COMPARISON OF REGULARIZATION AND OPTIMIZATION METHODS FOR PROCESS RECOGNITION WITH USE OF DEEP NEURAL NETWORK

Zuzana Képešiová, Štefan Kozák

Abstract:

This paper brings the topic of comparison of optimization techniques using gradient descent, gradient descent with momentum, Adam and learning rate decay in combination with previous optimization algorithms for face recognition using deep neural network. This paper compares several settings of these techniques as well as techniques among themselves with combination of regularization techniques varying in using no regularization, L2 regularization and dropout to successfully recognize the sex of a person captured. The result of the evaluation is taken in a matter of successfully recognized face rate for training data and test data, cost and deep neural network learning time.

Keywords:

Regularization, optimization, gradient descent, learning rate decay, machine learning.

ACM Computing Classification System:

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

Introduction

Currently humanity relies on computers in more extent than ever before and this trend is still on its arise. One of abilities of computers is also image processing called computer vision. Computer vision could be used in wide range of application as object detection, situation development prediction, object identification, object recognition and many more.

In this paper we will analyze the topic of success of different optimization algorithms with different learning rates and regularization techniques used in the context of person´s sex recognition based on the given facial photo as a subject of face recognition. As a dataset the UTKFace library is used. Original dataset consists of over 20 000 face images with annotation of age, sex and ethnicity. The images cover large variation in pose, facial expression, illumination, occlusion, resolution, etc. [1].

1 Regularization Techniques

Regularization presents one of methods used for reducing the overfitting of data. One of the options is to increase training data, but another are regularization methods such as *L2 regularization* or *dropout.*

A. L2 Regularization

L2 regularization technique has many synonyms and it is known also as weight decay or Tikhonov regularization and as ridge regression in statistics. *L2 regularization* is most used method of regularization. The principle of *L2 regularization* is to add extra term to the cost function and regularize the cost function. A modified cost function is expressed by formula:

$$J(\theta) = J(\theta) + \frac{\lambda}{2n}\sum_W W^2 \tag{1}$$

Where $J(\theta)$ is a cost function, λ is a regularization parameter, n is size of a training set and W is a weight in network [4].

B. Dropout

Not only overfitting, but also the neural network size could be solved by regularization. In case of large neural network, we could focus on a technique called dropout 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 abandoned. Randomly dropping neurons ensure each neuron to learn something useful on its own instead of relying on other neurons [5].

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 [5].

2 Optimization Techniques

In the experiment of recognizing sex of a subject by provided face photography, optimization methods such as *gradient descent*, *gradient descent with momentum* and *Adam* were used. As the algorithms provide different approaches to the problem, they also generate different results for settings provided.

A. Gradient Descent

Gradient descent is considered as one of the most popular method to optimize neural network. The main technique is performance of minimizing cost function *J(θ)* with the parameter θ updating parameters in reverse direction of the gradient of the cost function $\nabla_\theta J(\theta)$ with respect to the parameters [2].

Gradient descent varies in volume of data processed to compute the gradient of the objective function. *Gradient descent* could be a *batch gradient descent*, *stochastic gradient descent* and *mini-batch gradient descent*. The difference is, that *batch gradient descent* computes the gradient of the cost function for the whole dataset, while *mini-batch gradient descent* computes gradient respectively to data divided into small datasets and *stochastic gradient descent* computes gradient for each training example. In this paper we will consider a *gradient descent* as *batch gradient descent*. For computing the gradient of the cost function with respect to the parameters we will use the following:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta) \tag{8}$$

where θ is standing for parameters, that are updated by gradient $\nabla_\theta J(\theta)$ multiplied by parameter η, a learning rate [2], [6].

B. Gradient Descent with Momentum

As we could see in the previous part, there are some oscillations across epochs. These oscillations slow down the convergence to minimum. *Gradient descent with momentum* is an optimization algorithm which relies on exponentially weighted averages. This method smoothens the oscillations by adding a fraction γ of the update vector of the past time step to the current update vector v and update parameters θ by:

$$v_t = \gamma v_{t-1} + \eta \cdot \nabla_\theta J(\theta) \tag{9}$$

$$\theta = \theta - v_t \tag{10}$$

The momentum term γ is usually set to 0.9 or a similar value [3].

