Data Augmentation Using Principal Component Resampling for Image Recognition by Deep Learning

Olusola Oluwakemi Abayomi-Alli[1], Robertas Damaševičius[1,2], Michał Wieczorek[2], and Marcin Woźniak[2(✉)]

[1] Department of Software Engineering, Kaunas University of Technology, Kaunas, Lithuania
{olusola.abayomi-alli,robertas.damasevicius}@ktu.edu
[2] Faculty of Applied Mathematics, Silesian University of Technology, Gliwice, Poland
michal_wieczorek@hotmail.com, marcin.wozniak@polsl.pl

Abstract. Image recognition by deep learning usually requires many sample images to train. In case of a small number of images available for training, data augmentation techniques should be applied. Here we propose a novel image augmentation technique based on a random permutation of coefficients of within-class principal components obtained after applying Principal Component Analysis (PCA). After reconstruction, newly generated surrogate images are employed to train a deep network. In this study, we demonstrated the applicability of our approach on training a custom convolutional neural network using the CIFAR-10 image dataset. The experimental results show an improvement in terms of classification accuracy and classification ambiguity.

Keywords: Image recognition · Convolutional neural network · Principal component analysis · Data augmentation · Small data · Deep learning

1 Introduction

The neural network (NN) is considered as one of main models of deep learning. The advantage of NN is the ability to effectively learn useful domain features in diverse areas such as image and signal processing [1]. This ability enables the neural network to learn deep models on domain data, which have proven successful in numerous areas of Artificial Intelligence (AI) such as object detection, defect recognition, speech recognition, voice evaluation, remote sensing, and medical decision support. Convolutional neural networks (CNNs) have been popularly used in computer vision and other related fields [2]. Recently, a lot of very large-scale deep CNN models were proposed such as VGG and ResNet. However, previous studies showed that despite increase in accuracy, oversized deep neural network models contribute to generate a lot of redundant features which are either the shifted version of one another or are closely related or display slight or no variations, thus resulting in redundant computations [3]. However, many parameters to be trained can be a disadvantage when training is performed on a limited amount of data available.

L. Rutkowski et al. (Eds.): ICAISC 2020, LNAI 12416, pp. 39–48, 2020.
https://doi.org/10.1007/978-3-030-61534-5_4

Data augmentation can be applied for training of neural network models to enhance the classification accuracy and model performance. The main idea of data augmentation is that the transformations applied to the already labeled data result in new, surrogate training data. Image augmentation techniques include geometric image transforms, mixing images, color space transforms, feature space augmentation, kernel filters, random erasing, etc. [4]. Data augmentation is relevant in case of small data problem [5], when a dataset is too small to train a deep neural network effectively.

The aim of this paper is to propose a novel image augmentation technique based on random permutation of coefficients of within-class principal components (PCs) obtained after Principal Component Analysis (PCA). The remaining parts of the paper are as follows: related work is presented in Sect. 2, while Sect. 3 discusses the proposed methods with detailed description. Section 4 discusses results and compares with known state-of-the-art methods. Finally, the paper concludes in Sect. 5.

2 Related Work

The use of data augmentation techniques has been considered in several recent papers. Leng et al. [6] presented joint Bayesian analysis for augmenting, while Chen et al. [7] proposed fuzzy operation rules for developing new data attributes and increasing data dimensionality for the small dataset learning. Truong et al. [8] presented augmentation methods based on 2D image sequence and 3D transformation. The classification model used was cascaded fully convolutional neural architecture. Li et al. [9] suggested to pair adjacent pixels and to use their combinations as additional data for hyperspectral image classification with deep CNN. Haut et al. [10] used random occlusions to generate new images for training of CNN for hyperspectral image classification. Finally, our proposed method has similarity to method for microscopy images proposed by Drivinskas et al. [11], however they use a different (multiplication) based scheme to modify principal components for augmentation. Similarly in [21] Najgebauer et al. proposed also deep learning based microscopic image processing for special sampling. Some other examples of data augmentation were given in [22] and [23]. Despite their usefulness, the existing data augmentation methods have limitations such as over-fitting, high computational time, poor accuracy of models, etc. In this article, a novel image augmentation technique based on a random permutation of coefficients of within-class PCs obtained after PCA. After image reconstruction, new images are used to train a deep network.

