IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 17, 2022, accepted February 22, 2022, date of publication March 2, 2022, date of current version March 10, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3155873

Hybrid Restricted Boltzmann Machine–Convolutional Neural Network Model for Image Recognition

SZYMON SOBCZAK AND RAFAL KAPELA
Institute of Automatic Control and Robotics, Poznań University of Technology, 60-965 Poznań, Poland
Corresponding author: Rafal Kapela (rafal.kapela@put.poznan.pl)

This work was supported by the Poznań University of Technology, Institute of Automatic Control and Robotics, under Contract 0211/SBAD/0121.

ABSTRACT Convolutional Neural Networks (CNNs) have become a standard approach to many image processing dilemmas. Consequently, most of the proposed CNN architectures tend to increase the model deepness or layer complexity. Thus, they are composed of many parameters and need considerable computing resources and training examples. However, some recent works show that either shallow neural networks or architectures without convolutions can achieve similar results with these models often being used in systems with limited resources. Consideration of these aspects led us to a relatively simple preprocessing layer that increases the accuracy of CNN or may reduce its complexity. The layer is composed of two parts: the first is used to transform RGB data to binary representation, the second is a neural network that transforms the binary data into a multi-channel, real-value matrix and is trained in a fully unsupervised manner. Our proposal also includes a metric that may be used for measuring the similarity of training data, with the latter proving useful when performing transfer learning. Our experiments show that the resulting architecture not only helps to improve accuracy but is also more robust to image noise, including adversarial attacks, when compared to state-of-the-art models.

INDEX TERMS Artificial intelligence, binary patterns, deep learning, local binary pattern, image recognition, restricted Boltzmann machine.

I. INTRODUCTION

Due to the consolidation in artificial intelligence (AI) many problems are being approched from the deep learning perspective [1]. To date, there were plenty of deep models, architectures, training methods designed and implemented for better general solutions efficiency. Also, there has been many evaluations, surveys or reviews the existing state of the art models like [2]–[5] to just mention a few recent works. Having so many existing solutions there are even methods to compare the models between each other based on their behavioral responses [6]. The majority of these solutions are based on the supervised approach since, due to its deterministic nature, are easier to evaluate. On the other hand we have an extremely important, in many applications, parameter that stands partially for the complexity of the model – the number of its parameters. There is a clear trend that shows that more accurate solutions have more parameters and take more time to compute the output. Since our work aims to optimize the model complexity by lowering the number of parameters of the model and also introduces a hybrid architecture trained on both: supervised and unsupervised data we would like to present an overview of the trends in both areas with respect to the model size as an evaluation factor.

First layers in our hybrid model are trained in a fully unsupervised manner, which means we can use unlabelled data. Many methods can handle these kinds of problems and for models with categorical output, the solution most likely leads to clustering-based methods [7]. Otherwise, the most common tasks would be either data transformation [8], [9], regression [10] or any sort of structured predictions [11]. These are relatively simple techniques with low parameter count, proving useful in many scenarios. However, these methods have major disadvantages one of which is primarily the lack

The associate editor coordinating the review of this manuscript and approving it for publication was Jiachen Yang.

of simple measures describing how a model fits the data. Generally, this depends on the model hyper-parameters that have to be chosen before model fitting which is not simple to estimate. Moreover, there are no simple methods to measure how models generalize to the new data that have not been seen in the training process, especially for large inputs like images. Improved performance may be achieved with models that combine data transformation and clustering such as the neural network-based models called auto-encoders [12]. The way these models work is that they transform one space to another with lower dimensionality and they fit to auto-associative memory to recognize potential clusters [13].

