Low-Cost Autonomous Trains and Safety Systems Implementation, using Computer Vision

Dan Andrei Suciu[1], Eva-H. Dulf[1] and Levente Kovács[2]

[1] Technical University of Cluj-Napoca, Faculty of Automation and Computer Science, Department of Automation and Applied Informatics, George Barițiu 26-28, 400027,Cluj-Napoca, Romania; suciu.fl.dan@student.utcluj.ro, eva.dulf@aut.utcluj.ro

[2] Óbuda University, Physiological Controls Research Center, Bécsi út 96/b, 1034 Budapest, Hungary; Kovacs.Levente@nik.uni-obuda.hu

Abstract: The need of modern transport solutions is a tendency that has been developed also in the railway transport. This study provides a possible implementation of a fully autonomous train system with low impact on the railway infrastructure, using computer vision and machine learning concepts. It could be implemented on various existing safety and infrastructure systems. The system has been tested on a H0 scale modified model train and a Raspberry Pi with a Pi Camera as processing unit. The proposed system combines several software and hardware technologies into a single embedded system that provide the required safety on railways and can set the trend for real trains. Furthermore, the main motivation of the concept is that the railway transport automation represents an essential step in transforming this domain into one as flexible as road transport. In this regard, over the years, a multitude of control and safety assurance systems, based on various technologies have been developed to lead to the most optimal outcome. The primary innovation of the study resides in the application of neural network quantization to enhance temporal efficiency, alongside the advancement of a comprehensive autonomous railway transportation system.

Keywords: automation; computer vision; European Rail Traffic Management System; machine learning; railway; quantization; safety.

1 Introduction

In recent years, there has been an observed “worldwide” trend towards the automation of the public transport [1] [2], especially the ”green” alternatives. The railway is one of the most important transport modes because of the high safety standards compared to the road transport. The number of accidents on railways is significantly lower, according to the latest studies. The main cause of railway accidents, is the human error [3], made by the railway workers and also external factors (for instance the drivers that do not respect the level crossings rules [4]). For this reason, the majority of the railway accidents could be prevented by the automation of the safety and control systems.

The main objective of the concept described is the reduction of the error from the modern railway transportation system, by limiting the human intervention [3], using modern solutions of image processing and hardware integration. Furthermore, this study attempts to standardize the different existing railway safety systems into a unique solution based on European Rail Traffic Management System (ERTMS) [5] structure, described in the following sections of this paper. The proposed system has a flexible and universal character due to the fact that it could be adapted to operate on different types of trains. In addition, the system maintenance costs and complexity are low because of the small number of elements involved in the process of automation and due to the use of already existing infrastructure. The main sections described in this paper are the software and hardware implementations of the proposed systems and the analysis of the performance by testing at H0 scale.

2 Related Work

2.1 Railway Safety Systems

There are numerous types of safety elements or systems implemented in different countries [6]. Only a few of these systems are compatible with each other and this could create problems regarding the international trains that must be driven by different operators and hauled by local locomotives. This argument highlights the need of a standardized system [7] for compatibility assurance between the infrastructure of different countries. The most efficient standardized system already implemented is European Rail Traffic Management System (ERTMS). The system includes two main components: European Train Control System (ETCS) and Global System for Mobile Communications – Railway (GSM-R) [5] [7]. ETCS includes 4 automation levels, implemented with different types of railway infrastructure elements, software solutions and communication protocols. The system is described in Figure 1.

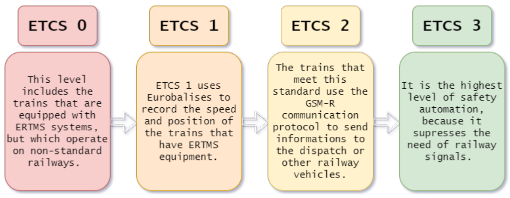
ETCS 0
This level includes the trains that are equipped with ERTMS systems, but which operate on non-standard railways.
ETCS 1
ETCS 1 uses Eurobalises to record the speed and position of the trains that have ERTMS equipment.
ETCS 2
The trains that meet this standard use the GSM-R communication protocol to send informations to the dispatch or other railway vehicles.
ETCS 3
It is the highest level of safety automation, because it supresses the need of railway signals.