C. Adam

Adaptive Moment Estimation or also so-called *Adam* is an optimization technique computing adaptive learning rates for each parameter. *Adam*, just as *Adadelta* or *RMSprop*, stores exponentially decaying average of past squared gradient s_t, but on the other hand, also keeps an exponentially decaying average of past gradients v_t, like momentum:

$$v_t = \beta_1 v_{t-1} + (1-\beta_1)\nabla_\theta J(\theta) \tag{11}$$

$$s_t = \beta_2 s_{t-1} + (1-\beta_2)\nabla_\theta J(\theta)^2 \tag{12}$$

v_t and s_t are estimates of the mean and uncentered variance of the gradients respectively. Both variables are initialized as vectors of zeros. Problem is, they tend to bias towards zero and therefore they must be bias corrected first. The bias-correcting looks like:

$$v_t^{corrected} = \frac{v_t}{1-\beta_1^t} \tag{13}$$

$$s_t^{corrected} = \frac{s_t}{1-\beta_2^t} \tag{14}$$

With corrected mean and variance, the Adam update parameters rule is as follows:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{s_t^{corrected}+\varepsilon}} v_t^{corrected} \tag{15}$$

As the authors [12] of the algorithm propose default values of $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$ [2][3][7], this experiment adopt these values as well.

D. Learning Rate Decay

Learning rate decay is a form of learning rate scheduling, which if done well, can result in significantly faster convergence and convergence to a better minimum. Initial large steps enable a rapid increase in the objective function, but later smaller steps are needed to descend into finer features of the loss landscape.

Some of the learning rate scheduling methods include exponential scheduling, where learning rate decays as:

$$\eta_t = \eta_0 * 10^{\frac{-t}{r}} \tag{16}$$

Another learning rate decay as a power scheduling is used by Bottou [8] or Xu [9] as follows:

$$\eta_t = \eta_0 (1 + \frac{t}{r})^{-c} \tag{17}$$

where *c* is a "problem independent constant" [10].

A learning rate decay used for this experiment is inspired by one used by Andrew Ng [11]:

$$\eta_t = \frac{1}{1+\eta_0 i} \tag{18}$$

3 Case Study

Deep neural network models applied on this experiment were built, trained and tested on a laptop consisting of Intel® Core™ i5-7200U 2.50GHz – 2.71GHz Processor and 8GB GeForce 940MX RAM.

For this experiment the original dataset was cut in the numbers and age scale. Used dataset consists of 454 face images of people ranged in age of 18 to 60 included. The composition of dataset was a composition of 185 male and 269 female face images, which were selected randomly.

Fig.1. Example of training data.

As a training set and test set face images were preprocessed to size of 64x64 pixels and the rgb color palette was shrank into greyscale and standardized to divide the number representing a pixel by 255 for faster data processing.

There we 81 different deep neural network models created to test the accuracy of reading and recognizing the sex of a person by face. Models were a combination of *no regularization technique*, *L2 regularization* and *dropout* method with optimization algorithms such as *gradient descent*, *gradient descent with momentum* and *Adam.* Afterwards, *learning rate decay* optimization method with different values was used in a combination with all mentioned regularization and optimization methods to support wider range of models and their settings. Initial *learning rate* varies in value in 0.05, 0.005 and 0.0005. A general model setting is a 3NN layer model containing 4096 input nodes in input layer representing each pixel of a photo, 20 nodes in first hidden layer, 5 nodes in second hidden layer and 1 node in output layer resolving in 0 for a male and 1 for female.

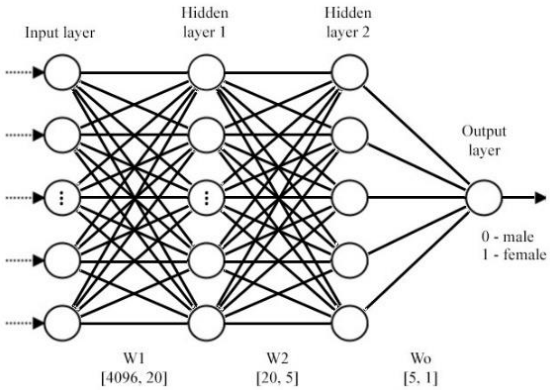

Fig.2. 3NN Deep neural network architecture.

