JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 26 (2023) 7-12
Received: 13 September 2022
Accepted: 11 October 2022

Comparison of an effectiveness of artificial neural networks for various activation functions

Porównanie efektywności sztucznych sieci neuronowych dla różnych funkcji aktywacji

Daniel Florek*, Marek Miłosz

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

Activation functions play an important role in artificial neural networks (ANNs) because they break the linearity in the data transformations that are performed by models. Thanks to the recent spike in interest around the topic of ANNs, new improvements to activation functions are emerging. The paper presents the results of research on the effectiveness of ANNs for ReLU, Leaky ReLU, ELU, and Swish activation functions. Four different data sets, and three different network architectures were used. Results show that Leaky ReLU, ELU and Swish functions work better in deep and more complex architectures which are to alleviate vanishing gradient and dead neurons problems. Neither of the three aforementioned functions comes ahead in accuracy in all used datasets, although Swish activation speeds up training considerably and ReLU is the fastest during prediction process.

Keywords: activation functions; artificial neural networks; artificial intelligence

Streszczenie

Funkcje aktywacji, przełamując linową naturę transformacji zachodzących w sztucznych sieciach neuronowych (SSN), pozwalają na uczenie skomplikowanych wzorców występujących w danych wejściowych, np. w obrazach. Wzrost zainteresowania wokół SSN skłonił naukowców do badań wokół różnolitych aktywacji, które mogą dać przewagę podczas uczenia jak i przewidywania, ostatecznie przyczyniając się do powstania nowych, interesujących rozwiązań. W artykule przedstawiono wyniki badań nad efektywnością SSN dla funkcji ReLU, Leaky ReLU, ELU oraz Swish, przy użyciu czterech zbiorów danych i trzech różnych architektur SSN. Wyniki pokazują, że funkcje Leaky ReLU, ELU i Swish lepiej sprawdzają się w głębokich i bardziej skomplikowanych architekturach, mając za zadanie zapobieganie problemom zanikającego gradientu (ang. Vanishing Gradient) i martwych neuronów (ang. Dead neurons). Żadna z trzech wyżej wymienionych funkcji nie ma przewagi w celności (ang. Accuracy), jednakże Swish znacznie przyspiesza uczenie SSN, a ReLU jest najszybsza w procesie przewidywania.

Słowa kluczowe: funkcje aktywacji; sztuczne sieci neuronowe; sztuczna inteligencja

*Corresponding author

Email address: daniel.florek@pollub.edu.pl (D. Florek)

1. Introduction

In the last ten years Artificial Neural Networks (ANN) [1] have entered daily use in our lives, they are being utilized everywhere in the mobile phones and the internet. They are employed extensively particularly in the computer vision area, where they had most success and are irreplaceable with other machine learning algorithms. Currently ANNs are applied to solve perception problems such as object detection, image segmentation, image classification, speech recognition or language translation.

At heart ANN is a simple linear transformation combined with non-linear activation functions and backpropagation algorithm that allow it to learn complex patterns from the data provided during training. There are multiple activation functions designed for different purposes but the one that gained a lot of popularity during 2012 revolution is Rectified Linear Unit (ReLU) [2]. Due to its simplicity and speed, it is used widely in computer vision tasks that often process multiple frames per second.

Because of the role that activation functions fulfill there have been many attempts to find better performing functions in order to get higher accuracy, faster training and possibly speed up inference.

In this paper the effectiveness and efficiency of selected activation functions for different architectures and datasets is analyzed. For the purpose of this research thesis and hypotheses have been devised:

- T. Use of different activation functions results in significant changes in effectiveness of ANNs for various datasets.
- H1. Activation functions can be sorted by efficiency and effectiveness during the training process of ANN.
- H2. Activation functions can be sorted by efficiency during prediction process.
- H3. The order of activation functions doesn't change for different datasets.