A good example of the above-mentioned models is the Boltzmann Machine (BM) [14] and its simplified version – the Restricted Boltzmann Machine (RBM) [15] which has proven to be effective in many tasks such as feature extraction [16], dimensionality reduction [17], classification [18], or collaborative filtering [19]. An RBM is a generative model that can learn the probability distribution in the training dataset. This feature is extremely useful in many different tasks when a model can detect the most important features occurring in the image data [20]–[22]. It is fundamental in the research presented in this paper since the preprocessing we propose is meant to transfer the input image into high-dimensional features useful in semantic object recognition. The RBMs are characterized with rather low parameter count ranging in thousands which make them compact and efficient models. Another good examples of generic machine learning models trained in unsupervised manner can be observed in projects that use Variational Autoencoders (VAE) [23]. VAEs are used as a pre-trained first convolutional layers in the CNN model. There exist many variations of VAEs [24] that are useful, among others in image processing [25]. Although VAEs are much more complex models than RBMs where number of parameters range between 8 to 40 millions [26] making them quite complex to compute. They also require two-phase training for effective solutions [27]. To achieve accuracy comparable to VAE-based models RBMs have been stacked together to create Deep Believe Networks [28]. This also allows training RBM models with similar VAEs training techniques [29]. These methods improved RBM performance (and increased number of parameters) but not to the level of modern very deep CNNs [30]. This is due to their binary nature and complexity for high-dimensional data and leads to the second challenge taken in this paper which is data preprocessing for the binary input of RBMs.

The backbone of the hybrid model introduced in this paper is a classic convolutional network. Since in our experiments we used it to categorize images it is trained in the supervised manner. It is done by introducing the preprocessed images by the RBM layer to the input of the backbone. Due to the preprocessing technique the backbone can be limitted in number of parameters making the entire model less complex to compute. As it was mentioned there exist a number of models in the unsupervised training category that are featured with high accuracy together with comprehensive rewievs. However, the number of parameters in these models ranges in tens of millions [31] that makes them very complex and GPU dependant. In addition, newer models usually add complexity with respect to the previous solutions [32].

Our proposition for an efficient and relatively shallow network is the concatenation of the DBN with the input preprocessing unit with the classic convolution back-end. The natural choice is to use binary descriptors, moreover, the data transformed with binary descriptors contain more complex features of an image than its raw RGB form. Among many descriptors we decided to use a Local Binary Pattern with eight neighbors (LBP8) [33] because it performs best combined with the RBM layer [33]. In this paper though, we propose how to expand it to include the colorful nature of processed images. This model provides us with feedback on the data that can be used for assessment if the model has to be retrained for any unseen data. Moreover, it is also featured with high accuracy, similar to the deep [30] and shallow [34] counterparts. Due to the RBM layer the robustness to the noise is increased. This is true even for heavy adversarial noise which constitutes a serious problem in other state-of-the-art models [35]. Experimental results show that our model's accuracy can be higher by tens of percentage points in the case of adversarial attacks.

II. MATERIALS AND METHODS

A. COLOR LOCAL BINARY PATTERN

The LBP descriptor performs feature coding in gray-scale colorspace, hence the color information is lost. What we also lose is the intensity of the center pixel as this transformation relies only on comparisons of the intensity of the neighboring pixels. This may be sufficient for simple tasks and has been successfully used for example in face recognition [36], [37]. However, more complex classification problems require including color features as they can add important information. To compensate, the LBP8 descriptor was enhanced with another 8 bits that represent the color and intensity of the processed feature. The value of the additional binary feature is obtained from the center pixel in the currently processed blob. For each pair of colors: R - red, G - green, B - blue 2 bits are computed ($d = [b_0, b_1]$). For single colors pair - C_0 and C_1 we propose to compute the corresponding bits according to the following logic formulas:

$$\begin{aligned} b_0 &:= (C_1 > C_2), \\ b_1 &:= (C_2 > (C_1 + T)) \text{ or} \\ &\quad \times ((C_1 < (C_2 + T)) \text{ and } (C_1 > C_2)), \end{aligned} \tag{1}$$

where $T = \frac{2-\sqrt{2}}{2} MAX(C)$ is a threshold. This procedure equalizes the probability of each possible descriptor over all the color pairs.

Assuming the color pixel values are in range $[0; 255]$, T is equal to 73, the descriptor distribution can be visualized in Fig. 1.

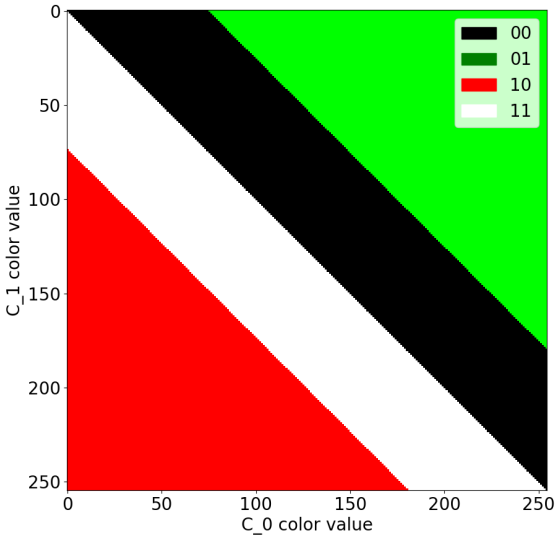


FIGURE 1. 2-bit descriptor distribution for single colors pair, the color in the image indicates the descriptor value according to the legend.

