International Journal
Information Technology Applications (ITA)
Volume 12, Number 1, September 2023

SAFETY ASSISTANCE SYSTEM USING CONVOLUTIONAL NEURAL NETWORKS

Ján Cigánek, Filip Žemla

Abstract:

The aim of this paper is to create a safety assistance system for a vehicle that is able to recognize dangerous situations while driving using the front camera. In addition to the detection of these events, the system is able to warn the driver using sound and visual effects to prevent a possible accident. The main principle of the work consists of creating and training a convolutional neural network that will be able to detect a set group of objects in images, videos or camera images captured in real time and to visualize these objects in the created application.

Keywords:

Neural network, security, vehicle, detection, assistance system, application.

ACM Computing Classification System:

Computing methodologies, Artificial intelligence, Computer vision, Image and video acquisition, Active vision.

Introduction

Various safety systems have been implemented in vehicles for several years. These systems are constantly innovating and coming up with new improvements. It is a trend whose main task is to increase the safety of the vehicle crew, but also to increase comfort during driving. This is an area of automotive technology that will constantly advance and will be implemented to a greater extent, especially in autonomous vehicles, the development of which is currently on the rise. In this area, safety systems that are able to detect dangerous situations and to assist in driving the vehicle are a key factor.

The most common risk when driving a vehicle is a collision with another object. In 2021, up to 64% of collisions from the total number of traffic accidents were caused by this reason. If safety systems used in vehicles were not developed, this number would increase every year due to the ever-increasing number of vehicles on the road [1].

The main task of this paper is to create a security system that will be able to detect objects around the vehicle while driving: that could endanger the vehicle; or objects that could be endangered by the vehicle itself. The system has to recognize the danger in real time and inform the vehicle driver about the detected problem. This warning consists of two parts: a visual part, which can be represented by a warning inscription depending on the type of detected object on the display in the vehicle or the monitor of the additional device; the second part of the driver's warning is an audio signal that sounds immediately after recognizing a dangerous object. Both warnings (warning text and warning sound signal) differ according to the type of danger.

This paper serves as a tool and demonstration of a security system based on convolutional neural networks used in the control unit of vehicle. It can also serve as a basis for later work, which would further develop and improve it according to current requirements.

This action could be performed thanks to a relatively simple extension of the system, which would be able to detect a larger number of objects or would contain other security elements.

1 Safety Assistance Systems in Vehicles

An important element on which great emphasis is placed in the development and production of cars is also their safety and the security systems associated with it.

Individual systems are required not only by customers, but also by laws in the countries for which the cars are manufactured. The main goal of introducing such systems is to protect the health of the driver and the entire car crew, but also to protect people who are around the vehicle.

Various types of safety systems are implemented in automobiles to achieve safety requirements. They are generally divided into two categories: active vehicle safety features and passive vehicle protection features [1, 2].

Active safety elements include the following systems [1, 2]:

- Good driving characteristics and optimal traction,
- Quality brakes and precise steering,
- ABS – a system preventing the wheels from locking when braking,
- ESP – helps stabilize the vehicle,
- BAS – braking effect assistant,
- Autonomous emergency braking,
- Driver fatigue monitoring,
- Lane keeping assistant,
- Airbags.

Passive safety elements include the following systems [1, 2]:

- Seat belts,
- Seat belt pretensions,
- Laminate and polycarbonate glasses,
- Active head restraints,
- FPS – a system interrupting the supply of electrical energy during an accident,
- Emergency rescue call.

Currently, the automotive industry uses a wide range of systems that are used to detect objects located either in the near or far vicinity of the vehicle. These systems are specially designed for different conditions such as fog, rain, driving at night or, conversely, high intensity of ambient light.

Initially, these systems were designed with the aim of increasing traffic safety, but with the advent of autonomous vehicles, their use has expanded even further. Without their use, the functioning of autonomous vehicles would be almost impossible, or very dangerous. Thanks to these systems, it is possible to use functions such as adaptive cruise control, lane control, reading traffic signs, automatic parking, a bird's-eye view of the car or automatic braking in front of an obstacle [3, 4].

We use the following systems to detect objects:

- Infrared systems (Fig.1),
- Radar systems (Fig.2),
- Camera systems.

Fig.1. Detection of people on the road using an infrared camera [4].

Fig.2. Demonstration of the operation using radar systems [2].

Despite the fact that every car manufacturer uses the same safety systems, such as reversing cameras, object detection, night vision or parking assistant, these systems differ from each other in many elements. This is due to the fact that most car producers or concerns develop their own systems and methods of how these systems work. Even in the case of using components from suppliers, which several car manufacturers may have the same, these systems often differ in the used software.

2 Neural Networks

Nowadays, there are many operations and situations that various devices and computers handle better than humans. These are, for example, complex mathematical calculations, creation of various animations or simulations. However, when it comes to ordinary thinking, imagination and thinking, humans are still ahead of computers. It is thanks to the inspiration of the human brain that artificial neural networks provide solutions that make computers more and more like people, and often these systems come up with their own solutions to problems.

Artificial neural networks are massively parallel computing models built on the basis of the properties and structures of biological neural networks. They have the ability to store information, which allows for further processing, while mimicking the human brain in gathering knowledge in the learning process. It can be said that it is a kind of artificial simulation of the brain.

Their goal is to arrive at a decision or final result based on a certain amount of input information. The ability to learn is a fundamental aspect of intelligence.