As an evaluation tool, a successful sex recognition of provided face image from training set and test set, cost function and learning time of a model was used to ensure a selection of a model that would not be only accurate in the results of face recognition, but also it would be the fastest one to guarantee the recognition in real time. The total value of a model is computed by the following relation:

$$V = 0.8 \cdot train\_A + test\_A + \frac{max\_tt - tt}{max\_tt} \tag{19}$$

where *V* is a value of network, *train_A* is accuracy of training data, *test_A* is accuracy of test data, *max_tt* is maximum total time of model training out of all created models and *tt* is total time of current model training. By increasing the value *V* we obtain better mathematical model. The best and the worst results will be provided to not overload with unnecessary data.

A. Models Compared by Regularization Method

a) *No regularization* method in proposed main model setting and with provided data varied in results of value of network from 1.06 to 2.35.

Table 1. Models with no regularization.

Total value	**Description**			
	Settings	***Training set accuracy [%]***	***Test set accuracy [%]***	***Total training time [s]***
2.35	Gradient descent, lr = 0.005, lrd = 0	99.74	84.05	86.7958
1.06	Adam, lr = 0.005, lrd = 0.5	59.22	59.42	301.3619

As we could see in the Table 1., without use of any regularization method, the best results provides model with gradient descent using learning rate 0.005 with use of no learning decay.

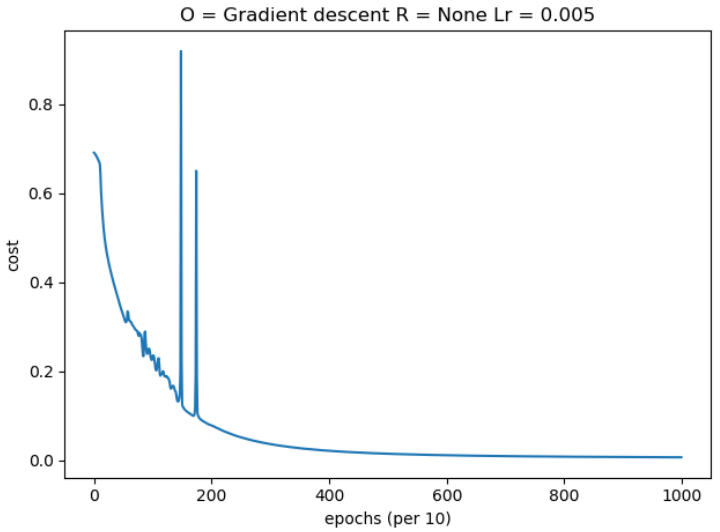

Fig.3. Cost function of model using gradient descent, no regularization and learning rate 0.005.

b) *L2 regularization* vary in value of network from 1.22 to 2.33. The bottom line is better than with use of *no regularization*, but the upper line is a bit worse, what favors *no regularization* at all in a comparison of *L2 regularization* and *no regularization* in this case.

Table 2. Models with L2 regularization.

Total value	**Description**			
	Settings	***Training set accuracy [%]***	***Test set accuracy [%]***	***Total training time [s]***
2.33	Gradient descent with momentum, lr = 0.005, lrd = 0	2.33	85.50	2.33
1.22	Adam, lr = 0.005, lrd = 0.5	1.22	59.42	1.22

By comparison of *no regularization* method and *L2*, we could conclude in poor results for *Adam* algorithm, but with *L2 regularization* in better total training time.