The three comparisons from *R-B*, *G-B*, *R-G* pairs add another 6 bits to the LBP8 descriptor. To enhance it with the intensity of the center pixel another 2 bits are added representing the intensity value in gray-scale compressed from 8 to 2 bits. The final color LBP descriptor (referred to further as CLBP) is composed by concatenating all the descriptors into a single 16-bit vector as presented in Fig. 2.

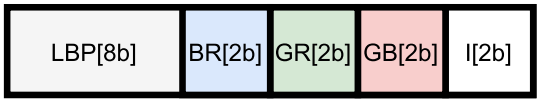


FIGURE 2. Color LBP descriptor.

It is important to note that since computing the CLBP descriptors for all pixels in the image is independent, it can be done at the same time which makes this feature extraction algorithm very fast, especially on GPU accelerated systems.

The LBP8 is computed in a standard way. For given part of image:

$$I(i,j) = \begin{bmatrix} p(i-1,j-1) & p(i-1,j) & (i-1,j+1) \\ p(i,j-1) & p(i,j) & p(i,j+1) \\ p(i+1,j-1) & p(i+1,j) & p(i+1,j+1) \end{bmatrix}, \tag{2}$$

where p is a single pixel in grayscale, a single bit of the descriptor is given by this formula:

$$b(k,l) = s(p(i,j) - p(k,l)), \tag{3}$$

where

$$s(x) = \begin{cases} 1, & if\ x > 0 \\ 0, & if\ x \leq 0 \end{cases} \tag{4}$$

and j, k pairs start at $(i-1, j-1)$ and are chosen sequentially clockwise, the LBP8 descriptor is a vector as follows:

$$\begin{aligned} LBP8(I) = \big[& b(i-1,j-1) \quad b(i-1,j) \quad b(i-1,j+1) \\ & b(i,j+1) \quad b(i+1,j+1) \\ & b(i+1,j) \quad b(i+1,j-1) \quad b(i,j-1) \big]. \end{aligned} \tag{5}$$

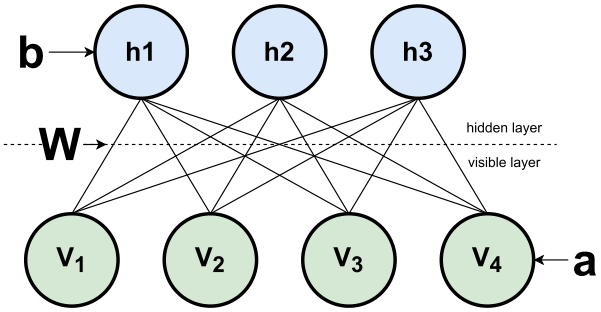


FIGURE 3. Diagram of a restricted boltzmann machine.

B. RBM AS BINARY PATTERNS PROCESSOR

The RBM can be presented as an undirected graph with input vector $\mathbf{v}$, hidden units $\mathbf{h}$, weights matrix $\mathbf{W}$, visible biases $\mathbf{a}$ and hidden biases $\mathbf{b}$ as presented in Fig. 3. The probability of hidden units activation can be computed as follows [22]:

$$\begin{aligned} P(h = 1|\mathbf{v}) &= \sigma(\mathbf{b} + \mathbf{W} \cdot \mathbf{v}), \\ P(v = 1|\mathbf{h}) &= \sigma(\mathbf{a} + \mathbf{W} \cdot \mathbf{h}), \end{aligned} \tag{6}$$

where σ is a sigmoid function.

In our pipeline, RBMs are used to transform the binary data obtained by using the CLBP descriptor into real values vectors. This data represents image features like edges, blobs, or simple shapes, that are usually recognizable by the first layers of the CNNs filters trained in a supervised way [38]. Usage of the RBMs permits skipping these layers which results in a smaller architecture to solve given image classification task. As a consequence, our preprocessing and the RBM layers allow us to process more complex features in the first convolutional layers.

