International Journal
Information Technology Applications (ITA)

Volume 8, Number 2, December 2019

DETECTING AWARENESS STATE OF A DRIVER DURING DRIVING

Zuzana Képešiová, Ján Cigánek, Štefan Kozák

Abstract:

The presented paper deals with automatic detection of driver drowsiness. Detecting the driver's drowsiness behind the steering wheel and then alerting him may reduce road accidents. Drowsiness in this case is captured using a car camera, whereby, based on the captured image, the neural network recognizes whether the driver is awake or tired. The convolutional neural network (CNN) technology has been used as a component of a neural network, where each frame is evaluated separately and the average of the last 20 frames is evaluated, which corresponds to approximately one second in the training and test dataset. First, we analyze methods of image segmentation, and develop a model based on convolutional neural networks. Using an annotated dataset of more than 2000 image slices we train and test the segmentation network to extract the driver emotional status from the images.

Keywords:

Image segmentation, artificial neural networks, convolutional neural networks, Python, deep learning.

ACM Computing Classification System:

Computer vision, machine learning, artificial intelligence, neural networks.

Introduction

Over the past ten years there has been an unprecedented development of digital information, communication and intelligent technologies that affects the quality and lifestyle at our planet in a revolutionary way. World-wide innovation trends in complex embedded computerization and digitization processes in smart cars are now leading to the need for research, development and implementation of new system solutions and changes, introduction of new intelligent methods for diagnosing critical processes in modern cars and in many other areas.

The degree of up-to-date use of soft computing methods in modern car architectures is very high, as it is a dominant and indispensable part of modern smart car production. Smart car architecture and intelligent platforms represents the realization and naming of large-scale changes in the automotive industry. These changes include, in particular, digitization, automation and ICT integration at all levels process of the car control and diagnosis.

The proposed paper is based on the development of effective soft computing methods using convolution neural networks and deep-learning techniques to solve the problem of monitoring, detecting and modelling critical situations and emotional stress of a smart car driver. Fast detection and monitoring of the selected car parameters the attentive and emotional status of the driver are critical for safety and comfort of driving.

Face recognition as a typical biometric identification technology is recognized as an essential technology to establish secure control in many areas. Face recognition has attracted much attention from researchers and engineers over the past decades owing to its wide range of applications in many fields including information security, identity authentication, law enforcement, smart cards, access control systems and so forth. The entire face recognition procedure consists primarily of two operations: feature extraction and classifier design. These two steps have a substantial influence on effectiveness and reliability of various recognition approaches.

The paper is organized as follows. Section 2 introduces a deep lerning methods for classification. Section 3 shows case study results.

1 Deep-Learning Methods

The methodology of the presented paper corresponds to the latest global trends in research and development of advanced soft computing methods based on machine and deep-learning methods for classification, pattern and image recognition with possible use in the various fields (robotics, smart cars, health processes, biotechnology, etc.).

Deep learning is making major advances in solving problems that have resisted the best attempts of the artificial intelligence community for many years. Deep-learning methods are representation-learning methods with multiple levels of representation obtained by composing simple but non-linear modules that each transforms the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. With the composition of enough such transformations, very complex functions for large-scale applications can be learned.

A. Artificial Neural Networks (ANNs)

ANNs are widely used not only for segmentation, but also for preprocessing, compression, feature extraction, enhancement, reconstruction or restoration of images.

As with biological neural networks, the basic ANN calculation unit is the neuron (Fig.1). Each neuron has one or more inputs (called dendrites) and just one output (an axon). Axon is linked by synaptic connections to dendrites of other neurons. The power of these synaptic connections is referred as weight w and is changed during training. In the basic model, dendrite carries a signal to the body of the neuron, where signals from all dendrites add to each other with the addition of bias b. This sum is the input of the activation function f of the neuron; a common activation function is sigmoid function:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{1}$$

or ReLU, out = *max(0, in)*. The result of the activation function goes to the output of the neuron, the axon.