Figure 1
ETCS levels description

The second component of ERTMS is GSM-R, which is the main communication protocol used by ERTMS trains to share information with other trains and the railway traffic control system. [8] The GSM-R platform offers features such as Voice Broadcast (VBS), Group Calling (VGCS) and emergency call pre-emption, all of which contribute significantly to the safety levels required by the standard. The main disadvantage of this system is the necessity of the human presence in the train, because it cannot ensure a maximum traffic safety.

2.2 Railway Automation Systems

There are many systems that contributes to different levels of railway automation: machine learning and neural networks algorithms, radar, GPS, Monorail, CAN, etc. All these systems could be interconnected to increase the redundancy and the levels of safety of the driverless trains as:

- Radar - It registers the position of a train trough the railway stations it passes. The system cannot be considered as real time, because of the distance between the stations. [9]
- GPS - The GPS system is used to determine the exact speed and position of the trains that are present on a railway section. Compared to Radar it is more suitable for railway systems because of the low error levels. [9] [10]
- Internet of Things (IOT) - This concept is represented in railway automation by the microelectromecanic (MEMS) sensors which could register the exact position of a train by the rail vibrations. [2] [11]
- Artificial Intelligence (AI) - It is the latest approach to railway automation. The AI railway automation systems are currently unstable but there are already many prototypes that could be used on real trains, with human presence redundancy. Line detection and railway traffic signals detection algorithms have been developed to ensure the safety of the AI systems used in train automation. The main limitations of these systems are the low resolution of the acquisition equipment or the necessity of powerful and expensive computers that could run the neural networks and the high complexity processing algorithms. [2]
- Other technologies - One of the most important aspects of the railway automation is the necessity of good communication between trains and dispatch. One of the best technologies proposed for railway automation is 5G, because it ensure a better broadband communication. With the help of Non-Orthogonal Multiple Access (NOMA) technology, the system could be shared between multiple users simultaneously. [2] [12]

3 Design and Implementation

The system proposed contains two main parts: hardware and software implementation. The hardware component consists of all the physical modules that should be designed and mounted on the railway infrastructure and on the autonomous trains. The software part implements the communication between the hardware components and the functionality of all the systems and automated tools that ensure the safety of the trains traffic and those that helps the railway vehicles to move. All the components are implemented on the H0 scale model, but they could be extended to real trains.

3.1 Hardware Implementation

The main hardware components are presented in Table 1. In addition to the elements presented, there have also been used other parts, such as transistors and relays. The main power source is a 12 V supply, connected to the rail tracks. The locomotive DC motor is powered by the voltage collected by the train wheels from tracks, via an H bridge, used for controlling the speed and the direction of movement. The H bridge is controlled by the Raspberry Pi board via General Purpose Input/Output (GPIO) pins.

Table 1

Hardware components

NR.	Hardware component	Description
1	NodeMCU Board	Used as base station (dispatch). It reads the sensors and controls the switches
2	Raspberry Pi 4B Board	The main board mounted on the train, used for data processing and control.
3	Raspberry Pi Camera	Used for data acquisition
4	Reed Sensor	Used as train presence sensor. Mounted on the track; Triggered by permanent magnet.
5	74HC4067 Multiplexer	Used for multiplexing the inputs and outputs of the NodeMCU board

In Figure 2 the schematic diagram of the main hardware components and the physical connections between them is presented. The hardware system is composed of two parts: the fixed component, represented by the trackside base station circuit that is controlled by NodeMCU and the mobile component, mounted on the train that is managed by Raspberry Pi.

Due to the small number of Input/Output (I/O) pins of the NodeMCU board, there are necessary two multiplexers to increase the amount of the sensors or the relays that could be interfaced with the controller.

On the bottom of the locomotive a permanent magnet is attached, so when the train will pass over the reed sensor that is fixed on the track, it will send a short impulse towards the NodeMCU controller to notify that the line is busy. This operation mode is similar to the Eurobalises used in the ERTMS system described in section 2.1. Technically, both systems are used to determine the exact location of a train, information that can later be used to control rail traffic and avoid collisions. The communication between NodeMCU and Raspberry is made wireless. The communication process is described in section 3.2.2. All other electronic components are wired to NodeMCU, respectively to Raspberry board. The NodeMCU board is powered from a 5 V supply and the relay interface board is connected to the main power source (12 V). The Raspberry Pi board is interfaced with the Pi camera and the H bridge circuit used to control the locomotive parameters and it is powered from a LM2596 step-down module, used for converting the voltage acquired from the train wheels on the rails (12 V) into a 5 V supply.

