Errors after training the network.

is doing much better with classifying using more neurons on output layout, than returning precise results.

References

[1] D. Połap, M. Woźniak, Meta-heuristic as manager in federated learning approaches for image processing purposes, Applied Soft Computing 113 (2021) 107872.

[2] X. Liu, S. Chen, L. Song, M. Woźniak, S. Liu, Self-attention negative feedback network for real-time image super-resolution, Journal of King Saud University-Computer and Information Sciences (2021).

[3] D. Połap, K. Kęsik, A. Winnicka, M. Woźniak, Strengthening the perception of the virtual worlds in a virtual reality environment, ISA transactions 102 (2020) 397–406.

[4] M. Wozniak, J. Silka, M. Wieczorek, M. Alrashoud, Recurrent neural network model for iot and networking malware threat detection, IEEE Transactions on Industrial Informatics 17 (2021) 5583–5594.

[5] M. Woźniak, D. Połap, C. Napoli, E. Tramontana, Graphic object feature extraction system based on cuckoo search algorithm, Expert Systems with Ap-

plications 66 (2016) 20 – 31. doi:10.1016/j.eswa.2016.08.068.

[6] M. Woźniak, M. Wieczorek, J. Siłka, D. Połap, Body pose prediction based on motion sensor data and recurrent neural network, IEEE Transactions on Industrial Informatics 17 (2020) 2101–2111.

[7] G. Capizzi, G. Lo Sciuto, C. Napoli, E. Tramontana, M. Woźniak, A novel neural networks-based texture image processing algorithm for orange defects classification, International Journal of Computer Science and Applications 13 (2016) 45–60.

[8] G. Capizzi, G. Lo Sciuto, C. Napoli, R. Shikler, M. Wozniak, Optimizing the organic solar cell manufacturing process by means of afm measurements and neural networks, Energies 11 (2018). doi:10.3390/en11051221.

[9] G. De Magistris, S. Russo, P. Roma, J. Starczewski, C. Napoli, An explainable fake news detector based on named entity recognition and stance classification applied to covid-19, Information (Switzerland) 13 (2022). doi:10.3390/info13030137.

[10] K. Grzesica, J. Wadas, Fuzzy system as a method of controlling lego linefollower vehicle using c programming language, in: SYSTEM 2020: Symposium for Young Scientists in Technology, Engineering and Mathematics, CEUR-WS, ????, pp. 9–15.

[11] N. Brandizzi, V. Bianco, G. Castro, S. Russo, A. Wajda, Automatic rgb inference based on facial emotion recognition, volume 3092, 2021, p. 66 – 74.

[12] M. Akmal, N. Jamin, N. A. Ghani, Fuzzy logic controller for two wheeled ev3 lego robot, in: 2017 IEEE Conference on Systems, Process and Control (ICSPC), IEEE, 2017, pp. 134–139.

[13] N. Azlan, F. Zainudin, H. Yusuf, S. Toha, S. Yusoff, N. Osman, Fuzzy logic controlled miniature lego robot for undergraduate training system, in: 2007 2nd IEEE Conference on Industrial Electronics and Applications, IEEE, 2007, pp. 2184–2188.

[14] G. Capizzi, C. Napoli, S. Russo, M. Woźniak, Lessening stress and anxiety-related behaviors by means of ai-driven drones for aromatherapy, volume 2594, 2020, p. 7 – 12.

[15] N. N. M. Khairi, S. S. S. Ahmad, The effectiveness of lego mindstorms nxt in following complicated path using improved fuzzy-pid controller, International Journal of Innovative Science and Research Technology 2 (2017) 155–161.

[16] M. Carbonaro, M. Rex, J. Chambers, Using lego robotics in a project-based learning environment, The Interactive Multimedia Electronic Journal of Computer-Enhanced Learning 6 (2004) 55–70.

[17] A. B. Williams, The qualitative impact of using lego mindstorms robots to teach computer engineering, IEEE Transactions on Education 46 (2003) 206.

[18] R. Brociek, D. De Magistris, F. Cardia, F. Coppa, S. Russo, Contagion prevention of covid-19 by means of touch detection for retail stores, volume 3092, 2021, p. 89 – 94.

[19] A. Krizhevsky, I. Sutskever, G. E. Hinton, Gradient-based learning applied to document recognation, Commun. Acm 60 (2017) 84–90.

[20] G. Lo Sciuto, G. Capizzi, S. Coco, R. Shikler, Geometric shape optimization of organic solar cells for efficiency enhancement by neural networks, Lecture Notes in Mechanical Engineering (2017) 789–796. doi:10.1007/978-3-319-45781-9_79.

[21] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.