However, with artificial neural networks, learning cannot be understood in the same sense as with humans, but it is a continuous solution to a given problem, during which the knowledge of the network increases and its ability to solve the problem constantly increases [5].

Artificial neural networks consist of a large number of interconnected individual elements called neurons that work to solve a problem. These are groups of algorithms designed for recognizing and distinguishing various patterns and structures. Individual patterns and structures such as images, sounds or text are converted into the forms of numbers and vectors. In some cases, networks consist of several hidden layers in which data is processed, allowing this data to be explored in depth and to make connections between the acquired experiences of the network. The number of hidden layers can vary from one for normal neural networks to 200 or more hidden layers, in this case they are called deep neural networks [6].

The basic element of an artificial neural network is **a neuron** (Fig.3). It generally has several inputs from other neurons and only one output. The process of a neural transformation (inputs into outputs) is relatively simple, but a complex activity occurs in the connection of many neurons into one network.

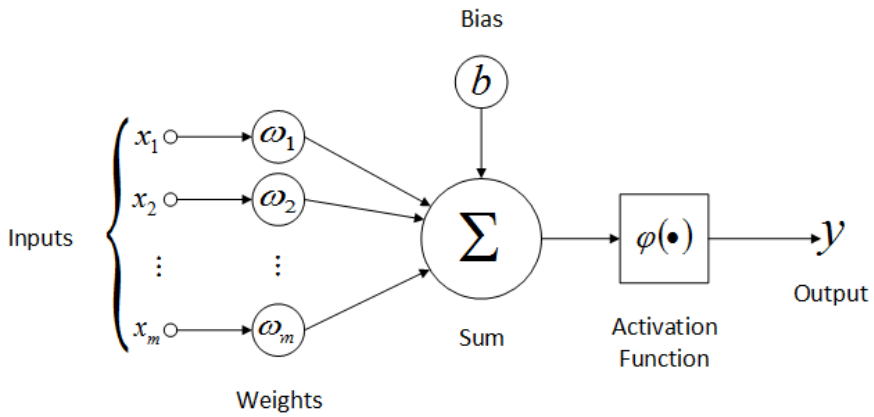

Fig.3. Model of a neuron [6].

In the vast majority of neural networks, neurons are arranged in individual layers. Such a division facilitates program implementation and also enables a mathematical description of the activity of the neural network. In practice, they are divided into two main groups according to the direction of signal propagation:

- **Forward neural networks** - called Feed-forward in English. The main essence of these networks is that the information and thus the signal spreads in only one direction, from the neurons of the input layer to the neurons in the hidden layers to the neurons located in the output layer.
- **Recurrent neural networks** - unlike forward ones, here the signal can also propagate from the neurons in the output layer to the neurons in the hidden layers and even to the neurons located in the input layer. In general, recurrent neural networks are much more complex than feedforward ones.

A. Perceptron

Perceptron is the simplest model of artificial neural network consisting of only one neuron and therefore it has one layer. It is a supervised learning algorithm that allows neurons to learn new knowledge based on input data. It belongs to the group of feed-forward neural networks [6, 7].

The most common implementation of the combination of feed-forward neural network with supervised learning is **Multi-layer perceptron** (MLP). The structure of such a network (Fig.4) consists of one input layer, one or more hidden layers and one output layer. A typical feature and at the same time the reason for their frequent connection and usage is the relatively simple implementability and versatility of using such a network.

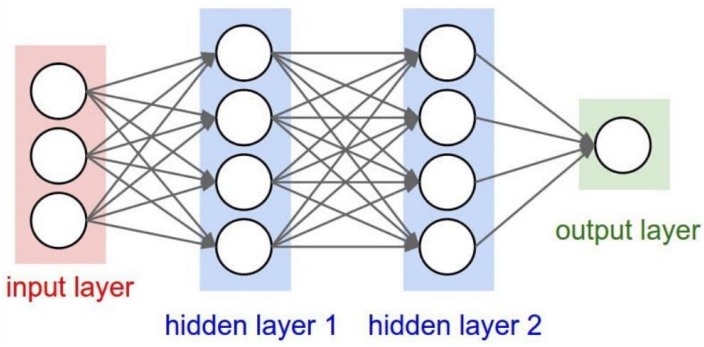

Fig.4. An example of MLP network with two hidden layers [7].

B. Artificial Neural Networks

Hopfield neural network - this type of network is created by a combination of recurrent neural network and supervised learning method. It uses a completely different network structure than MLP. Individual artificial neurons are both input and output and are connected to all other neurons. Such type of neurons is called dual. All neurons are located on the same layer and thus there is no input, hidden or output layer in such a topology [6, 7].

Convolutional neural networks (CNN) are the main type of artificial neural networks used in the field of computer vision. Specifically, it involves the detection and recognition of objects in images, videos or the classification of images into different groups. It involves recognizing human faces, animal species, handwritten text, dangerous driving situations and a large number of other uses. Individual operations are part of almost every CNN and they are always performed in a strictly defined order. The method of their use will be described on the following task example, where the assignment would be to determine which object is in the picture (Fig.5).

An image containing one or more objects would be fed to the input of the network. Subsequently, operations one to three would be run several times in a sufficient amount of repetitions so that the network managed to extract all the necessary information from the input image. All the findings are then combined into one so-called symptom map within fully connected layers. The map created in this way is processed by the last operation - classification, in which it reveals individual objects. The output of the network is a list of detected objects, often together with the percentage success rate of their detection [7, 8].