2. Literature overview

There is few works that extensively compare activation functions performance across multiple datasets and architectures. Ramachandran et al. [3] propose Swish function as well as compare performance of several most known functions on CIFAR datasets [4], ImageNet and WMT 2014 English→German dataset using deep and wide architectures with high number of trainable parameters such as ResNet-164, Wide ResNet 28-10, Inception architectures, Mobile NASNet-A or attention based Transformer Architecture. According to the results Swish function wins or ties most of the time when compared to other functions, while ELU (Exponential Linear Unit) [5] and LReLU (Leaky ReLU) [6] don't seem that reliable and often underperform when compared to ReLU. Although [3] contains performance comparisons of different activation functions it mostly focuses on Swish function.

The paper [7] summarizes information on many currently used activation functions in deep learning. Authors list cons and pros of each function along with their formulas. Although the authors do not compare activation functions against each other, this work is nonetheless great source of information.

Empirical Evaluation of Rectified Activations in Convolution Network [6] compares ReLU, Leaky ReLU, PReLU (Parametric ReLU) and RReLU (Randomized Leaky ReLU) performance on CIFAR-10, CIFAR-100 and National Data Science Bowl Competition dataset. Results show that PReLU, RReLU and Leaky ReLU with $a = 5.5$ are better than ReLU. Leaky ReLU with $a = 100$ performs similar to ReLU. PReLU has lowest training error on all datasets but RReLU taking first place in test error which implies that PReLU may be overfitting while RReLU combats it and Leaky ReLU with lower a value seems to come close to RReLU.

3. Experiment Settings

3.1. Environment and Libraries

Experiment was conducted on an older machine (Table 1) using Tensorflow [8], Keras [9] and Scikit-Learn [10] libraries with Python programming language [11]. Environment was set up using Anaconda platform [12].

Table 1: Specification of the machine the experiment was conducted on

Name	Description / Version
CPU	Intel i5-4670K @4.2GHz
GPU	Nvidia GTX 1060 3GB
RAM	16GB 1600MHz
OS	Windows 10 Pro 64-bit 21H2
GPU drivers	Nvidia 511.23
CUDA	CUDA Toolkit 11.2
cuDNN	cuDNN SDK 8.1.0,
Python	Python 3.9.7
Tensorflow	Tensorflow 2.7.0
Scikit-Learn	Scikit-Learn 1.0.2
Anaconda	Anaconda 4.11.0

3.2. Selected Activation Functions

This experiment is focused on non-linear activation functions which are used in hidden layers of the ANN models. Those functions need to be fast, therefore they are simple but effective in deep learning.

Rectified Linear Unit [2] is most widely used activation function and is when looking for new functions it is often considered as a baseline in research. ReLU replaced tanh and Sigmoid activation functions because of higher performance, better generalization and being easy to optimize.

ReLU formula is:

$$f(x) = \max(0, x) \tag{1}$$

Leaky Rectified Linear Units [6] was supposed to be improvement over ReLU to alleviate potential dead neurons problem. LReLU sacrifices sparsity for a gradient which should be more robust during optimization but is less memory efficient which is important thing to consider in mobile systems. In the paper [6] LReLU performs comparably to ReLU, but authors notice slightly faster convergence. Negative values are controlled by α hyperparameter, which is usually set lower than 1.

Leaky ReLU formula is:

$$f(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \le 0 \end{cases} \tag{2}$$

Exponential Linear Unit [5] is an activation function similarly to LReLU and tries to improve on ReLU and is supposed to speed up training by alleviating vanishing gradient problem. ELU allows negative values which saturation is controlled by α hyperparameter. In the paper [5] authors state that ELUs lead to faster learning and significantly better generalization compared to ReLUs and LReLUs.

ELU formula is:

$$f(x) = f(x) = \begin{cases} x, & x > 0 \\ \alpha \exp(x) - 1, & x \le 0 \end{cases} \tag{3}$$

Swish [3] activation function was found using reinforcement learning-based search when looking for a better alternative to ReLU. Authors claim that Swish performs better in deeper models than ReLU and other activation functions.