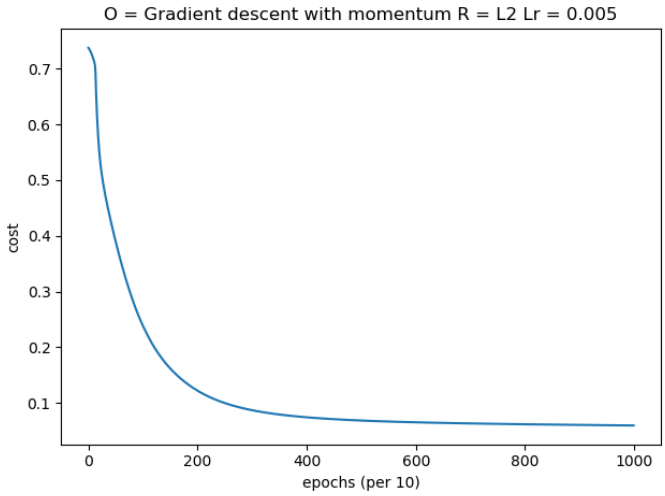

Fig.4. Cost function of model using gradient descent with momentum, L2 regularization and learning rate 0.005.

c) *Dropout regularization* results are the worst out of all regularization methods. As in other cases, in this time *Adam* has the worst results and on the other hand, *gradient descent* leads the regularization in this case too, but with overall poor results. The cost function of the best model using dropout shows us, that the algorithm can provide better results, but in cost of more epochs and so longer training time.

Table 3. Models with dropout.

Total value	**Description**			
	Settings	***Training set accuracy [%]***	***Test set accuracy [%]***	***Total training time [s]***
2.01	Gradient descent, lr = 0.005, lrd = 0	72.46	75.36	96.7259
1.13	Adam, lr = 0.005, lrd = 0	59.22	59.42	280.5022

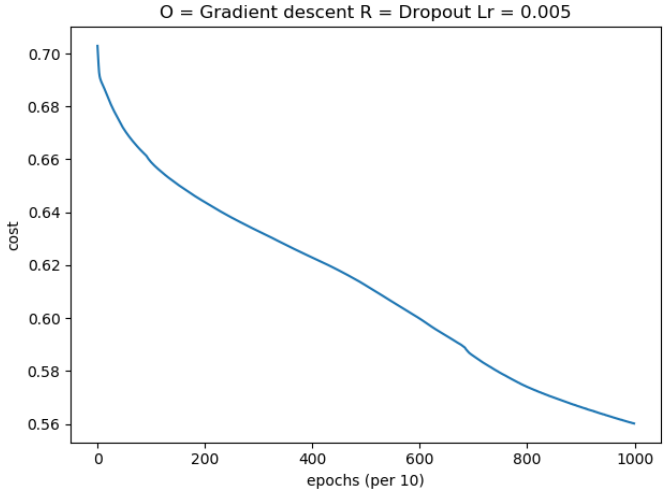

Fig.5. Cost function of a model using gradient descent, dropout regularization and learning rate 0.005.

B. Models Compared by Optimization Method

a) *Gradient Descent* is taking the all data at once and therefore it may appear slower than the other solutions, but in this case of data it was necessary to use the whole batch instead of mini-batches, because mini-batches of size of 64 or 128 samples created noise around one certain value without any sign of improvement as we can see below:

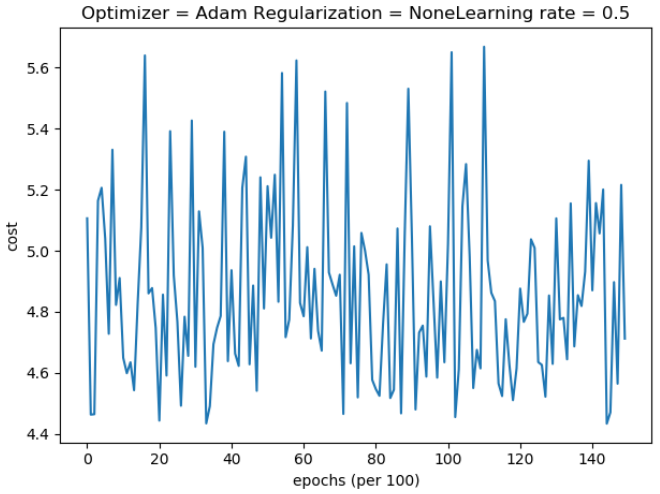

Fig.6. Use of mini-batch in experiment creating oscillations and fluctuations.