Fig.5. Simplified CNN operation procedure [8].

Since **convolution** is the first operation of the network, it works with the original input images. Its first task is to determine the dimensions of the image such as height, width and depth. In most cases, the input images are colored and therefore in RGB format their depth is 3. Width and height are the basic data about the image. For example, a color image would be loaded as an 8x8x3 array of matrices, and a black and white image as an 8x8x1 array of matrices. The first layer is the input layer, which usually has specified exact dimensions, but it is not a condition for the correct functioning of the network.

The next layers are convolutional layers, of which there are several dozen in the network. These layers consist of a set of filters used to extract all the necessary information and elements from the image. This process takes place by storing the values of all pixels in matrices. The matrices are then multiplied by the values of the individual filters, which are also placed in the matrices, and the output is a matrix in which the output values of this operation are found separately for each pixel. Within one layer, a larger number of different filters are applied to all pixels of the image. The output of the convolution is the matrix of output values for individual pixels and the black-and-white form of the input RGB image. The important information is that it is the individual values of the filters that are important in the learning process and based on these values KNS learns to detect objects [7].

Just like the previous operations, **classification** can be performed using different methods. One of these methods is the Softmax function. Let's imagine that the network should be able to detect a certain number of classes, which we denote as n, while they can be, for example, classes such as a person, a vehicle, a road sign or an animal. This function assigns to each of the classes the probability with which the object in the image belongs to one of the classes. The sum of the probabilities of all classes must always equal one. Subsequently, the probability for each class is calculated using formula 1, where N is the total number of classes and z is the output vector from the previous layer:

$$S(z)_j = \frac{e^{-z_j}}{\sum_{n=1}^{N} e^{z_n}}, \; j = 1,2,3, \dots N \tag{1}$$

This equation calculates for each class the probability that the object in the image belongs to it. This value ranges from 0 to 1. The last step is to select the class with the highest probability value. This value can often be seen next to the description of the detected object in the image [7].

3 Methods for Detection of Objects in the Image

Computer vision is a field of artificial intelligence that has been gaining more and more prominence in recent years and is beginning to develop. This development began with the creation of convolutional neural networks and the production of autonomous vehicles.

Object detection is an important part of computer vision. Currently, there are many approaches, algorithms and solutions to detect individual objects. The difference between object detection algorithms and classification algorithms, which are also used in other areas of artificial intelligence, is that object detection algorithms draw bounding boxes around detected objects, thereby visually signaling individual points of interest. Their number in the examined image is not limited by anything and there may be several dozen of them depending on the number of objects in the image.

The reason why standard convolutional neural networks using fully connected individual layers are not used for image object detection is that the length of the output layer is variable and therefore not constant. This is because the detection of individual objects does not decide which category the image as a whole belongs to, but which category the objects in the image belong to. For this reason, there can be a large number of given outputs. Therefore, for this case, standard convolutional networks were modified to meet all requirements and could also be used in this area. Specific types of these solutions will be described in the following part of the work.

R-CNN (Region-Convolutional Neural Network) is one of the methods used to detect objects from input images. This method uses the so-called selective search algorithm, which divides the input image into 2000 regions, which are then further processed. In this way, it proposes the aforementioned 2000 candidate areas where potential objects could be located [9].

Fast R-CNN method was created by the same author as the classic R-CNN network. Compared to this version, it brings a significant improvement in the duration of object detection (Fig.6).

The essence of this network is similar, but instead of sending individual 2000 regions to the convolutional network, the whole image is directly sent to this part, which is not divided in any way in this step [9].

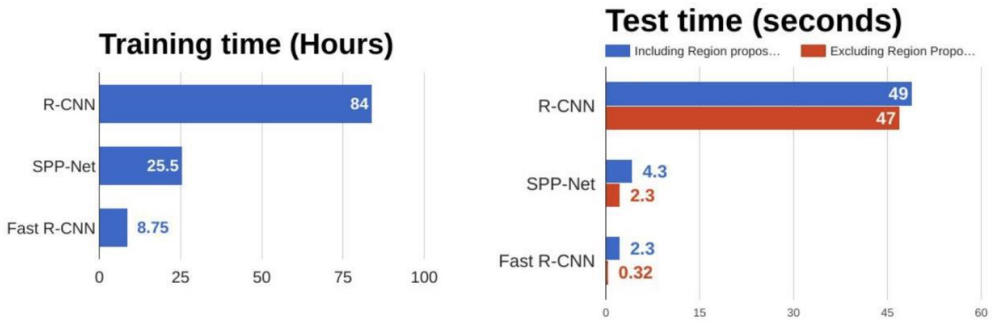

Fig.6. Comparison of Fast R-CNN and classic R-CNN network speeds [9].

YOLOv3 is a fully convolutional neural network (FCNN) that is used to detect objects in an image. This is already the third version of this algorithm, which has been constantly evolving over time. Currently, *YOLOv3* is one of the fastest detectors of objects in the image among all the systems used in this issue [10].

The most striking element that distinguishes *YOLOv3* from other object detection methods is the processing of the entire image in one step. For a better idea of the processes involved in image processing and object detection, the R-CNN and Fast R-CNN functions were described in more detail in the previous two subsections. The fact that the entire process takes place in a single step has the most significant impact on reducing the overall detection time [10].