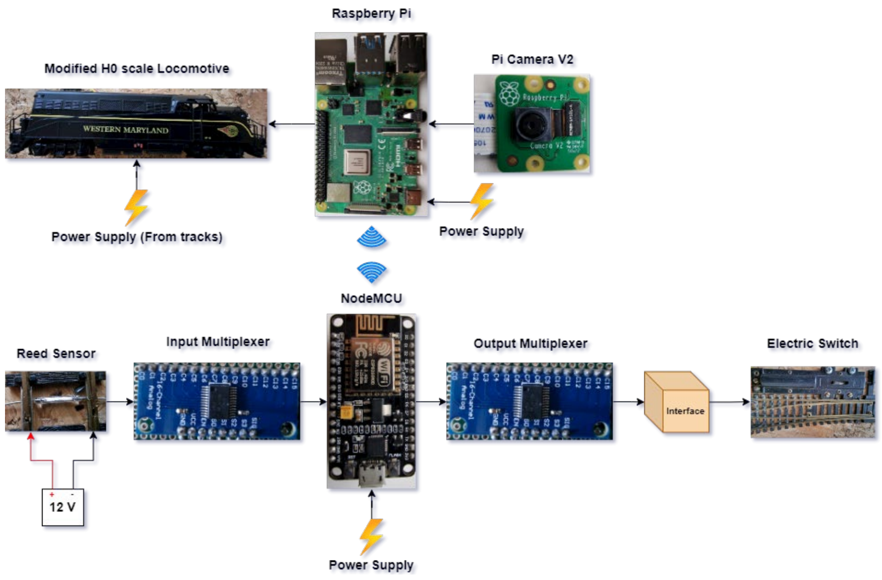

Figure 2

Hardware implementation diagram – components

3.2 Software Implementation

The software component of the system includes the control of safety elements, traffic management on the network, speed control, image processing and the sign recognition algorithm as well as the development of the neural network structures used in these processes. For this reason, the software component represents the main part of the system, as it is responsible for controlling and managing all the hardware components described in section 3.1. This section could also be divided in two parts:

the NodeMCU code (Arduino C++ language) and the Raspberry Pi code (Python programming language), whose characteristics are described below.

3.2.1 NodeMCU Code:

The main purpose of the NodeMCU board is to interpret the values of the reed sensors from the system and to control the position of the railway switches along the simulated route. Thus, NodeMCU could be considered the dispatch which manage the railway traffic and ensure the safety of the entire system. It checks the presence of the trains on the tracks. In this way, it can determine the approximate position of every train and redirect the others, with the help of the railway electronic controlled switches, on the most suitable route to avoid the possible accidents.

To conclude, the NodeMCU controller ensures the safety, the routing and the scheduling of the trains that are operating within the local railway network. This mode of operating could be compared with the main principles used in the standardized ERTMS system, described in section 2.1 and ensures the real time nature, one of the most important characteristics of the railway systems. One of the main reasons for which the nodeMCU board was chosen, is the high performance relative to the low price of this device and the flexibility of the communication channels. The possibility of using the wireless internet connection is also a mandatory aspect for the proper functioning of the system.

The function diagram of the code developed for NodeMCU board is presented in Figure 3.

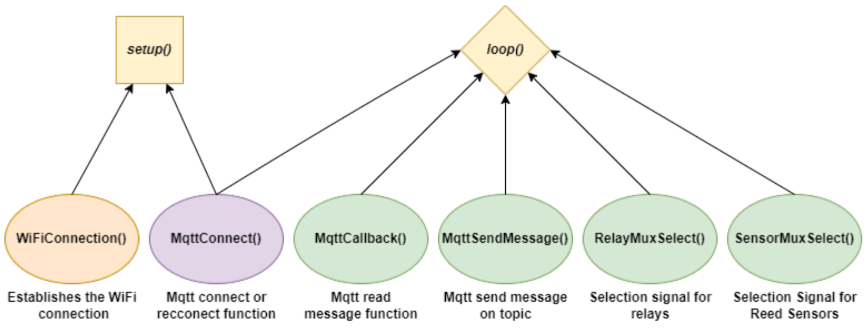
setup()
loop()
WiFiConnection()
MqttConnect()
MqttCallback()
MqttSendMessage()
RelayMuxSelect()
SensorMuxSelect()
Establishes the WiFi connection
Mqtt connect or recconect function
Mqtt read message function
Mqtt send message on topic
Selection signal for relays
Selection Signal for Reed Sensors