3 Proposed Method

This section presents a detailed description of the neural network models and the data augmentation techniques used in this study as depicted in Fig. 1.

3.1 Neural Network

This study focuses on small data [5] and tiny neural networks [12] for object recognition, these restrictions were applied for the design of the neural network. We do not use the

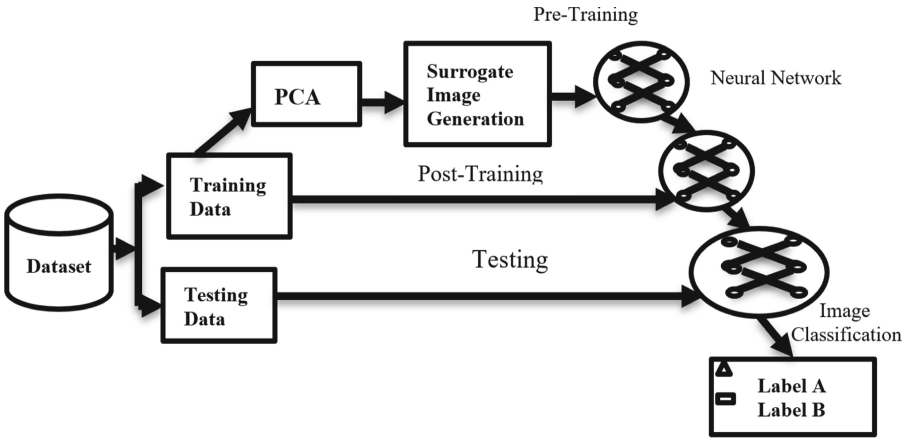

Fig. 1. Outline of the proposed method

ensemble models [13] and focus on a simpler structure. We did not adopt long training with more than 300 epochs (as suggested in [14]) due to hardware and time limitations. Different from very large very deep network models, we focused on a simple custom model allowing to demonstrate the advantages of data augmentation.

A 15-layer CNN with one input layer followed by 13 hidden layers and one output layer was designed (Fig. 2). The input layer consists of $32 \times 32 \times 3$ pixel images, i.e. it has 3072 neurons. The first hidden layer is the convolution layer 1 which is responsible for feature extraction from an input data. This layer performs convolution operation to small localized areas by convolving a $5 \times 5 \times 3$ filter with the previous layer. Rectified linear unit (ReLU) is used as an activation function at the end of each convolution layer to enhance the performance of the model. The next max pooling layers are used after each ReLU layer to reduce the output from the convolution layer and diminish the complexity of the model. The layer is followed by the convolution layer 2, ReLU layer 2 and pooling layer 2, operate in the same way except for their feature maps and kernel size varies. These are followed by a third set of layers (convolution layer 3, ReLU layer 3 and pooling layer 3). A fully connected layer FC 1 with 576 inputs and 64 outputs is followed by the final ReLU layer 4 and final fully connected layer FC 2 with 64 inputs and 10 outs, each corresponding to the target class. Using the FC layers is essential for the wider datasets, which have fewer examples per class for training [15]. Finally, softmax was employed as predictor to distinguish the classes. For optimization, we used the stochastic gradient descent with momentum (SGDM) optimizer with a learning rate of $\alpha = 0.001$, a learning rate drop factor of 0.1 and learning rate drop period of 8. The network is trained for 40 epochs.

3.2 Data Augmentation

In this study, we use a lower-dimensional representation of images obtained using Principal Component Analysis (PCA). PCA performs data decomposition into multiple orthogonal principal components (PCs) using the variance criterion. The PCs are the