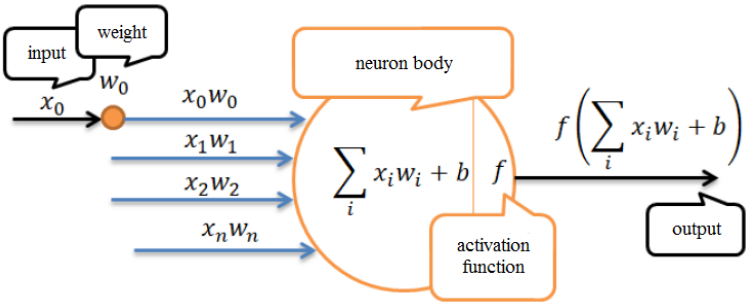

Fig.1. Artificial neuron.

Generally, the ANN consists of neurons grouped in three basic layers: input, hidden and output layer, with a different number of neurons. Principally, it is not allowed to connect neurons from the same layer, only from different ones (Fig.2) [1].

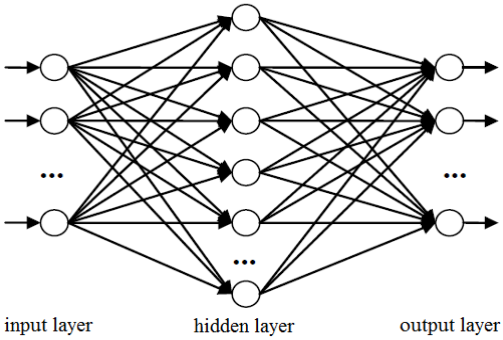

Fig.2. Artificial neural network scheme with 3 basic layers.

One type of ANNs is also forward neural networks. They comprise a plurality of layered neurons such that each neuron of the preceding layer is connected with each neuron of the next layer. The first layer of neurons is called the input layer (there are no calculations in the input layer), the last one is output layer and the layers between them are called hidden layers. The Single Layer Perceptron is a forward neural network without a hidden layer, and the Multi-Layer Perceptron (MLP) contains at least one hidden layer.

For MLP training we need data with assigned values of required outputs for a given sample of inputs. MLP is trained using a back-propagation algorithm in iterations. In each iteration, weights are adjusted using new training data. In this way they pass through all the layers up to the output layer. Subsequently, the MLP result is compared with the desired result from the training data.

A large neural network may take a lot of time to train and therefore a technique called dropout may be used. This method is used to reduce the size of neural network and to speed up the learning process of the algorithm. The idea of dropout is based on dropping out units, in this case neurons, in a neural network. These units are chosen randomly with a probability of q=1-p. If the unit is chosen to be dropped out, then all incoming and outcoming connections are discarded. Randomly discarded neurons guarantee each neuron to learn something useful on its own instead of being dependent on other neurons [3].

With consideration of a neural network of a size of L layers, where index of neural network layer l is $l \in \{0, \cdots, L-1\}$, $l = 0$ is an input layer and $l = L - 1$ is output layer. Input and output vectors in hidden layers are computed according to relation:

$$x^{(l+1)} = W^{(l+1)} y^{(l)} + b^{(l+1)} \tag{2}$$

$$y^{(l+1)} = a(x^{(l+1)}) \tag{3}$$

where $x^{(l)}$ refers to input vector of layer l, $W^{(l)}$ is weight parameter, $y^{(l)}$ refers to output layer vector to layer l, $b^{(l)}$ is bias parameter at layer l and $a(x^{(l)})$ is an activation function. Previous equations (2) and (3) change to:

$$\delta_i^{(l)} \sim Bernoulli(p) \tag{4}$$

$$\bar{y}^{(l)} = \delta^{(l)} \otimes y^{(l)} \tag{5}$$

$$x^{(l+1)} = W^{(l+1)} \bar{y}^{(l)} + b^{(l+1)} \tag{6}$$

$$y^{(l+1)} = a(x^{(l+1)}) \tag{7}$$

by applying *dropout*, where $\otimes$ is element by element multiplication, $\delta_i^{(l)}$ is a Bernoulli random value of neuron i at layer l [3].

B. Convolutional Neural Networks (CNNs)

CNNs are a type of ANNs, but they are modified to be more efficient in processing images as inputs. They use four operations: convolution, pooling, ReLU and MLP as their basic building unit. Each of these operations has a 3D image for both input and output.