Another advantage is that this method does not require an exact and uniform size of the input image. This facilitates the process of training the network, there is no need to take into account the size of the photos in the input dataset, as well as the images on which we want to perform object detection. As a result, there is one less operation in the image processing process compared

to other methods, as there is no need to additionally adjust the size of the candidate areas. This feature also makes the network faster [10].

The architecture consists of 2 parts. The first part of this architecture is *Darknet-53*. It is a block that acts as a flag extractor. This block consists of 53 convolutional neural layers. After the aforementioned *Darknet-53* block, there are several more interconnected convolutional layers within the network. The output from these layers is directed directly to the next part - the output layer. In this part, grids of different sizes are applied to individual images. These grids fall into three different categories, depending on what objects they target [10].

Individual types of outputs according to the used grid are divided into:

- Outputs for large objects – large objects are detected first within frames.
- Outputs for medium-sized objects – after processing several convolutional layers from the time of detection of large objects, the detection of medium-sized objects comes next.
- Outputs for small objects – here the same procedure as for the detection of medium-sized objects is applied, so the image goes through several convolutional layers again, and small objects on the given image are detected last.

During the entire process of detecting objects of different sizes, *Upsample* is used to combine information from previous detections and to obtain better results in the currently performed detection.

In Fig.7 a simplified procedure of the entire process of detecting objects with different sizes is shown (from large to small objects).

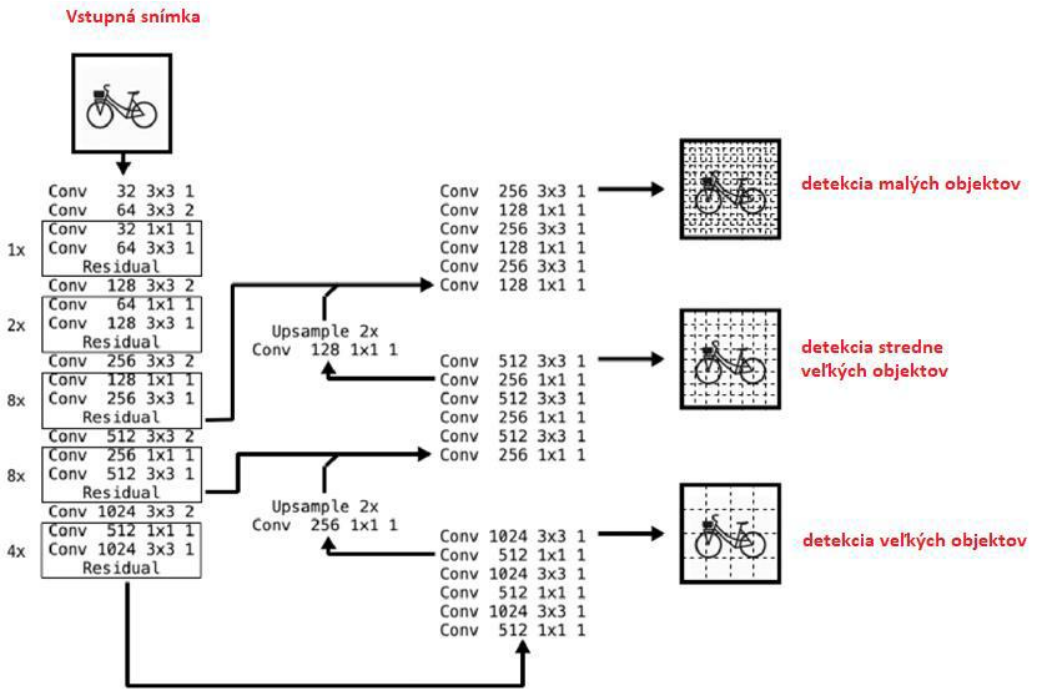

Fig.7. The process of detecting objects with different sizes [8].

4 Case Study

In this chapter, the complete process of creating the entire practical part will be described. The first step was the selection of a suitable algorithm for creating the CNN. This selection consisted of a survey of individual algorithms and approaches to the creation of neural networks. Another criterion was the systems that are currently used in the area of car safety. Algorithms used for image processing and subsequent object detection are constantly evolving and increasing their object detection accuracy along with shortening the time required for detection. From the currently used algorithms we chose the ***YOLOv3*** algorithm.

The second very important part of the work was the installation of necessary software for training and testing the neural network. The entire work is created in the ***Python 3.9*** programming language. ***Miniconda*** and ***pyCharm*** were chosen as the environment in which we developed the algorithm. In the field of computer vision and machine learning, we used the ***OpenCV*** tool. To create our own image annotations we used ***LabelImg***, which creates data for training of neural networks from our own images. We used the ***OIDv4 Toolkit*** to obtain thorn data. Next, we used the ***Darknet*** framework to use the performance of the graphics card to train the neural network and to configure YOLOv3 algorithm. The last tool we used was ***PyQt5m***, which enables the creation of user applications.

A. Photo dataset

In this phase of the work, it is necessary to know what kind of specific objects should be detected by CNN and to divide them into groups. In our case, CNN will be able to detect following 7 different types of objects:

- Vehicles,
- People,
- Animals,
- Traffic signs: prohibition, information, warning and command.