	Name	Type	Activations	Learnables
1	imageinput 32x32x3 images with 'zerocenter' normalization	Image Input	32×32×3	-
2	conv_1 32 5x5x3 convolutions with stride [1 1] and padding [2 2 2 2]	Convolution	32×32×32	Weights 5×5×3×32 Bias 1×1×32
3	relu_1 ReLU	ReLU	32×32×32	-
4	maxpool_1 3x3 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	15×15×32	-
5	conv_2 32 5x5x32 convolutions with stride [1 1] and padding [2 2 2 2]	Convolution	15×15×32	Weights 5×5×32×32 Bias 1×1×32
6	relu_2 ReLU	ReLU	15×15×32	-
7	maxpool_2 3x3 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	7×7×32	-
8	conv_3 64 5x5x32 convolutions with stride [1 1] and padding [2 2 2 2]	Convolution	7×7×64	Weights 5×5×32×64 Bias 1×1×64
9	relu_3 ReLU	ReLU	7×7×64	-
10	maxpool_3 3x3 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	3×3×64	-
11	fc_1 64 fully connected layer	Fully Connected	1×1×64	Weights 64×576 Bias 64×1
12	relu_4 ReLU	ReLU	1×1×64	-
13	fc_2 10 fully connected layer	Fully Connected	1×1×10	Weights 10×64 Bias 10×1
14	softmax softmax	Softmax	1×1×10	-
15	classoutput crossentropyex with 'airplane' and 9 other classes	Classification Output	-	-

Fig. 2. Architecture of neural network and its parameters

projections of the data along the eigenvectors of the covariance matrix of a dataset. The first PC is the axis with the most variance and each subsequent PC is calculated in the order of decreasing variance. The first PCs are the most significant, while the last ones are considered to represent only the "noise".

First, PCA discovers the eigenvectors and their matching eigenvalues of the covariance matrix of a data set and the eigenvectors are sorted by their decreasing eigenvalues. Given a dataset $\chi = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$ of samples drawn from a data source representing a specific class $\mathbb{C}$, and the covariance matrix C of the data set; the eigenvectors E are found by solving equation

$$\mathrm{CE} = \lambda \mathrm{E}, \tag{1}$$

here λ is the eigenvalue that matches E. Each eigenvector e_i can be expressed as

$$\boldsymbol{e}_i = \sum\nolimits_j \alpha_j^i \boldsymbol{x}_j, \tag{2}$$

The original data can be reconstructed by multiplying principal components with their loadings W as follows:

$$\hat{\mathrm{X}} = \mathrm{WE} \tag{3}$$

Now each eigenvector e_i represents a specific independent aspect of data samples belonging to the class $\mathbb{C}$. Next, we perform random reshuffling of these values:

$$\hat{W} = \Gamma W, \tag{4}$$

where Γ is a random permutation operator applied with a specific probability of p.

Then the modified image dataset is reconstructed using the reshuffled loadings $\tilde{W}$ and eigenvectors E. Note that in order to avoid excessive variability in the surrogate images, we did not perform permutation of loading on first two PCs, which encode the most essential information of image class. The outcomes of image augmentation are illustrated in Fig. 3 and the image augmentation method is summarized as:

Fig. 3. Illustration of image augmentation: original images (left) and surrogate (augmented) images (right)

1. Compute PCs for each class in dataset using classical PCA.
2. Perform random permutations of the PC loadings (with a predefined probability) starting from the loading representing the third principal component.
3. Construct surrogate images using the randomly permuted loadings.
4. Use the surrogate image dataset to perform pre-training of a neural network.
5. Freeze the learned weights of the selected layers of the neural network and perform post-training using the original (unchanged) training dataset.

The computational complexity of the method is determined by the calculation of PCA, which is $O\left(min\left(p^3, n^3\right)\right)$, here p is the number of pixels in an image, and n is the number images. For our experiments we construct different surrogate image datasets using 2%, 3%, 4%, 5%, 7%, 10%, 15%, 20%, 25% and 30% of permutations of the PC loadings. We generated 1000 new images for each image class and used them for network pre-training. For final training, we used the original images from the training set, while we explored different training scenarios: 1) Freeze only the first convolutional layer (CL) conv_1; 2) Freeze two first CLs conv_1 and conv_2; 3) Freeze all CLs conv_1 and conv_2, conv_3.

4 Experimental Results

We use the CIFAR-10 dataset [16], which is a known benchmark dataset in image classification and recognition domain. The dataset has 60,000 32 $\times$ 32 color images between 10 different classes, which represent the images of both natural (birds, cats, deer, dogs, frogs, horses) and artificial (airplanes, cars, ships, and trucks) objects. The dataset has 6,000 images of each class. The training set has 50,000 images (equally balanced), while a testing set has 10,000 images (1,000 images of each class). The classes do not overlap and they are fully mutually exclusive.