Swish formula is:

$$f(x) = x \cdot sigmoid(x) = \frac{x}{1+e^{-x}} \tag{4}$$

Shapes of selected activation functions (1)-(4) are presented in Figure 1.

3.3. Used Datasets

Quality and nature of dataset determine the quality and performance of the trained ANN model. Size also matters as the bigger the dataset and, in classification problem, the number of output targets increases the longer the training process and possibly higher need of hardware resources. For this reason datasets of reasonable size, number of targets and that have been tested by the machine learning community or researchers, have been selected.

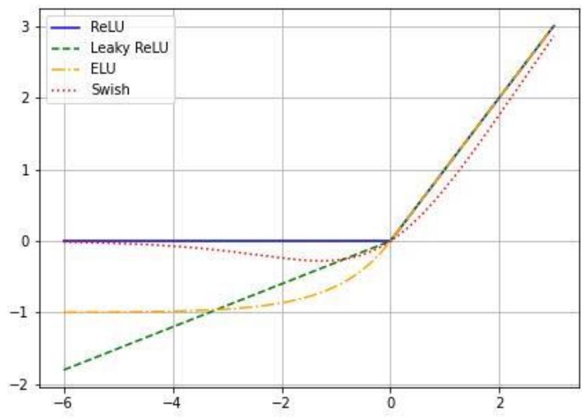

Figure 1: Activation functions shapes and values.

CIFAR-10 (Canadian Institute For Advanced Research) dataset [4] consists of 60,000 colour 32x32 pixel images split evenly between 10 classes. The targets are comprised of animals and transport vehicles. The dataset is split into training subset consisting of 50,000 images and 10,000 test images. The training subset is further split into five equal subsets for cross-validation.

CIFAR-100 dataset [4] has the same number of images and image size as CIFAR-10, but comes with 100 classes which are grouped into 20 superclasses such as fish, flowers, reptiles, people, trees, vehicles, household furniture etc. This dataset is also split in the same way as CIFAR-10, 50,000 training images which are further split into 5 equal size subsets and 10,000 test images.

Animals 10 dataset [13] has been posted on Kaggle website and it contains about 28,000 medium quality animal images with different sizes, split into 10 classes with varying number of images per class ranging from ~2,000 to ~5,000. In this research a subset of this dataset containing 1,400 of images per class has been used. Entire subsets contains 14,000 images resized to 112x112 pixels.

Intel Image Classification dataset [14] has also been put on Kaggle and it contains 14,034 training images, 3,000 test images across 6 classes and size 150x150. The dataset is almost evenly balanced and images are resized to 112x112 pixels. Classes consist of nature and urban areas scenes around the world.

3.4. Analyzed Artificial Neural Network architectures

Different architectures may have some impact on activation functions performance, therefore three architectures have been used. Two of those are well known among machine learning community and one is simple stacked convolutional layers. Layers use “same” padding with no bias vector, all layers use He normal kernel initializer [15] except logistic regression layer which uses Glorot uniform initialization [16]. During training all models use two data augmentation layers: random horizontal flip and random rotation. Strides are used instead of max pooling and after each layer batch normalization [17] and activation is applied.

ResNet [18] ANN network architecture is made of residual modules, each module consists of batch normalization, activation and convolutional layer with everything repeated two times, input of module is copied and goes through convolutional layer of size 1x1 and specified stride (1x1 or 2x2) and at the end is added to the output of the second convolutional layer. Strides in convolutional layers can be replaced by Max Pooling 2D layer. ResNet starts with convolutional layer, then several groups of residual modules with each group having one of its modules decrease the size by some factor (usually 2x2 stride), fully-connected (dense) block and one dense layer with Softmax activation (logistic regression layer) for classification.
An example of layers and connections in the residual module is presented in Figure 2.