The total number of used photos is 4400 and they are divided into 1100 photos of cars, 1100 photos of people, 1100 photos of large animals and 1100 photos of individual types of traffic signs. The objects in the individual photos are photographed at different angles, under different lighting conditions and also from different sides.

At the same time, the individual objects in the photographs are captured in such a way that they do not resemble each other too much. There were used pictures of vehicles of different sizes and brands, and people of different ages, genders or races.

Since it would take a very long time to acquire our own data for training, and ultimately the data would not be completely suitable for training due to the expected low variety of captured objects, we used already created photos for training the network, which are directly intended for training of neural networks. We used the ***OIDv4*** tool to retrieve photos from two different sources:

- German Traffic Sign Detection Benchmark (GTSDB) – the official list of traffic signs for the European Union,
- storage.googleapis.com – a repository with a large number of photos for neural network training.

The second option was to create these photos manually along with marking the objects in the picture. For this operation, we will use the already mentioned ***LabelImg*** tool. The first step is to load the photos. The second step is setting the format of preparing photos. In our case, we used the YOLO format. Next, it was necessary to mark the objects and to name them (Fig.8). The output of the entire process is text files with data about objects.

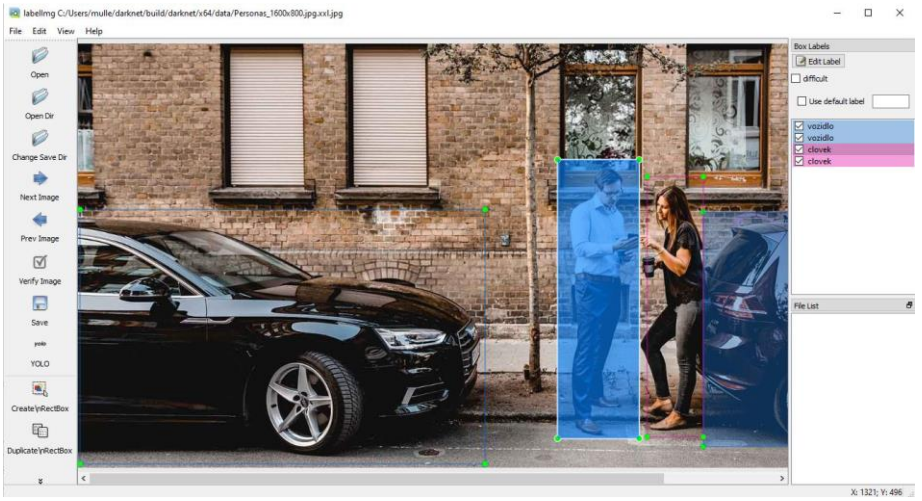

Fig.8. *LabelImg* user interface.

B. Neural Network Training

This is the most important part of the work, in which the main goal is to train own neural network based on actually assembled input images. After successful completion of this step, it will be possible to use the neural network to detect objects either in photos or on video. There are many ways to train a neural network, and the individual procedures differ in terms of difficulty and duration of the training itself. In general, two training procedures are known:

- using the processor and thus a CPU,
- using a graphics card and thus a GPU.

In our case, we started by training the neural network using the processor. Such training is suitable in the case of a small number of input images on which the given neural network is to be trained. But in the case of a larger dataset it is almost unrealistic to achieve the best possible results with this procedure (using a processor). The main disadvantage is too long training time, as the processing power of the processor is too low even in the case of high-performance processors. However, this is a simpler method, as there is no need for the difficult installation of additional software, which is required for other types of training.

The second chosen procedure was to train the neural network using a graphics card, in our case *NVIDIA GeForce GTX 1050 Mobile 2GB*. To train a neural network using a graphics card, installation of *CUDA* package is required. The advantage is that the computing power of the graphics card is not only used during the training of the neural network, but also during its subsequent usage and object detection. The usage of the graphics card is many times faster compared to the processor.

Despite the higher speed of training using the graphics card, the estimated time of the overall training process was estimated to be several days of pure time. It was due to the setting of a large number of iterations and also not too high performance of the graphics card. Despite investing a relatively large amount of time to install the required software and create all the necessary adjustments and settings, we decided not to continue with this method of neural network training.

The method we eventually chose and successfully trained our own neural network with was the graphics card trained version. But it was not a graphics card directly on our laptop, as in attempt number two, but a virtual graphics card offered by Google. This is a *Tesla K80* graphics card,

which is physically located on the side of the provider of this service. The entire training process took place using the *Google Collaboratory cloud service*. It is a service focused on creating your own projects in *Python* with the possibility of using various libraries such as *Keras, TensorFlow, PyTorch or OpenCV*. The great advantage of this service is that it provides very high processing power absolutely free! The only limitation is the usage time of this graphics card for a period of 8 hours. Therefore, it was necessary to save the progress and, after the specified time, start the whole process again, when we reconnected to the graphics card and had it available for another 8 hours. Thanks to the saving of interim results, we could always continue at the point where the training of the network ended. Even before starting work in *Google Collaboratory*, we saved the entire dataset of photos together with their annotations and remaining text documents on *Google Drive* (Fig.9).