We evaluated the performance using accuracy and uncertainty of classification based on ambiguity, i.e. the ratio of the second-largest probability to the largest probability of the softmax layer activations. The ambiguity is between zero (nearly certain classification) and 1 (the network is unsure of the class due to inability to learn the differences between them). We also evaluated the mean value of accuracy and ambiguity over the testing image set and the results are depicted in Tables 1 and 2.

Table 1. Accuracy improvement with image augmentation using CIFAR-10 dataset (larger values are better, best value is shown in bold). All improvement values are given with respect to the accuracy of baseline network on testing data without any image augmentation applied

PC loadings reshuffled, %	Accuracy improvement, % (first CL frozen)	Accuracy improvement, % (first two CLs frozen)	Accuracy improvement, % (all CLs frozen)
2	3.6600	−0.4900	−3.0900
3	6.5400	4.2400	2.9900
4	6.1200	4.3900	2.7500
5	3.5500	1.2800	−0.6100
6	4.7300	2.2900	0.5000
7	6.0100	2.4100	0.9600
10	3.7200	1.1700	−0.1600
15	4.9200	1.8700	0.4400
20	6.3700	5.3100	3.9100
25	**7.1800**	4.8500	3.0500
30	5.8200	5.0700	3.6700
Mean	5.3291	2.9445	1.3100
Std. dev	1.2795	1.9260	2.1679

The results show that best improvement in accuracy is achieved using a neural network pretrained with surrogate images generated with 25% of principal component loadings reshuffled and then post-trained with the weights of the first CL frozen (the improvement is significant at $p < 0.001$ using one-sample t-test). In terms of classification ambiguity, the best results are also obtained with the weights of only the first CL frozen (the reduction is significant at $p < 0.001$ using one-sample t-test).

The results are also summarized in Fig. 4. Note that here we presented the reduction of error rate instead of the improvement of accuracy for better comparison. These results show that larger shuffling rates lead to better results, while the best results are achieved by leaving the first CL frozen while retraining other layers.

For visualization of the activation maps, we use t-distributed stochastic neighbor embedding (t-SNE). The method uses a nonlinear map that attempts to preserve distances and maps network activations in a layer to two dimensions. See the results for the fully

Table 2. Classification ambiguity with image augmentation using CIFAR-10 dataset (smaller values are better, best value is shown in bold)

PC loadings shuffled,%	Ambiguity (first CL frozen)	Ambiguity (first two CLs frozen)	Ambiguity (all CLs frozen)
2	0.1552	0.2408	0.3206
3	0.0811	0.1102	0.1612
4	0.0871	0.1088	0.1632
5	0.1367	0.1714	0.2348
6	0.1202	0.1533	0.2035
7	0.0929	0.1408	0.1890
10	0.1319	0.1782	0.2456
15	0.1210	0.1476	0.2252
20	0.0883	0.1054	0.1499
25	**0.0788**	0.1014	0.1554
30	0.0901	0.0974	0.1366
Mean	0.1076	0.1414	0.1986
Std. dev	0.0262	0.0438	0.0547

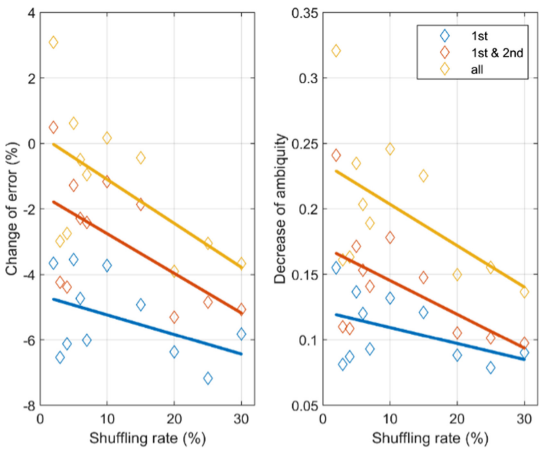

Fig. 4. Classification error and ambiguity vs principal component shuffling rate with trend lines. The results are shown after the network was post-trained with original training dataset with its 1st, 1st and 2nd, and all convolutional layers frozen, respectively.

connected fc2 layer in Fig. 5. One can see that the network tends to put natural and artificial object classes closer. This may mean that any misclassifications arise due to the similarity of semantically close classes such as dogs and cats.