Each image can be represented as a matrix of pixel values. 3D image consists of width, height and depth. The depth in this case is the number of channels. For example, a standard color image consists of three channels - red, green and blue. MRI or CT medical images are a black and white images that have only one channel.

The role of **convolution** in CNNs is to select the main properties from the input image. The convolution preserves the relationships of individual pixels in space to learn these properties from smaller squares of the image. It contains several learning filters to process the image. These filters (called kernels) have a small size (e.g. 5 pixels in width and height, and several pixels in depth). When processing an image, the filters move across its entire width and height. For example, if the image is 5x5 pixels in grayscale and the filter is a matrix of 3x3x1 elements, then the convolution results in a 3x3 matrix, as the 3x3x1 matrix can be shifted 3 times in width and 3 times in height. The convolution output matrix (called an activation map, a property map or a tensor) is the result of matrices multiplication (image and filter) by elements on all channels. (Fig.3) shows an example of convolution with a 3x3 size filter, with one input and one output channel.

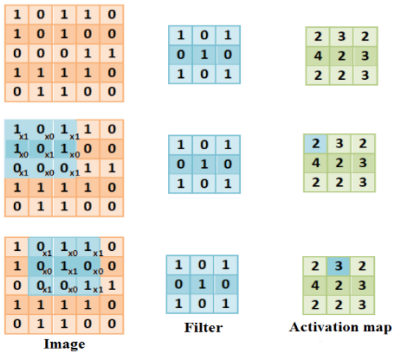

Fig.3. Convolution with a 3x3 size filter, with 1 input and 1 output channel.

The size of the convolution property map is affected by three parameters: depth, filter step, and zeros. Depending on the number of filters used in the convolution, the depth (number of channels) of the output increases. The filter step tells how many pixels the filter is to be shifted by the next matrices multiplication. Sometimes the image is supplemented with zeros to the edge so that the resulting property map has the same width and height as the original image.

The transposed convolution, or deconvolution, has the same operations as the convolution, but in the opposite direction. This means that instead of diminishing the image dimension, it expands the image. It is used to adjust the image to its original size, after convolution, so that CNN input has the same dimensions as the output (i.e., the indicated output pixels correspond to the same input pixels).

The 3x3 filter convolution, with one input and one output channel, can also be represented in a different way than in (Fig.3). First, the filter (3x3 from the previous example) becomes a convolution matrix (9x25), where each line represents a convolution operation with other window position. The input image values are stored in a column vector. Subsequently, the value of the activation map is calculated as a multiplication of the filter and image matrices. In this case, the output is a column vector whose values represent the activation map. The whole process is shown in (Fig.4).

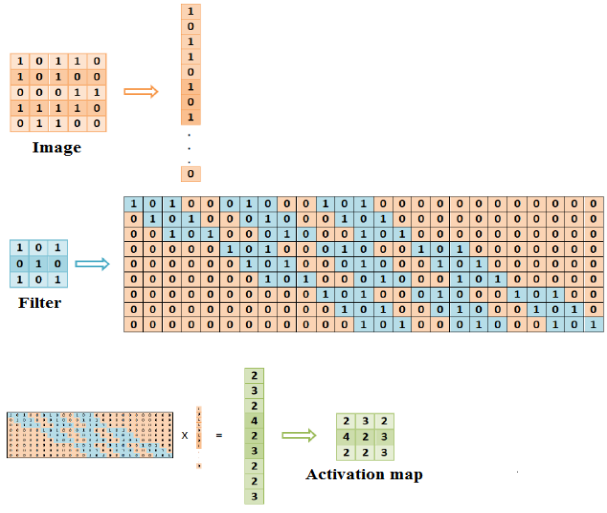

Fig.4. Convolution shown by matrix multiplication.

The transposed convolution uses the transposed convolution matrix (25x9), otherwise the procedure is the same. The transposed convolution matrix is multiplied with the input image converted into a column vector. The result is an activation map (5x5). The transposed convolution process is shown in (Fig.5). The output of the transposed convolution, using the same filter as in the previous example, is different from the initial image used at the beginning in (Fig.4).