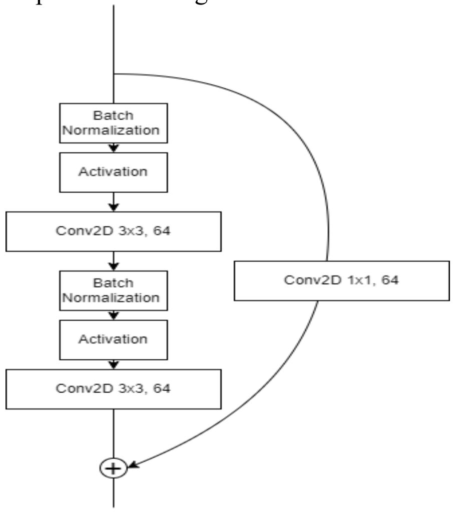

Figure 2: Residual module example.

Xception [19], similarly to ResNet, consists of Xception modules, which are residual modules but inside convolutional layers are replaced by depthwise separable convolutional layers. As the Xception architecture is based on the hypothesis that cross-channel and spatial correlations can be mapped completely separately, the input has to go through at least one convolutional layer which will help the hypothesis as cross-channel and spatial correlations tend to be very high in input images.

Simple convolutional neural network is straight flow, without any other connections consisting of multiple convolutional layers with batch normalization and activation, global max pooling, one or more fully-connected layers and logistic regression layer.

3.5. Data augmentation and supply

Data supply method can be a bottleneck during training process. One way to deal with this problem is to use tf.data.Dataset API [20] from Tensorflow library which can also help with data augmentation. In this experiment training data is loaded, cached then during training randomly augmented, shuffled and batched. Prefetch function is used to ready the data before the next epoch so the data preparation doesn’t happen at the start of the epoch. All the transformations applied in tf.data.Dataset pipeline are executed on a CPU. Data augmentation consists of random hue, saturation, brightness and contrast adjustments within specified range. Random horizontal image flip and image rotation is applied by the model layers.

3.6. Research methodology

In this paper effectiveness of an activation function is defined to be the accuracy and value of the loss function computed on the test set. Efficiency on the other hand is defined as the prediction time, epochs required to converge and time required for a single epoch. Research has been conducted with the use of the stratified k-fold Cross Validation [21] for each dataset and activation function. For CIFAR-100 and Intel Image Classification we use variations of Xception architecture, while for CIFAR-10 and animals 10 ResNet and simple convolutional neural network has been employed respectively. Dataset is loaded, shuffled, split into training and test subsets (if there is no provided separate test set) then the training subset is split into 5 stratified subsets of even size. During training four of the splits are used for training and the remaining one for validation. This is repeated five times with different split used for validation. Each activation function is trained five times for every dataset. Training and validation loss and accuracy are recorded along with time spent on each epoch.

Each trained artificial network model uses cross-entropy as a loss function. Only the model with best loss value across all epochs is saved. Early stopping with 10 epoch patience is used to investigate the speed of convergence for each activation function. After training every model is evaluated for test loss, accuracy and prediction time.

During training process Adam [22] optimization algorithm with initial learning rate 0.001 and exponential decay for all models is used.

Activation function hyperparameters have been set to following values:

- $\alpha = 0.3$ for Leaky ReLU,
- $\alpha = 1.0$ for ELU.

4. Research results analysis

In this research more shallow models are trained, with fewer hidden layers and thus smaller number of trainable parameters compared to experiments conducted by Ramachandran et al. [3]. Results tables present mean values of all splits for each activation function.

Models trained on CIFAR-10 dataset use ResNet architecture with 283,595 trainable parameters and 1,628 non-trainable parameters (Table 2). It should be noted that it is deeper than wider as the number of filters is rather low. Batch size is set to 128. The results of experiment are presented in Table 3.