Figure 3
NodeMCU code – functions diagram

The ”setup()” and ”loop()” functions, are the mandatory functions of the code, where all other functions are called. The code is composed of two parts: the controller component and the wireless communication implementation. The controller segment sets the inputs and the outputs used for the sensors and the switches and it controls the two multiplexers by the selection signal. The values read and written are manipulated in the infinite loop section of the code for transmitting the commands to the raspberry pi controller via Message Queuing

Telemetry Transport (MQTT or mosquito) communication protocol presented in 3.2.2 section. The communication section implements the wireless connection and sets up the custom MQTT commands that are sent wirelessly to the Raspberry Pi controller.

3.2.2 Raspberry Pi Code:

The Raspberry Pi board represents the ”brain” of the locomotive and the main processing unit. It is the interface between the external environment of the autonomous train and the internal railway management and control system, due to the Pi Camera that is connected to it. The purpose of the python code written on Raspberry is to process the images captured by the camera and to classify the data by the predefined labels and then to send the information extracted to the base station via MQTT protocol. In addition, Raspberry must control the DC motor of the locomotive by the GPIO pins and the attached H bridge circuit depending on the requests made by the base station.

The essential part of the code is the machine learning algorithm. The neural network developed was built on a more powerful processor than the Broadcom BCM2711, Quad core Cortex-A72 (1.8 GHz) chip with which Raspberry Pi board is equipped, because of the better speed and performance obtained while training and testing the AI model. The processor used was Intel Core i7 - 10750H CPU (2.60 GHz). Two versions of the model were developed and tested to analyze the performance in context of processing and classification speed and accuracy values obtained after training and validation of the neural network. The first version is a convolutional neural network (CNN) and the second one is a modified quantized model. The quantized network was built based on the first model.

The first model is a sequential convolutional neural network that contains three convolutional layers (from which, two are hidden), each one followed by a max pooling layer and two different fully connected layers. The structure of the neural network described is presented in Figure 4. The reason for using a quantized neural network is to improve the performance on Raspberry Pi. The Broadcom BCM2711 CPU is a low performance processor and the memory of the board is small, therefore the code could work slow if all operations are performed in floating point. Thus, the quantization can be a solution to increase speed. The principle of quantization method is the transformation of the floating-point values that are manipulated by the network into lower precision inputs.

This process can lead to a higher speed, but the disadvantage is that the accuracy will decrease significantly, depending on the complexity of the calculus that should be performed.

In conclusion, the quantized network will have the same structure as the original network, but the difference is the representation of values in the board memory and the background operations execution mode. The comparison between the performances obtained by the two networks will be performed in chapter IV.

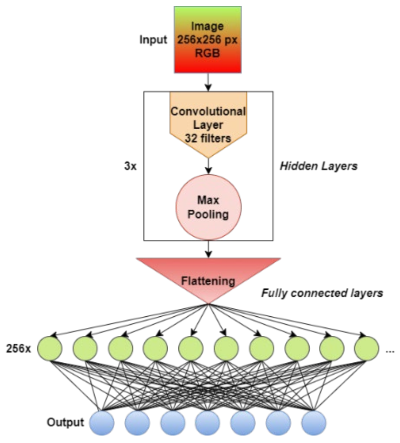

Figure 4
Convolutional neural network – structure diagram

The Pi camera that is mounted on the locomotive send the frames of the live video as inputs to the neural network and the output is the corresponding label of the railway signal detected. The training data was collected with the same camera and was labelled with the help of Keras Tensorflow tool. There are seven H0 railway signs that are processed by this network, but there can be added more track signals. After this process, the label obtained is sent via MQTT to NodeMCU board, that take the matching decision depending on the signal detected. Message Queuing Telemetry Transport (MQTT or mosquito) communication protocol is one of the Internet of Things (IoT) transmission methods, used for message exchange between two boards. Depending on the configuration, it can be unidirectional or bidirectional, but there will be always a publisher-subscriber relation between the entities. The main component is the broker, in this case, the Raspberry pi board, that manage the entire communication system and connects all the subscribers to the network. In this case the Raspberry pi board works as broker and also as one of the clients. The other client is NodeMCU. The principle of communication is based on topics. All the clients that have subscribed to a topic will receive the data published on that topic. In this case, the data transmitted is composed of a byte, because there are seven railway signs. Every bit from the byte represents a railway signal. When a bit value is ”1”, it means that the sign corresponding to the bit position in the byte was detected.