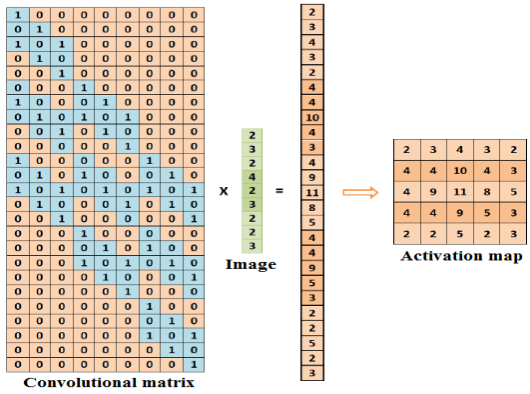

Fig.5. Transposed convolution with a 3x3 size filter.

ReLU is a nonlinear operation. The output value is calculated according to function out = max (0, in), where in is the input and max is a function that returns the maximum value of two or more elements. ReLU is an operation by elements (applied in pixels). If ReLU input is a negative number, then the output is zero, otherwise the input remains unchanged. Its role is to bring nonlinearity to the convolutional network. In contrast to sigmoid functions, ReLU has the advantage of suppressing the problem of vanishing gradients [2].

Pooling reduces the dimension of activation maps while keeping a clear signal throughput. There are several types of pooling: maximum, average, sum and others. In the case of maximum pooling (Max Pooling), the window size is first defined, the window is scrolled through the image as a filter in convolution, with the difference that the next window does not overlap with any previous one and the output from the window is the largest element. In the case of average pooling, the output is the average of all window elements. (Fig.6) shows an example of Max Pooling with a 2x2 window size.

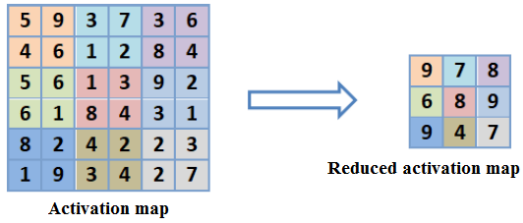

Fig.6. Max Pooling with a 2x2 window size.

After using several convolutions and poolings, the image is processed by MLP. The role of MLP is to use image properties obtained from previous operations to divide CNN inputs into multiple classes (segments).

As well as ANNs, CNNs also consist of layers (convolutional, pooling, ReLU and MLP). Most often, when creating a CNN, several convolutional and pooling layers are stacked. This pattern is repeated until the input image is divided into small pieces. There may also be parallel branches between the layers. Finally, the MLP layer follows.

It is good to know that the only difference between MLP and convolution layer is that neurons in convolution layer are connected only with the local part of the input image. But the neurons in both networks calculate the output in the same way, so it is possible to convert MLP to a convolution layer and vice versa.

2 Case Study

The aim of this study is to successfully detect driver drowsiness based on fascial expression recorder by car camera.

A. Hardware Characteristics and Software Environments

As it comes to hardware, a laptop consisting of Intel® Core™ i5-7200U 2.50GHz – 2.71GHz Processor and 8GB GeForce 940MX RAM was used for making computations.

As a software environment PyCharm development environment was used. PyCharm is one of software applications from JetBrains company used for developing python scripts and applications. One of these applications is also neural networks and machine learning area. As a help for a developer, many frameworks were created for making scripting faster and easier. Tensorflow and Keras, were used in this project.

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications. [4]

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation [5].

B. Dataset and Data Processing

Collected data consists of 20 different subjects. Each subject has 8 different behavioral patterns recorded on approximately 5 seconds long image sequence. These behavioral patterns are divided to two different states:

- **Aware (0)** – High concentration (1), looking around (2), happiness (3), usual face expression (4)
- **Sleepy (1)** – Slow eyelid movement (5), yawning (6), falling head (7), lethargy (8)