In this particular combination of dataset and architecture ELUs give the best accuracy and loss. It is important to note that ELUs have also trained for highest amount of epochs with mean of 65.4 (Table 3). Second come Swish activations with accuracy lower by value of 0.4% but with significantly lower number of epochs before training stop. ReLUs are fastest during prediction, ELU and LReLU have comparable prediction time. Difference in mean time spent on one epoch can be mostly attributed to resources taken by system.

Table 2: Model layers for CIFAR-10 dataset

Layer name	Layer description
Input	Shape (32,32,3)
Conv2D	Size 5x5, filters 32
Residual module 1	Size 3x3, filters 16, strides 2
Residual module 2	Size 3x3, filters 16
Residual module 3	Size 3x3, filters 16
Residual module 4	Size 3x3, filters 32, strides 2
Residual module 5	Size 3x3, filters 32
Residual module 6	Size 3x3, filters 32
Residual module 7	Size 3x3, filters 64, strides 2
Residual module 8	Size 3x3, filters 64
Residual module 9	Size 3x3, filters 64
Global Average Pooling 2D	
Dense	Units 128
Dense (Classification)	Units 128, activation Softmax

Table 3: Experiment results for CIFAR-10 dataset

Func.	Test Acc.	Test Loss	Val. Loss	Pred. Time	S. Epoch	Epoch Time
ReLU	0.7758	0.6666	0.6526	0.55	46.2	21.99
LReLU	0.7806	0.6468	0.6275	0.58	60.6	21.38
ELU	0.7927	0.6159	0.6098	0.58	65.4	21.16
Swish	0.7881	0.6358	0.6132	0.62	51.2	22.36

Xception architecture was employed for training models on CIFAR-100 dataset with 1,421,956 trainable parameters and 6,208 non-trainable parameters (Table 4). Here Dropout layer has been used, for better generalization and Max Pooling for downsampling layers output instead of strides in convolutional layers. Batch size is set to 128.

Table 4: Model layers for CIFAR-100 dataset

Layer name	Layer description
Input	Shape (32,32,3)
Conv2D 1	Size 5x5, filters 64
Conv2D 2	Size 3x3, filters 96
Xception module 1	Size 3x3, filters 192, max pooling 2x2 strides 2
Xception module 2	Size 3x3, filters 256, max pooling 2x2 strides 2
Xception module 3	Size 3x3, filters 512, max pooling 2x2 strides 2
Separable Conv2D	Size 3x3, filters 512
Global Average Pooling 2D	
Dense	Units 512
Dropout	Rate 0.5
Dense (classification)	Units 100, activation Softmax

Table 5: Experiment results for CIFAR-100 dataset

Func.	Test Acc.	Test Loss	Val. Loss	Pred. Time	S. Epoch	Epoch Time
ReLU	0.5736	1.7489	1.7613	2.17	33.4	53.4
LReLU	0.6118	1.6058	1.6264	2.45	43.6	52.6
ELU	0.6098	1.6210	1.6388	2.43	39.8	51.7
Swish	0.5981	1.6360	1.6647	2.84	26.6	59.9

We can see that LReLUs perform the best, with small difference to ELU activation (Table 5). Again we can see that worst performing function has trained for much lower number of epochs on average but Swish seems to have very fast convergence compared with other activations. Again Swish takes longest amount of time for predictions and single epoch.

For Animals 10 datasets we used Simple Convolutional neural network architecture with 598 218 trainable parameters and 1 792 non-trainable parameters (Table 6). In this particular architecture Global Max Pooling 2D has been used, instead of Global Average Pooling. Batch size is set to 8.