Changing the optimization or regularization technique did not get rid of the noise around one certain point and therefore the idea of using *mini-batch* in a case of this experiment was not considered, but instead a batch *gradient descent* was used. The Table 4. depicts the same settings for *no regularization* method and for *gradient descent* optimization as the best combination so far. As for *gradient descent*, the worse thing to do is pick *L2 regularization*, high learning rate and too fast *learning rate decay*.

Table 4. Models with gradient descent.

Total value	**Description**			
	Settings	***Training set accuracy [%]***	***Test set accuracy [%]***	***Total training time [s]***
2.35	No regularization, lr = 0.005, lrd = 0	99.74	84.05	86.7958
1.69	L2, lr = 0.05, lrd = 1	59.22	59.42	112.4739

b) *Gradient Descent with momentum* method in proposed main model setting and with provided data varied in results of value of network from 1.06 to 2.35.

Table 5. Models with gradient descent with momentum.

Total value	**Description**			
	Settings	***Training set accuracy [%]***	***Test set accuracy [%]***	***Total training time [s]***
2.34	No regularization, lr = 0.005, lrd = 0	99.74	84.05	87.7204
1.73	Dropout, lr = 0.0005, lrd = 1	63.63	60.86	114.5733

Gradient descent with momentum compared to *gradient descent* with same parameters gives the same result, but in a bit longer time, while the worst results are an improvement in training set accuracy as in test set accuracy.

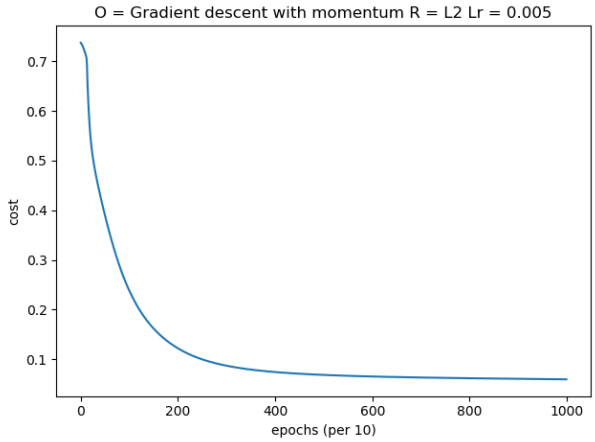

Fig.7. Cost function of model using gradient descent with momentum, L2 regularization and learning rate 0.005.

c) *Adam* optimization method brings comparable results when it comes to training and test accuracy, but the total training time is almost double of *gradient descent*, what lowers a total value of the model used with this optimization algorithm for selected problem.

Table 6. Models with Adam.

Total value	**Description**			
	Settings	***Training set accuracy [%]***	***Test set accuracy [%]***	***Total training time [s]***
2.08	No regularization, lr = 0.0005, lrd = 1	99.74	84.05	167.1711
1.06	No regularization, lr = 0.005, lrd = 0.5	59.22	59.42	301.3619

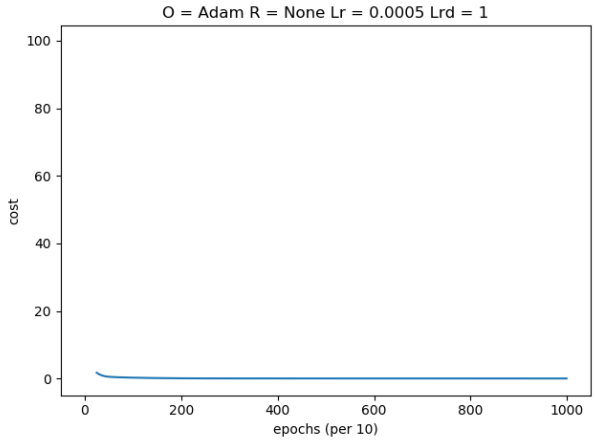

Fig.8. Cost function of a model using Adam, no regularization, learning rate 0.005 and learning decay 1.