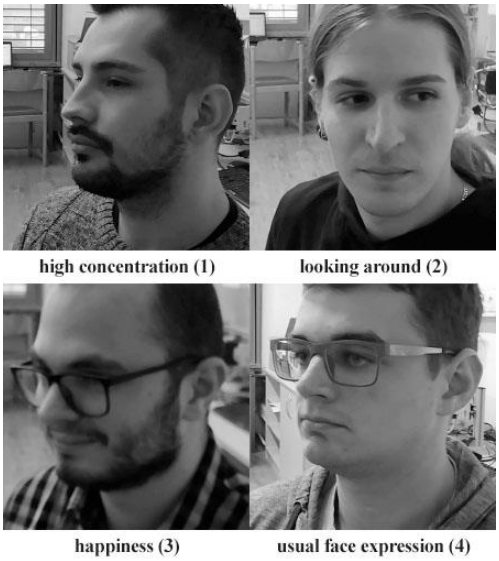

Fig.7. Example of dataset frames expressing states of awareness – high concentration, looking around, happiness, usual face expression.

Fig.8. Example of dataset frames expressing states of sleepiness – slow eyelid movement, yawning, falling head, lethargy.

Each video consists of one of these behavioral patterns in rate 28.16 frames per second. Each frame has dimensions of 1920 pixels wide and 1080 pixels high. Videos are recorder as color videos; therefore, they are referred as RGB videos. Each video was cut frame by frame and every fifth frame was taken to an image dataset, that created dataset of size 5258 frames divided into two categories: sleepy, that consists of 2619 frames and aware, that consists of 2639 frames.

Fig.9. Unpreprocessed video frame.

As we can see on (Fig.9), a camera is capturing a lot of junk data, that are telling us nothing about driver's state. Therefore, image preprocessing in inevitable. At first, color of frame is not needed, and a frame was converted from RGB color spectrum to greyscale. After obtaining greyscale version of a frame, the face area is cut out and picture's dimensions are changed to 224px per 224px. After obtaining small image cut from the frame, the image is also flipped horizontally.

Fig.10. Preprocessed video frame.

After collecting and preprocessing data, a model design follows.

C. Developed Model

Convolutional neural networks take a lot of computational power and a great amount of time to train, especially deeper these networks are. The idea of convolutional neural networks is not new, and many models were already created. There are models such as DenseNet [6], ResNet [7], Iception [8], VGG [9] any many others. These models also offer pretrained weights available online as part of Keras framework or as h5py file containing the model [10]. In first stages DenseNet and ResNet were tried out, but without much of success and therefore were not selected. For these problematics a model VGG Face was chosen, since it was trained on human faces, a dataset like dataset of this project. VGG Face is a CNN divided into 11 blocks, were 8 blocks are convolutional blocks, where

each block containers one or more convolution layers followed by ReLu activation layer and/or max pooling layer. A structure of VGG Face is available on (Fig.11).

Fig.11. VGG Face structure [9].

VGG Face was trained over 2.6 million of visual inputs and can categorize them into 2622 different categories. A required input is an 224x244 big image in RGB channel. Our data are in greyscale channel, since the color doesn't matter in this case, therefore data were converted to RGB channel but visually remain as greyscale, what makes them more universal and not color dependent.

The final model uses pretrained VGG Face as a base, but to fit our needs, the last layer of VGG Face is removed and on top of that a small head model is added. This head model consists of fully connected layers, followed by batch normalization, tanh activation function and dropout.

D. Model Training, Testing and Results

Developed model was trained with custom dataset divided into batches sized of 32 during 50 epochs long process and with nadam optimizer and categorical crossentrophy loss function. VGG Face layers were not retrained, but frost and a head model was tuned only. The whole training process on mentioned hardware took about 15 hours of training, while the results of the model were 90.77% training data accuracy, 98.02% validation data accuracy.

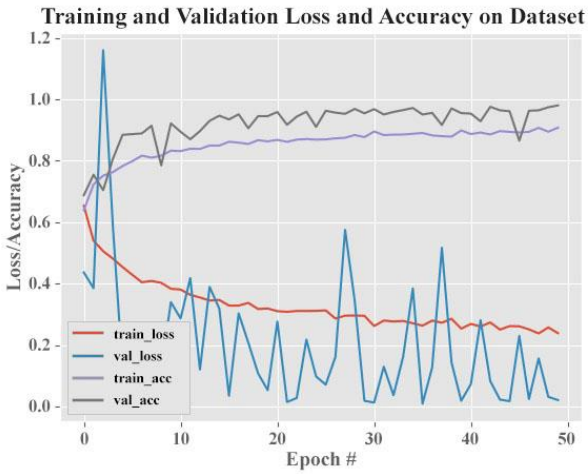

Fig.12. Training and validation loss and accuracy on dataset.