Table 6: Model layers for Animals 10 dataset

Layer name	Layer description
Input	Shape (112,112,3)
Conv2D 1	Size 7x7, filters 64, strides 2
Conv2D 2	Size 3x3, filters 64
Conv2D 3	Size 3x3, filters 128, strides 2
Conv2D 4	Size 3x3, filters 128
Conv2D 5	Size 3x3, filters 128, strides 2
Conv2D 6	Size 3x3, filters 128
Global Max Pooling 2D	
Dense	Units 128
Dense	Units 128
Dropout	Rate 0.5
Dense (classification)	Units 10, activation Softmax

Table 7: Experiment results for Animals 10 dataset

Func.	Test Acc.	Test Loss	Val. Loss	Pred. Time	S. Epoch	Epoch Time
ReLU	0.6907	0.9131	0.9346	0.98	41.8	25.4
LReLU	0.6824	0.9405	0.9530	1.05	54.0	23.9
ELU	0.6782	0.9574	0.9937	1.05	44.2	26.5
Swish	0.6625	0.9897	0.9987	1.23	30.8	28.3

Surprisingly ReLUs result in the best mean test accuracy and loss, with second to Swish activations lowest number of epochs (Table 7). Again ReLUs have the shortest prediction time, LReLUs and ELUs perform similiarly in this regard and Swish activations take longest time to predict. Swish activations also have longest mean time spent on single epoch.

Models trained on Intel Image Classification dataset use Xception architecture with 668,038 trainable parameters and 4,736 non-trainable parameters (Table 8). In comparison to CIFAR-100 models here we have a bit deeper models but lower number of filters or units in layers. Batch size is set to 16.

Table 8: Model layers for Intel Image Classification dataset

Layer name	Layer description
Input	Shape (112,112,3)
Conv2D	Size 5x5, filters 64
Conv2D	Size 3x3, filters 64
Xception module 1	Size 3x3, filters 96, max pooling 2x2 strides 2
Xception module 2	Size 3x3, filters 128, max pooling 2x2 strides 2
Xception module 3	Size 3x3, filters 192, max pooling 2x2 strides 2
Xception module 4	Size 3x3, filters 256, max pooling 2x2 strides 2
Separable Conv2D	Size 3x3, filters 512
Global Average Pooling 2D	
Dense	Units 256
Dense	Units 128
Dropout	Rate 0.5
Dense (classification)	Units 6, activation Softmax

Table 9: Experiment results for Intel Image Classification dataset

Func.	Test Acc.	Test Loss	Val. Loss	Pred. Time	S. Epoch	Epoch Time
ReLU	0.8546	0.4757	0.4770	1.70	32.4	44.5
LReLU	0.8561	0.4720	0.4546	1.76	41.4	46.8
ELU	0.8517	0.4663	0.4499	1.76	37.0	45.8
Swish	0.8594	0.4568	0.4485	2.05	31.4	49.6

In this combination of dataset and architecture we have the closest results between all activation functions in terms of accuracy and loss so far, with Swish taking the crown (Table 9). Again prediction time results are similar to previous results. Swish activations need lower number of epochs to converge but take the longest time for single epoch.

5. Conclusions

From the results a conclusion can be drawn that H1 and H2 are true but H3 is only partially true as the activation functions order of effectiveness and efficiency changes in every dataset and architecture combinations.

In regards to H1 it is true that although Swish activations take the longest time to train for a single epoch they also need significantly lower number of epochs to converge thus decreasing total training time. More complex models seem to favor activations that try to alleviate vanishing gradient and dead neurons problems although the functions are not consistent in their effectiveness across all datasets.

Without a doubt the H2 is true and the order doesn't change for different datasets as is stated in H3. Swish activations slow down prediction time by value ranging from ~12.7% up to ~30.9% compared to ReLUs. LReLUs and ELUs are comparable in terms of prediction time but are still slower than ReLUs by varying value of ~3.5% to ~13%. This may have considerable impact on performance of cloud services that employ ANN models especially in moments of high traffic.