After the NodeMCU board receives the information from the locomotive, it sends back the corresponding commands to Raspberry Pi, also through MQTT. The DC motor speed and direction is then adjusted to ensure the proper operation of the train. The MQTT communication principal diagram is shown in Figure 5. The reasons for choosing Raspberry Pi were its small size, low price and the possibility of simple integration in the system. Also, the compatibility with others IoT platforms was a decisive aspect for choosing this board. In addition, the fact that Python is the main language for Raspberry is an advantage regarding the flexibility, simplicity and the high possibilities when it comes to high level machine learning technology.

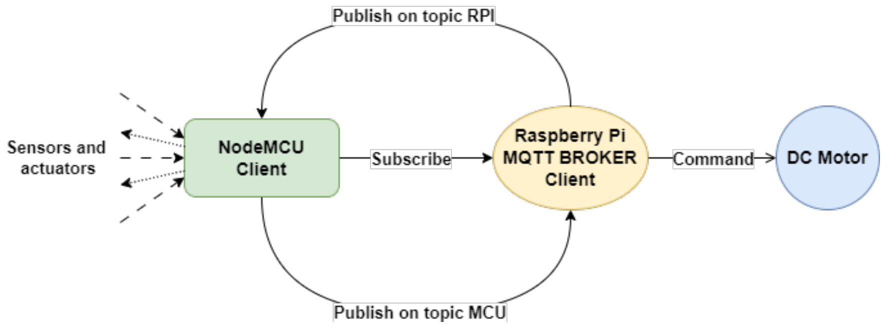

Figure 5
MQTT communication principal diagram

4 Experimental Results

In this section, a comparison will be made between the two neural networks described in section 3.2.2. The main characteristics that will be analyzed concerns the speed of the neural network and the accuracy of the results returned after the training, testing and validation of the algorithm.

In the case of the H0 scale model, the speed of the software and hardware components does not affect too much the performance of the entire system, but measured on real trains, the time is decisive, because of the higher risks when it comes to delays in the control process. Therefore, the hardware and software components must be optimized to the highest level in order to ensure the railway safety required by the proposed standard.

In addition, the hardware performance is analyzed in terms of the system's reaction time, an essential aspect in railway traffic control. The primary aspect to be analyzed, is the response time of the hardware components responsible for ensuring the safety and organization of rail traffic.

4.2 The Initial Convolutional Neural Network Performances

The input of the neural network is a 255 × 255 pixels scaled color picture, so the real data instance dimensions will be 255 × 255 × 3. For the optimization of the network build process, the data was labeled and randomly grouped into three groups: Training, Validation and Testing. The initial data (576 images sorted into 7 categories) is divided into batches of 32 images. The training data consists of 10 batches, the validation data of 5 batches and the test data of 3 batches. It is necessary that the number of training and validation data is greater than the number of test data to ensure the performance of the network and to avoid overfitting.

The test batch is used only to confirm the validation accuracy and it will not be used in the last version of the neural network for memory usage minimization. The RELU (rectified linear unit) activation function (1) is used for the first three layers and for the first fully connected layer. For the output layer, the sigmoid activation function (2) is used.

$$f(x) = \begin{cases} x, if\ x > 0 \\ 0, otherwise \end{cases}, where\ x - input \tag{1}$$

$$f(x) = \frac{1}{1+e^{-x}}\ , where\ x - input \tag{2}$$

The optimizer used for the minimization of the loss function is Adam. The Adam optimizer is an extended version of the stochastic gradient descent method and it provides a much higher performance in terms of number of iterations. The Adam optimizer use two moving averages to estimate the iterations:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{3}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{4}$$

In the presented relationships (3) and (4), *m* represents the "momentum" element, which is an instantaneous calculation of the gradient mean, while v denotes the squared sum of previous gradients. The terms β_1 and β_2 are the hyperparameters of the Adam optimization method, and these elements represent the main novelty compared to other gradient methods. The g_t element, which appears in all three equations, signifies the gradient of the current optimization step applied to the error function.

The loss and accuracy graphs for the neural network training are shown in Figure 6, respectively in Figure 7. The logarithmic shape of the loss and accuracy curves brings out the decrease in error over time.