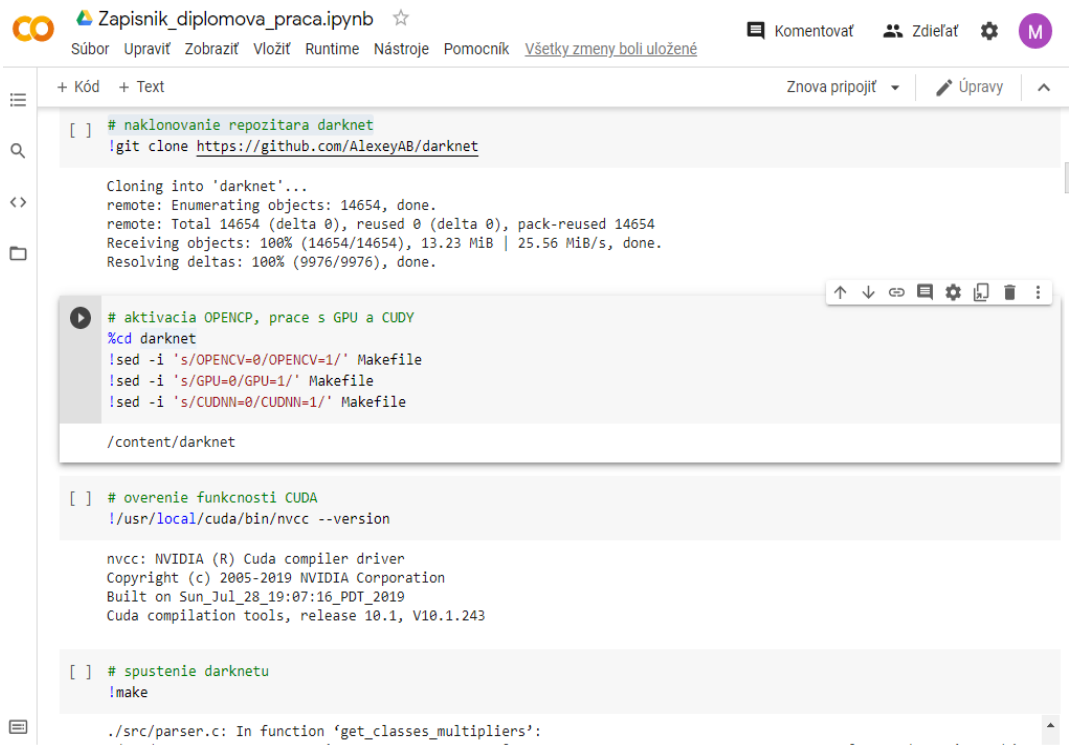

Fig.9. A preview of *Google Collaboratory* user experience.

The total training time of the neural network was 32 hours of pure time and the result is several files. These files were always created after performing 1000 iterations. From among all these files, we then had to choose the one that would provide the best results. As expected, it should have been one of the last files created. The file that reached the highest value of the so-called mean average precision was the most suitable. To find this value for individual weight files, we used a function provided by *darknet*. We used this function for individual files with weights and found that the weights in the last file reached the highest value. This file contains the trained final convolutional neural network, which we will subsequently work with in object detection.

C. User Application

We created the entire user application using *Qt Designer library* using the *PyQt5* tool, which enabled the entire application creation process to be performed in *Python* programming language. We have created two different applications, for detecting objects in images and for detecting objects from video captured in real time using a webcam. The creation and work with the application for object detection from video captured in real time will be described in detail.

The application environment consists of the following parts:

- Button to start the camera - after starting the application, the webcam is turned off by default for security reasons. The user can turn it on at any time and then turn it off as needed. The button called "Launch camera" is used for both of these actions. The button is located at the bottom of the screen and is of the Push Button type.
- Display area - this part of the application is used to display captured video from the webcam. If the video is running, this area occupies the majority of the entire application. It is also used to mark detected objects and thus the driver can spot objects that he could miss under certain conditions. It is located in the middle part of the screen and is of the Label type. In Fig.10, its area is marked with blue squares.
- Control of safety systems – this section is used to turn on and off individual warning safety systems. In the basic state, all warning safety systems are switched off, so the driver can switch them on as needed. These are the following systems:
    - Vehicles – if this security system is activated and CNN detects a vehicle in the captured images, an audio warning will be triggered and the system will play the sound "car car car..." and a visual warning with a large inscription "CAUTION" will also appear on the display.
    - Signs – if this security system is activated and CNN detects a traffic sign on the captured images, an audio warning will be triggered and the system will play a message based on the type of traffic sign. If it is a prohibited road sign, the warning message will be "prohibited prohibited prohibited...", if it is an informational road sign, the warning message will be "information informational informational...", in the case of a warning road sign, the warning message will be "warning warning warning ..." and the last kind is a command traffic sign, for which the message will be "command command command...". At the same time, a visual warning with a large inscription "WARNING" will also appear on the display.
    - People – if the system detects a person in the vicinity of the vehicle and the people warning safety system is activated, the audible warning sign "person person person..." will be triggered. Along with this sound signal, the warning text "CAUTION" is also displayed on the display.
    - Animal - the last safety system that can be turned on is a warning to the driver in case of detection of large animal in the vicinity of the front part of the vehicle. If there is a tall animal on the image captured by the camera and the CNN successfully detects it, the sound warning message "animal animal animal..." will start and the warning text "BRAKES" will appear on the display.

Starting and turning off individual security systems is carried out through check boxes and is located on the upper right side of the screen within the application.