Despite good results, when the model is tested on a completely new person, that was not included in dataset, a neural network struggles to identify drowsiness in a person's fascial expression and suggesting a person is aware most of the time. Efficient gesture that is recognized in other people is slight head falling. On the other hand, all the videos of captured subjects, even ones recorded later, were recognized successfully.

Conclusion

We decided to create an application that could help drivers with detecting their mental state of drowsiness and we achieved to create a model, that works almost perfectly for separate frames and for sequence of 14 frames, what displays a 0.5s sequence, without fail. For every new person a network must be retrained to obtain satisfactory results for the subject. As the dataset consists of only 20 different subjects, there is a high probability of improvement of the algorithm by achieving dataset wider in its variety.

Outputs from the created model can then be used for sound, respectively. Light signaling to ensure reliable driving of the vehicle. Ensuring reliable vehicle operation is critical in the case of health problems such as cerebral defeat or heart attack. The proposed model approach of sensing and recognizing changes in the face of the car driver will reliably predict the occurrence of a dangerous situation.

Acknowledgement

This paper was supported by the Slovak Grant Agencies VEGA 1/0819/17, KEGA 030STU-4/2017, and by the Scientific Grants APVV-17-0190, SEMOD-79-2/2019 and Semod-80-7/2019.

References

[1] J. Ciganek and J. Osusky, "Structure Optimization of Artificial Neural Networks Using Pruning Methods", IEEE Int. conf. Cybernetics & Informatics 2018, Lazy pod Makytou, Slovak Republic, 2018.

[2] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", Advances In Neural Information Processing Systems, pp. 1-9, 2012.

[3] Z. Kepesiova, D. Rosinova and S. Kozak, "Comparison of Optimization Techniques for Process Recognition Using Deep Neural Network", IEEE Int. conf. Advanced Control Circuits and Systems (ACCS'019) and New Paradigms in Electronics & information Technology (peit'019), Hurghada, EGYPT, 2019

[4] TensorFlow, [Online]. Available: https://www.tensorflow.org/. [Accessed Nov.9, 2019]

[5] "Home - Keras Documentation" keras.io, [Online]. Available: https://keras.io/. [Accessed Nov.9, 2019]

[6] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pp. 4700–4708, 2017.

[7] He, K.; Zhang, X.; Ren, S.; and Sun, J. "Deep residual learning for image recognition", In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 770–778, 2016

[8] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. "Rethinking the inception architecture for computer vision,. In IEEE CVPR, pages 2818–2826, 2016.
[9] O. M. Parkhi, A. Vedaldi and A. Zisserman, "Deep Face Recognition", British Machine Vision Conference, 2015
[10] S. I. Serengil, "Deep Face Recognition with Keras" sefiks.com, [Online]. Available: https://sefiks.com/2018/08/06/deep-face-recognition-with-keras/. [Accessed Nov.9, 2019]

Authors

Ing. Zuzana Képešiová
Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology in Bratislava, Slovakia
zuzana.kepesiova@stuba.sk
Currently a student of doctoral studies at Slovak University of Technology in Bratislava. The main focus of her studies is oriented to development of a remote laboratory to monitor and control of mechatronic systems using digital twin and mixed reality.

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.

prof. Ing. Štefan Kozák, Phd.
Faculty of Informatics, Pan-European University, Bratislava, Slovakia
stefan.kozak@paneurouni.eu
Currently at the Institute of Applied Informatics at the Faculty of Informatics, Pan-European University in Bratislava. His research interests include system theory, linear and nonlinear control methods, numerical methods and software for modeling, control, signal processing, IoT, IIoT and embedded intelligent systems for digital factory in automotive industry.