The overall accuracy of the neural network is 97.014% and the average loss is 0.071. The average execution time while training is 23 seconds (2 second/step) and the average time during inference is 2.362 seconds (135 milliseconds/step).

Following the analysis of loss and accuracy graphs for both the initial neural network and the quantized one, a relevant comparison of the performance of the two automation solutions can be conducted, as described in section 4.4.

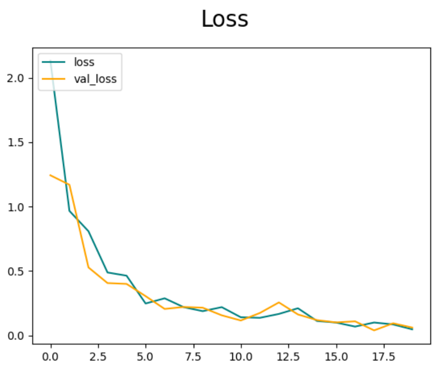

Figure 6
Original network loss graph

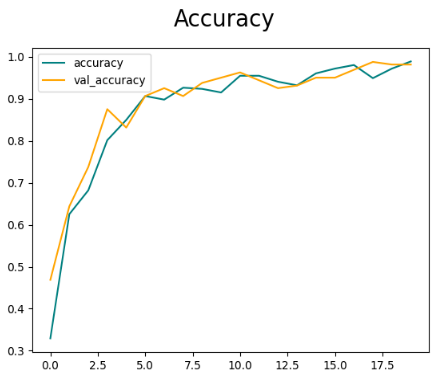

Figure 7
Original network accuracy graph

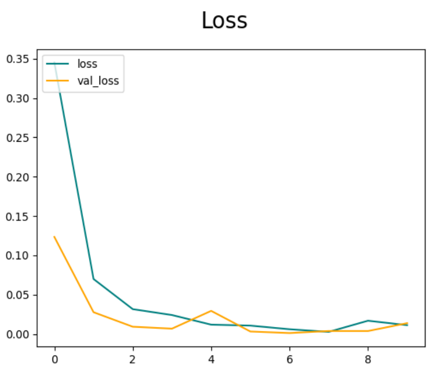

Figure 8
Quantized network loss graph

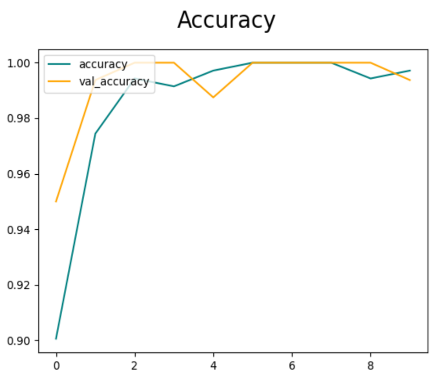

Figure 9
Quantized network accuracy graph

4.3 The Quantized Neural Network Performances

The quantized neural network was obtained by modifying the original neural network using an adapted version of the Learned Step Quantization (LSQ) method [13, 14]. The LSQ method adjusts the scaling factor of each layer of a neural network during training. Thus, by tuning the parameters of the method, it can help the model to run faster, by reducing the number of bits used to represent the weights dynamically, while training the neural network. The main parameters of the method are: the number of quantization bits n, the scale factor q (it will change dynamically after each layer), the incremental shift, that is the value with which the scale factor is modified and ϵ which is a very low number used to avoid the division by 0 when the scale factor is computed.

Equation (5) represents the calculation formula of the current layer scale factor. The x factor represents the current layer weights vector.

$$q_i = \frac{|\max(x) - \min(x)|}{(2^n - 1) + \epsilon} \tag{5}$$

The method provides high performance, because it reduces only the dimensions of the weights set, maintaining the float32 representation and yet it obtains an approximate 10% reduction of the execution time. The overall accuracy of the quantized network is 98.507% and the average loss is 0.024. The average execution time while training is 15 seconds (1 second/step) and the average time during inference is 2.216 seconds (96 milliseconds/ step). The loss and accuracy graphs for the quantized network training are shown in figure. 8, respectively in Figure 9.