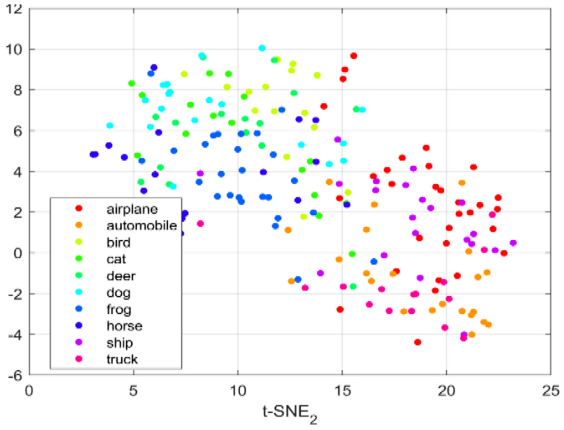

Fig. 5. Activation visualization of the final fully connected layer using t-SNE

The examples of misclassification by the neural network are presented in Fig. 6. They confirm that most misclassifications are between similar classes such as dog and cat images.

Fig. 6. Examples of misclassifications: images classified as dog, dog, dog, bird (top row), bird, frog, cat, deer (bottom row)

In order to visualize the features learned by the network, we use DeepDream [17] to obtain images that fully activate a specific channel of the network layers. The results for conv2 and conv3 layers are presented in Fig. 7.

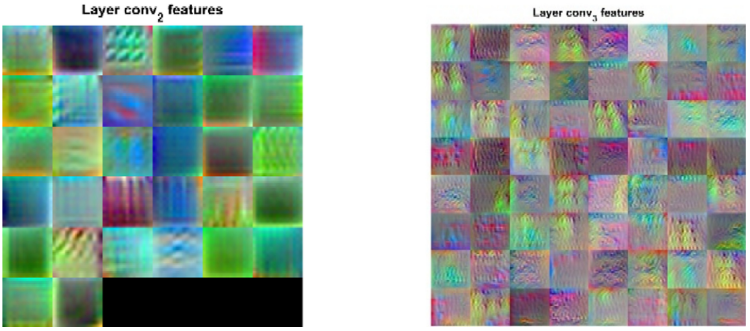

Fig. 7. Feature visualization of outputs from convolutional layers conv2 and conv3

Finally, a comparison of the results of our approach with the results of other authors in Table 3. Our results compare well in the context of other state-of-the art methods, considering that we used a smaller neural network with only 15 layers.

Table 3. Comparison with other works using CIFAR-10 dataset

Method description	Error rate (%)	Reference
Maxout Networks	9.38	[18]
Densely Connected CNN-BC (100-layer DenseNet)	6.3	[19]
CNN with PCA based data augmentation	5.34	This paper
16-layer-deep Wide Residual Network	3.8	[20]

5 Conclusion

This paper presents a novel image data augmentation technique based on the random permutation of coefficients of within-class principal component scores obtained after Principal Component Analysis (PCA). After reconstruction, the newly generated surrogate images are used to pretrain a deep network (we used a custom 15-layer convolutional neural network). Then one or more convolutional layers of the neural network were frozen and the final training was performed using the original images.

This study also showed the practical applicability of our approach on training the custom-made neural network using CIFAR-10 image dataset. The approach allowed both to improve accuracy (up to 7.18%) and reduce ambiguity of classification. Thus, it can be used for addressing the small data problem, when there is only a small number of images available for training a neural network.

In future work, we will examine and compare our approach with other types of image dataset augmentation approaches. Also, we will explore the use of dropout and batch regularization to improve the accuracy of the custom neural network.

Acknowledgments. Authors acknowledge contribution to this project of the Program "Best of the Best 4.0" from the Polish Ministry of Science and Higher Education No. MNiSW/2020/43/DIR/NN4.