Curve of cost function of the best Adam model was incomputable at the early stages for its high value.

d) *Learning rate decay* during experimental process acquired several values: 0, 0.5 and 1, which significantly influenced the whole learning process.

Table 7. Models with Adam.

Total value	**Description**			
	Settings	***Training set accuracy [%]***	***Test set accuracy [%]***	***Total training time [s]***
2.35	Gradient descent, No regularization, lr = 0.005, lrd = 0	99.74	84.05	86.7958
1.06	Adam, No regularization, lr = 0.005, lrd = 0.5	59.22	59.42	301.3619

Conclusion

The evaluation of several models with different regularization techniques such as models with no regularization technique at all, L2 regularization also called weight decay and dropout regularization was done. In this case, considering training set accuracy, test set accuracy and total training time, a best option would be skip the regularization completely.

Out of all optimization method, the one that works best with this kind of a problem and provided data set, a gradient descent would be the best resulting option, while Adam would be the least option. From another point of view, learning rate decay in this case has no place here and it would be the best to omit it.

If we would look at the results solely by test set accuracy, we would receive different results, a L2 regulated model with use of gradient descent with momentum optimizer, learning rate of 0.0005 and learning rate decay of 1, but in cost of total training time. Sometimes it is better to have a little bit less accurate system than have one too slow.

Acknowledgement

This paper was partially supported by the Slovak Research and Development Agency (VEGA 1/0819/17), by the Cultural and Educational Grant Agency of the Ministry of Education, Science, Research and Sport of the Slovak Republic (KEGA 030STU-4/2017), and by the Scientific Grants SEMOD-79-2/2019 and Semod-80-7/2019.

References

[1] J. Geralds, “UTKFace Large Scale Face Dataset,” github.com, [Online]. Available: https://susanqq.github.io/UTKFace/. [Accessed Sept.11, 2019].

[2] Thi-Thu-Huong Le, Jihyun Kim, and Howon Kim, “An Effective Intrusion Detection Classifier Using Long Short-Term Memory with Gradient Descent Optimization,”, International Conference on Platform Technology and Service, PlatCon 2017.

[3] S. Ruder, “An overview of gradient descent optimization algorithms”, arXiv preprint arXiv:1609.04747, 2016.

[4] M. Nielsen, “Neural networks and deep learning,” neuralnetworksanddeeplearning.com, [Online]. Available: http://neuralnetworksanddeeplearning.com/chap3.html. [Accessed Sept.11, 2019].

[5] Phaisangittisagul, E. “An Analysis of the regularization between L2 and dropout in single hidden layer neural network”. In: Proceedings of the 7th International Conference on Intelligent Systems, Modelling and Simulation, ISMS 2016, pp. 174–179. IEEE (2016)..

[6] D.R. Wilson and T.R. Martinez, “The general inefficiency of batch training for gradient descent learning,” Neural Networks, vol. 16, no. 10, pp. 1429-1451, 2003.

[7] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[8] Leon Bottou, “Large-scale machine learning with stochastic ´gradient descent,” in Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT’2010), Yves Lechevallier and Gilbert Saporta, Eds., Paris, France, August 2010, pp. 177–187, Springer

[9] W. Xu, “Towards optimal one pass large scale learning with averaged stochastic gradient descent”, CoRR, vol. abs/1107.2490, 2011

[10] Andrew Senior, Georg Heigold, Marc’aurelio Ranzato, and Ke Yang, “An empirical study of learning rates in deep neural networks for speech recognition” in Proc. ICASSP. IEEE, 2013, pp. 6724–6728

[11] A. Ng, K. Katanforoosh and Y.B. Mourri, “Learning rate decay,” coursera.com, [Online]. Available: https://www.coursera.org/learn/deep-neural-network/. [Accessed Sept.2, 2019]

[12] Diederik P. Kingma and Jimmy Ba, “Adam: A Method for Stochastic Optimization”, arXiv preprint arXiv:1412.6980, 2014.

[13] Z. Kepesiova and S. Kozak, “An effective face detection algorithm for client-side web-based solutions,” Information Technology Applications, vol. 7, no. 2, pp. 43-53, 2018.

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.

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.