- Area for warning text – a warning text is written in this place in case the CNN detects one of the target objects. Under normal circumstances, this place on the desktop is. The text displayed here is always displayed in red capital letters. Its goal is to attract the driver's attention so that he can react in time to a threatening event. The text is displayed here if one or more security systems are activated at the same time. The individual warning texts that can be displayed here are "CAUTION", "WARNING", "BRAKES" or, if several safety systems are activated at the same time, they are "CAUTION/WARNING". This field is located in the upper left part of the screen and is of type Label.
- Main title – this is the name of the application, at the same time it indicates what activity the application is intended for. It is located in the upper part of the area.

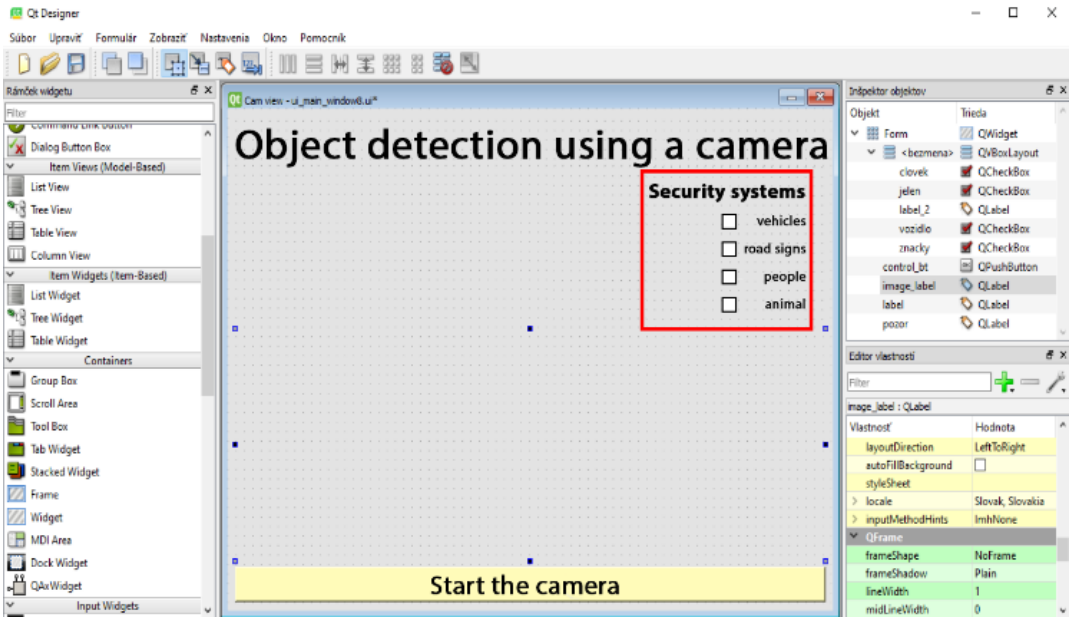

Fig.10. Creating an application in *Qt Designer program.*

The next step was the creation of a *Python* script that serves to connect the user environment with the already trained neural network. This script is called "*detekcia_webkamera.py*" and at the same time it is the main script from which the entire application is launched. For the sake of clarity, we have created a folder named "*functionalAPP_webcamera*", which contains all the necessary files for the proper functioning of the application:

- A folder called "detective" - in this folder there is a text document called "classes.names", which lists the names of all objects that CNN works with. Furthermore, it is the final output of the network training process "yolov3_custom_final.weights", which contains the weights of our CNN, which are used for object detection. The last file in this folder is the configuration file "*yolov3_ train.cfg*".

- Script called "*detekcia_webkamera.py*" - as already mentioned, it is a script used to connect the user environment with an already trained neural network and at the same time to start the entire application.
- Script named "*user_forecourt.py*" - the appearance and layout of all elements of the user environment, which was created in Qt Designer program, can be found here.
- Audio recordings – these are 7 audio recordings in .mp3 format. These recordings are used to soundly warn the driver in case of danger.

Fig.11. Person detection using a real-time object detection application.

The very process of creating this application was in many steps similar to creating an application for object detection from video. The user interface has a similar appearance and was also created using *Qt Designer*. The script itself, which is intended for object detection, is different, as the procedure for processing photos and video is different. In the case of this application, the user is able to choose any image on which he wants to perform the detection. He performs this action by pressing the "Select image" button, after which a window will open with the possibility to browse individual directories on the computer in order to select a specific photo. Even in this case, there is the possibility to turn on and off the individual safety systems (Fig.12), where the safety system is turned on to warn the driver in the presence of people.

Compared to an application designed for object detection from video, this application contains several pieces of information provided to the user:

- Total time duration in seconds of objects detection from the selected photo.
- The second value is the total number of detected objects. It is a figure that represents the number of objects before the unification. This information tells how many objects from the given classes are in the picture in total.

- The last value is the total number of detected objects after unification. This information tells how many bounding-boxes are drawn on the selected image.