References

1. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015). https://doi.org/10.1016/j.neunet.2014.09.003
2. Rawat, W., Wang, Z.: Deep convolutional neural networks for image classification: a comprehensive review. Neural Comput. **29**(9), 2352–2449 (2017)
3. Ayinde, B.O., Inanc, T., Zurada, J.M.: Regularizing deep neural networks by enhancing diversity in feature extraction. IEEE Trans. Neural Netw. Learning Syst. **30**(9), 2650–2661 (2019). https://doi.org/10.1109/TNNLS.2018.2885972

4. Shorten, C., Khoshgoftaar, T.M.: A survey on image data augmentation for deep learning. J. Big Data **6**(1), 1–48 (2019). https://doi.org/10.1186/s40537-019-0197-0
5. Qi, G.-J., Luo, J.: Small data challenges in big data era: a survey of recent progress on unsupervised and semi-supervised methods. CoRR abs/1903.11260 (2019)
6. Leng, B., Yu, K., Jingyan, Q.I.N.: Data augmentation for unbalanced face recognition training sets. Neurocomputing **235**, 10–14 (2017)
7. Chen, H.Y., Li, D.C., Lin, L. S.: Extending sample information for small data set prediction. In: 2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), pp. 710–714 (2016)
8. Truong, T.N., Dam, V.D., Le, T.S.: Medical images sequence normalization and augmentation: improve liver tumor segmentation from small dataset. In: 3rd International Conference on Control, Robotics and Cybernetics (CRC), pp. 1–5 (2018)
9. Li, W., Chen, C., Zhang, M., Li, H., Du, Q.: Data augmentation for hyperspectral image classification with deep CNN. IEEE Geosci. Remote Sens. Lett. **16**(4), 593–597 (2019)
10. Haut, J.M., Paoletti, M.E., Plaza, J., Plaza, A., Li, J.: Hyperspectral image classification using random occlusion data augmentation. IEEE Geosci. Remote Sens. Lett. **16**(11), 1751–1755 (2019). https://doi.org/10.1109/LGRS.2019.2909495
11. Dirvanauskas, D., Maskeliūnas, R., Raudonis, V., Damaševičius, R., Scherer, R.: Hemigen: human embryo image generator based on generative adversarial networks. Sensors **19**(16), 3578 (2019)
12. Womg, A., Shafiee, M.J., Li, F., Chwyl, B.: Tiny SSD: a tiny single-shot detection deep convolutional neural network for real-time embedded object detection. In: 15th Conference on Computer and Robot Vision (CRV), Toronto, ON, pp. 95–101 (2018)
13. Zhou, Z.-H., Wu, J., Tang, W.: Ensembling neural networks: Many could be better than all. Artif. Intell. **137**(1–2), 239–263 (2002). https://doi.org/10.1016/s0004-3702(02)00190-x
14. Amory, A.A., Muhammad, G., Mathkour, H.: Deep convolutional tree networks. Future Generation Comput. Syst. **101**, 152–168 (2019). https://doi.org/10.1016/j.future.2019.06.010
15. Basha, S.H.S., Dubey, S.R., Pulabaigari, V., Mukherjee, S.: Impact of fully connected layers on performance of convolutional neural networks for image classification. Neurocomputing **378**, 112–119 (2019). https://doi.org/10.1016/j.neucom.2019.10.008
16. Krizhevsky, A.: Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto, Canada (2009)
17. Mordvintsev, A., Olah C., Tyka, M.: Inceptionism: Going deeper into neural networks. Google research blog (2015)
18. Goodfellow, I.J., Warde-Farley, D., Mirza, M., Courville, A., Bengio, Y.: Maxout networks. arXiv:1302.4389 (2013)
19. Huang, G., Liu, Z., Weinberger, K. Q., van der Maaten, L.: Densely connected convolutional networks. arXiv preprint arXiv:1608.06993 (2016)
20. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv:1605.07146 (2016)
21. Najgebauer, P., Grycuk, R., Rutkowski, L., Scherer, R., Siwocha, A.: Microscopic sample segmentation by fully convolutional network for parasite detection. In: Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, Jacek M. (eds.) ICAISC 2019. LNCS (LNAI), vol. 11508, pp. 164–171. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-20912-4_16
22. Aizenberg, I., Luchetta, A., Manetti, S., Piccirilli, M.C.: A MLMVN with arbitrary complex-valued inputs and a hybrid testability approach for the extraction of lumped models using FRA. J. Artif. Intell. Soft Comput. Res. **9**(1), 5–19 (2019)
23. Costa, M., Oliveira, D., Pinto, S., Tavares, A.: Detecting driver's fatigue, distraction and activity using a non-intrusive Ai-based monitoring system. J. Artif. Intell. Soft Comput. Res. **9**(4), 247–266 (2019)