It should be noted that Swish may be comparable in terms of effectiveness or possibly better than LReLU and ELU since Early Stopping ends training early before learning rate has a chance to decrease to a value which can make finer changes to the model. Another thing to note is that authors of Searching for activation functions [5] have made their experiments on deep models with higher number of parameters than the models presented in this paper. This has effect on the performance of activation functions as the selected functions try to alleviate vanishing gradient and dead neurons problems which are less likely to occur in shallow models.

Ultimately research on performance of activation functions is not nearly completed by the results presented in this article. Future experiments should focus on verifying activation functions efficiency and effectiveness for different value of learning rate and its decay, parameters affecting activation function themselves such as α in LReLU or ELU, broader selection of architectures and datasets.

References

[1] A. Abraham, Artificial neural networks. Handbook of measuring system design, John Wiley and Sons Ltd., London (2005) 901-908, https://doi.org/10.1002/0471497398.mm421.

[2] V. Nair, G. E. Hinton, Rectified Linear Units Improve Restricted Boltzmann Machines in Proceedings of the 27th International Conference on International Conference on Machine Learning, Omnipress, Madison (2010) 807-814.

[3] P. Ramachandran, B. Zoph, Q. V. Le, Searching for activation functions, arXiv (2017), https://doi.org/10.48550/arXiv.1710.05941.

[4] A. Krizhevsky, V. Nair, G. E. Hinton, CIFAR-10 and CIFAR-100 datasets http://www.cs.toronto.edu/~kriz/cifar.html, [14.06.2022].

[5] D. A. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (elus), Published as a conference paper at ICLR 2016 (2015), https://doi.org/10.48550/arXiv.1511.07289.

[6] B. Xu, N. Wang, T. Chen, M. Li, Empirical evaluation of rectified activations in convolutional network, arXiv (2015), https://doi.org/10.48550/arXiv.1505.00853.

[7] C. Nwankpa, W. Ijomah, A. Gachagan, S. Marshall, Activation functions: Comparison of trends in practice and research for deep learning, arXiv (2018), https://doi.org/10.48550/arxiv.1811.03378.

[8] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean et al., TensorFlow: A System for Large-Scale Machine Learning, OSDI 16 (2016) 265-283.

[9] Keras, https://keras.io , [14.06.2022].

[10] F. Pedregosa et al., Scikit-learn: Machine Learning in Python, JMLR 12 (2011) 2825-2830, https://doi.org/10.48550/arXiv.1201.0490.

[11] G. Van Rossum, F. L. Drake, Python 3 Reference Manual, CA: CreateSpace, Scotts Valley, 2009.

[12] Anaconda platform website https://anaconda.org/, [14.06.2022].

[13] Animals 10 dataset https://www.kaggle.com/datasets/alessiocorrado99/animals10 , [14.06.2022].

[14] Intel Image Classification dataset https://www.kaggle.com/datasets/puneet6060/intel-image-classification , [14.06.2022].

[15] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification, arXiv (2015), https://doi.org/10.48550/arXiv.1502.01852.

[16] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, Journal of Machine Learning Research - Proceedings Track 9 (2010) 249-256.

[17] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, International conference on machine learning, PMLR 37 (2015) 448-456, https://doi.org/10.48550/arXiv.1502.03167.

[18] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, Proceedings of the IEEE conference on computer vision and pattern recognition (2016) 770-778, https://doi.org/10.48550/arXiv.1512.03385.

[19] F. Chollet, Xception: Deep learning with depthwise separable convolutions, Proceedings of the IEEE conference on computer vision and pattern recognition (2017) 1251-1258, https://doi.org/10.1109/CVPR.2017.195.

[20] tf.data.Dataset API https://www.tensorflow.org/api_docs/python/tf/data/Dataset , [20.06.2022].

[21] P. Refaeilzadeh, L. Tang, H. Liu, Cross-Validation. Encyclopedia of Database Systems. Springer, Boston (2009), https://doi.org/10.1007/978-0-387-39940-9_565.

[22] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv (2014), https://doi.org/10.48550/arXiv.1412.6980.