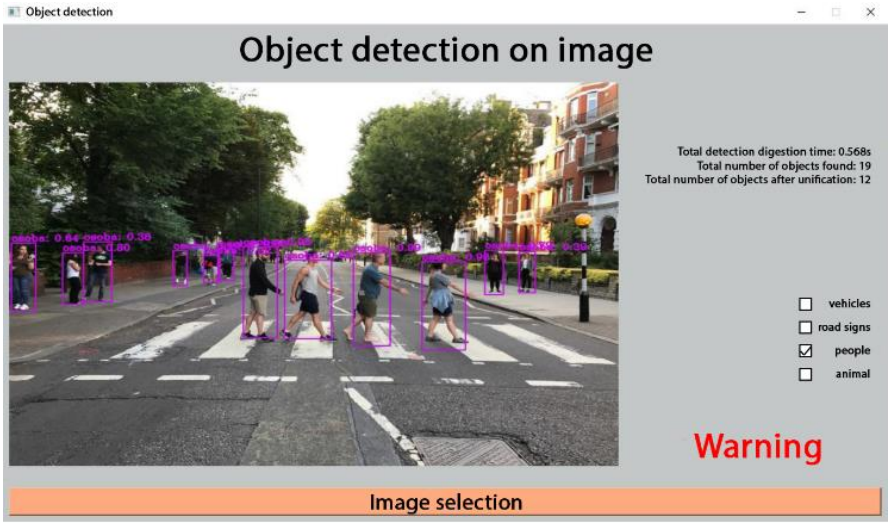

Fig.12. Creating an application in *Qt Designer* program.

Conclusion

The task of the paper was to create an assistance safety system for the driver of the vehicle, which will detect not only a certain group of objects representing a risk when driving the vehicle, but also vertical traffic markings. The condition was the usage of a convolutional neural network, which will be used to detect these objects. The task was to create, train and test our own convolutional neural network, which will be able to detect selected objects from images captured by the camera. The second main task was to create a user application that will be used to work with an already created convolutional neural network and at the same time control individual security elements.

We have successfully trained our own convolutional neural network that can detect and recognize 7 different objects. We verified the functionality of the network on different types of input data, such as photos and videos directly from real traffic or from video captured in real time using a camera. The success of the detection of individual objects depends on the quality of the resolution of the input images, the lighting and the environment in which the objects are located.

We have also successfully created two user applications. The first application is used to detect objects in selected photos. At the same time, it provides the possibility of activating the security system for warning in case of detection one of the threatening objects or vertical traffic markings. The application provides basic information about the detection process, such as the total detection time or the total number of detected objects in the selected photo.

The second application is used to detect objects from video that is captured in real time using a camera. The user has the option to turn the camera on or off at any time. It can also activate any number of security systems that work together. If one of the threatening objects is detected, an audio warning will be triggered along with a warning text statement directly in the application. During the whole time, the user has the opportunity to watch the image captured by the camera directly in the application, at the same time, individual objects are marked in the captured image if they are located in the vicinity of the vehicle.

Acknowledgement

This paper was supported by the Slovak Grant Agency VEGA 1/0107/22 and KEGA 039STU-4/2021, and by the Scientific Grant APVV-17-0190.

References

[1] European commission (2022). *2021 road safety statistics: what is behind the figures?* Retrieved online, January 10, 2024, https://transport.ec.europa.eu/background/2021-road-safety-statistics-what-behind-figures_en

[2] Martin, D. (2022). *Appen Delivers High-Quality Training Data for Autonomous Vehicle Manufacturers.* Retrieved online, January 10, 2024, https://www.unite.ai/appen-delivers-high-quality-training-data-for-autonomous-vehicle-manufacturers/

[3] Holts (2018). *A Guide to Your Car's Safety features*. Retrieved online, January 10, 2024, https://www.holtsauto.com/holts/news/guide-cars-safety-features/

[4] Procházka, J. (2014). *BMW Night Vision 3rd generation sees for you (In Slovak).* Retrieved online, January 10, 2024, https://techbox.dennikn.sk/bmw-night-vision-3-generacie-vidi-za-vas/

[5] Hardesty, L. (2017). *Explained: Neural networks.* Retrieved online, January 10, 2024, https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414

[6] Snives, J. (2018). *SimpleNeuralNetwork*. Retrieved online, January 10, 2024, https://github.com/snives/SimpleNeuralNetwork

[7] Cybiant (2020). *An introduction to Artificial Neural Networks*. Retrieved online, January 10, 2024, https://www.cybiant.com/knowledge/artificial-neural-networks/?v=13dd621f2711

[8] Muráň, J. (2019). *Introduction to convolutional neural networks (In Slovak)*. Retrieved online, January 10, 2024, https://umelainteligencia.sk/uvod-do-konvolucnych-neuronovych-sieti/

[9] Gandhi, R. (2018). *R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms*. Retrieved online, January 10, 2024, https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[10] Orješek, R. (2020). *YOLO architecture for object detection (In Slovak)*. Retrieved online, January 10, 2024, https://umelainteligencia.sk/architektura-yolo-pre-detekciu-objektov/

Authors

Ing. Ján Cigánek, PhD.
Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology in Bratislava, Slovakia
jan.ciganek@stuba.sk
He was born in 1981 in Malacky, Slovakia. He received the diploma and PhD. degree in Automatic Control from the Faculty of Electrical Engineering and Information Technology, Slovak University of Technology (FEI STU) in Bratislava, in 2005 and 2010, respectively. He is now Assistant Professor at Institute of Automotive Mechatronics FEI STU in Bratislava. His research interests include optimization, robust control design, computational tools, SCADA systems, big data, and hybrid systems.

Ing. Filip Žemla
Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology in Bratislava, Slovakia
filip.zemla@icloud.com
Currently a student of doctoral studies at Slovak University of Technology in Bratislava. The main focus of his studies is oriented to virtualization and optimization modern manufacture processes. His main skills are SCADA systems, database systems and front-end programming.