4.4 Comparison between the Two Networks

It can be observed that the performance graphs have similar shapes and the accuracy and the loss function values are comparable. It can also be observed that the quantized model is able to reach good values in less time (epochs) than the original one. In addition, the time required to compute the necessary calculations for training and inference is reduced in the case of the quantized network. In some cases, it has been observed that the inference accuracy was improved even if the precision of the initial network was reduced by the quantization algorithm. Table 2 presents all the performance characteristics of the two models for a better comparison.

Table 2

Presented neural networks comparison

Property	**Initial Network**	**Quantized Network**
Training Time	23 s (2 s/step)	15 s (1 s/step)
Inference Time	2.362 s (135 ms/step)	2.216 s (96 ms/step)
Training Accuracy	97.014%	98.507%
Inference Accuracy	90.453%	88.232%
Size on Disk	75.97 KB	37.961 KB

In conclusion, the quantized network has a higher performance than the original network and it fits better to be used on edge computing devices (like Raspberry Pi) which requires a lower memory usage, but the same performances. The method is also suitable for the implementation on real systems on the trains.

4.5 Hardware Performance

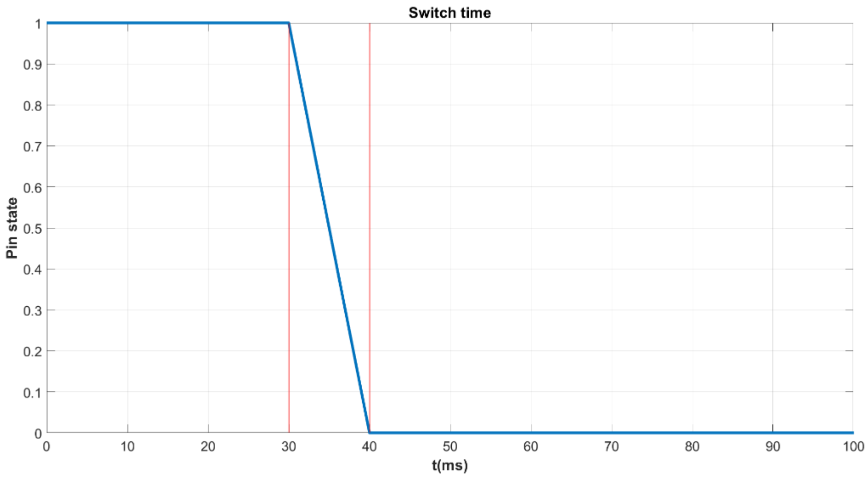

Figure 10
Switch Time

The hardware performance was evaluated by testing the speed of the component's responses. In the ideal case, the elapsed time since the command was initiated until the pin has changed the state from Low to High or vice versa, is 0. In reality it is very important that this time period is as short as possible, at the millisecond level. The time between the change of the output pin state and the moment of the relay switching is ignored, due to the very short period of signal transmission. The switching time can be observed between the two vertical red lines from Fig. 10.

After measuring the approximate switching time, it was concluded that the value of the commutation is under 10 milliseconds. Therefore, the hardware components show a sufficiently high performance to function on a real time system. In addition, the speed of the hardware components significantly contributes to the improvement of safety levels on the railway. In the case of real infrastructure, making simple modifications to both the physical components and traffic control systems can enhance the safety of future autonomous trains. This factor constitutes a notable cost advantage in terms of the impact on an automation project.

Thus, based on the analysis of software and hardware performance, it can be ascertained that the validity criteria of the study are met, paving the way for potential extensions in various developmental directions, as delineated in the concluding chapter. The final project board and the testing stand consisting of all hardware elements and railway traffic control and safety systems is presented in Figure 11.

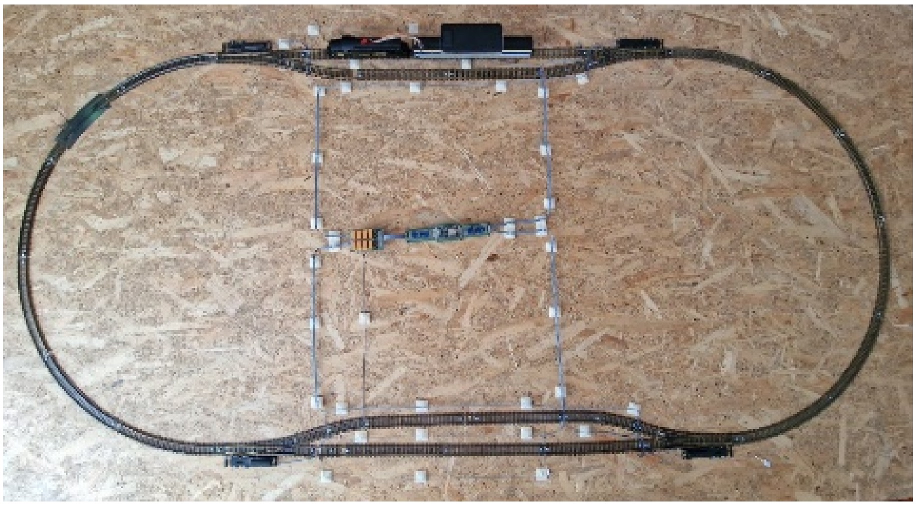

Figure 11
The final project board

Conclusions

The goal of the proposed system is to integrate the necessary hardware and software systems to control an autonomous train in maximum safety conditions by using image processing and classification algorithms, developed with the help of two neural networks. By comparing the two neural networks presented, the result was that the quantized network is more efficient and it ensures the good functionality on different edge devices. The system has been tested on an H0 scale train model and all the safety requests have been fulfilled.

In conclusion, the main goals of this paper were met. The proposed system manages to obtain higher performance than the existing systems by running on a quantized neural network and to integrate the hardware and the software components in a single low-cost implementation. The next goal is to extend the system on a larger scale and try other quantization methods for the neural network to improve even more the performances. The system can also be integrated in other railway safety environments, that are used in the non-standardized countries, without many changes to the infrastructure and with lower costs.

References

[1] MARKOVIC, Ljubo, et al. Evaluation of Alternative Solutions of General Design of Railway Lines with Regards to Environmental Protection, Acta Polytechnica Hungarica 19.6, 2022: 99-113

[2] SINGH, Prashant, et al. Deployment of autonomous trains in rail transportation: Current trends and existing challenges. IEEE Access, 2021, 9: 91427-91461

[3] YU, Guanhua, et al. Identification of significant factors contributing to multi-attribute railway accidents dataset (mara-d) using som data mining. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2018. pp. 170-175

[4] SINGHAL, Vivek, et al. Artificial intelligence enabled road vehicle train collision risk assessment framework for unmanned railway level crossings. IEEE Access, 2020, 8: 113790-113806

[5] RISPOLI, F.; NERI, A.; SENESI, F. Innovative train control systems based on ERTMS and satellite-public TLC networks. WIT Transactions on The Built Environment, 2014, 135: 51-61

[6] FLAMMINI, Francesco (ed.). Railway safety, reliability, and security: Technologies and systems engineering: Technologies and systems engineering. IGI Global, 2012

[7] DI MEO, Carlo, et al. ERTMS/ETCS virtual coupling: proof of concept and numerical analysis. IEEE transactions on intelligent transportation systems, 2019, 21.6: 2545-2556

[8] SNIADY, Aleksander; SOLER, Jose. An overview of GSM-R technology and its shortcomings. In: 2012 12th International Conference on ITS Telecommunications. IEEE, 2012. pp. 626-629

[9] RANI, V. Amala; AUSTALEKSHMI, TV Subha. Can protocol driverless train control system. May-IJRET: International Journal of Research in Engineering and Technology, 2014

[10] FILIP, Aleˇs; SABINA, Salvatore; RISPOLI, Francesco. A framework for certification of train location determination system based on GNSS for ERTMS/ETCS. International Journal of Transport Development and Integration, 2018, 2.3: 284-297

[11] ZHAO, Yuliang, et al. Continuous monitoring of train parameters using IoT sensor and edge computing. IEEE Sensors Journal, 2020, 21.14: 15458-15468

[12] ALSABA, Yamen, et al. 5G for Remote Driving of Trains. In: Communication Technologies for Vehicles: 15th International Workshop, Nets4Cars/Nets4Trains/Nets4Aircraft 2020, Bordeaux, France, November 16-17, 2020, Proceedings 15. Springer International Publishing, 2020. p. 137-147

[13] ESSER, Steven K., et al. Learned step size quantization. arXiv preprint arXiv:1902.08153, 2019

[14] ANDREI-ALEXANDRU, Tulbure; HENRIETTA, Dulf Eva. Low cost defect detection using a deep convolutional neural network. In: 2020 IEEE International conference on automation, quality and testing, robotics (AQTR). IEEE, 2020, pp. 1-5