Once the CLBP stage is done, the RBM processes the data. In the simplest form, the descriptor values are passed directly to the RBM so the number of visible units is 16, but the receptive fields can be larger which can be achieved by concatenating CLBP descriptors from a kernel of size K. The stride S can be also used to adjust the overlapping regions and the size of an output matrix. The dimension of the RBM input vector v is $n = K^2$. Fig. 4 presents how the input vector for the RBM is formed for $K = 2$. The input of the RBM layer is a matrix formed as follows:

$$RBM_{input} = \begin{bmatrix} v(1,1) & v(1,2) & \cdots & v(1,n) \\ v(2,1) & v(2,2) & \cdots & v(2,n) \\ \vdots & \vdots & \ddots & \vdots \\ v(n,1) & v(n,2) & \cdots & v(n,n) \end{bmatrix}, \tag{7}$$

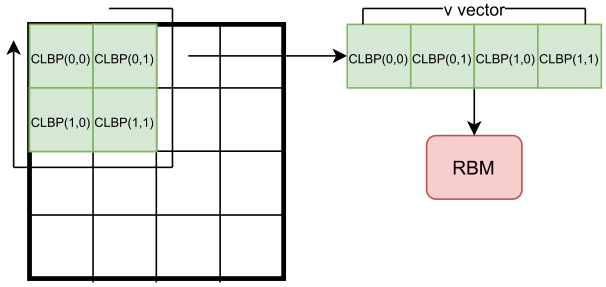


FIGURE 4. Preparing CLBP kernels.

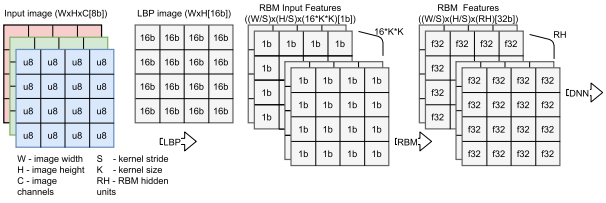


FIGURE 5. Preprocessing pipeline.

The RBM in this architecture becomes de facto a standard feed-forward neural network with the sigmoid activation function, so the value of h_j unit for the visible vector v is given by:

$$h_j = \sigma(b_j + \sum_i w_{ij} v_i). \tag{8}$$

So, the final equation for RBM layer in matrix notation is:

$$\sigma(B + W \cdot RBM_{input}). \tag{9}$$

The dimensionality of output data depends on the number of hidden units - RH and stride - S, so the matrix computed by the preprocessing pipeline is of size $[\frac{W}{S} \times \frac{H}{S} \times RH]$ where the dimensions of input image are $[W \times H]$. The CLBP-RBM transformation and the entire preprocessing pipeline are illustrated in Fig. 5.

C. DATASET COMPARISONS BASED ON RBM ENERGY

RBMs are energy-based models, which means that for every possible configuration of $\mathbf{v}$, $\mathbf{h}$ they assign a scalar value [39] (named usually as energy), defined as:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i w_{ij} h_j. \tag{10}$$

The energy is directly related to the probability of occurrence of given $\mathbf{v}$ and $\mathbf{h}$ configuration:

$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})}, \tag{11}$$

where Z is a partition function that sums $e^{-E(\mathbf{v},\mathbf{h})}$ over all possible $\mathbf{v}$,$\mathbf{h}$ configurations. The partition function is not tractable for high-dimensional RBMs and has to be estimated [40]. The marginal probability of given $\mathbf{v}$ is defined as a sum of the probabilities over all possible $\mathbf{h}$ configurations, but it may be computed with the use of the free energy value $F(v)$ [41] by using the following formula:

$$e^{-F(\mathbf{v})} = \sum_h e^{-E(\mathbf{v},\mathbf{h})}, \tag{12}$$

where the $F(\mathbf{v})$ is defined as:

$$F(\mathbf{v}) = -\sum_i v_i a_i - \sum_j log(1 + e^{x_j}), \tag{13}$$

and $x_j = b_j + \sum_i v_i w_{ij}$ is the total input to hidden unit j. Finally the marginal probability of a given $\mathbf{v}$ vector is defined by the following equation:

$$P(\mathbf{v}) = \frac{1}{Z} e^{-F(\mathbf{v})}. \tag{14}$$

By using this value in trained RBMs, we can compute how often a given $\mathbf{v}$ vector occurred in the training dataset. We propose to use this to check if previously trained RBMs can be used for transfer learning, if so, the weights in the preprocessing layer do not have to be optimized when the entire network is trained on new data. This is possible because if the datasets have a similar probability distribution over all the images the RBM should give similar responses in further CNN training. The obvious way to compare two datasets in our case would be to compare their histograms of binary pattern occurrences, but for high dimensionality patterns the histogram becomes very large because we cannot use binning. After transforming $\mathbf{v}$ vector into $P(\mathbf{v})$ value we can create a histogram for $P(\mathbf{v})$ that has a small number of bins. The histogram of the original data used for training has to be remembered as a reference when compared to the unknown data. For practical purposes, we recommend using the interquartile range (IQR) to get the most important part of the histogram, and then using Scott's rule [42] to optimize the number of bins.

Intervals for $P(V)$ to generate a histogram may be constant, given as $\frac{max(P(V))}{number\ of\ bins}$ or chosen automatically to flatten the reference histogram, then height of each bin is $\approx \frac{1}{number\ of\ bins}$. Flattening the histogram allow to avoid empty bins and large missbalance bettwen heights of particular bins, in this case the width of each bin is computed for reference histogram, then same widths are used to cumpute histograms for other datasets.

To measure the similarity of two datasets (T_1 and T_2) using their histograms(P and Q) with a scalar value d we can use the Chi-Squared (χ^2) distance [43] defined as:

$$d(T_1, T_2) = \frac{1}{2} \sum_i \frac{(P_i - Q_i)^2}{(P_i + Q_i)}, \tag{15}$$

where i denotes a particular histogram index.

D. NOISED PATTERN RECONSTRUCTION WITH RBMs

Since the RBMs are similar to Hoppfield network models, they may be used for data reconstruction [44]. The binary patterns processed by the network may be noised or corrupted. Especially in real-time systems, the data is being gathered directly from image sensors and there is no time for re-send procedures in the case of transmission failures.

For reconstruction, we propose using CLBP patterns. The RBM is a recurrent neural network, so the noised pattern may initialize the Markov Chain and after running several Gibbs steps, the data is closer to the patterns the RBM has been trained for. The simplified process of the reconstruction is presented in Fig. 6.

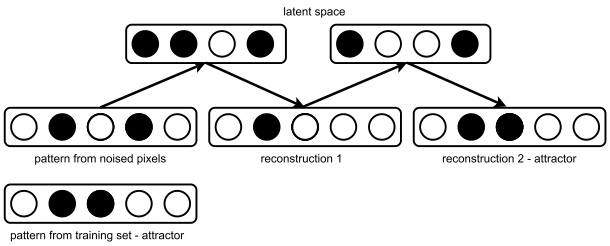


FIGURE 6. The simplified process of pattern reconstruction by RBM.

The algorithm that obtains the data for CNN for given v vector with reconstruction is as follows:

Algorithm 1 RBM Preprocessing Using Reconstruction

```
ph := update_ph_given_v(v)
step := 0
while step < number of steps do
    h := stochastic_bernoulli(ph)
    pv := update_pv_given_h(h)
    v := stochastic_bernoulli(pv)
    ph := update_ph_given_v(v)
    step := step + 1
end while
```

So the output is always a real-value vector as it is without reconstruction, but the intermediate values are binary. It is hard to predict how many Gibbs steps should be run to achieve the best reconstruction results, because in general, too few steps may not denoise the pattern correctly. On the other hand, too many steps may result in the RBM tending to meander in the atractor space. A detailed investigation of the number of Gibbs steps for our case is given later in the results section.

III. RESULTS

The previous section presents the idea of how Restricted Boltzmann Machines can be applied for input data preprocessing for convolutional neural networks. The main assumption is that the RBM is a relatively small and fast structure but its use may result in better performance of the overall image recognition pipeline. This section presents experiments that we performed to check this hypothesis. The experiments are split into 4 cases:

A) input data preprocessing in order to make use of unlabeled data,
B) input data preprocessing in order to reduce the size of convolutional neural network,
C) input data denoising for better recogniton ability,
D) images dataset comparison.

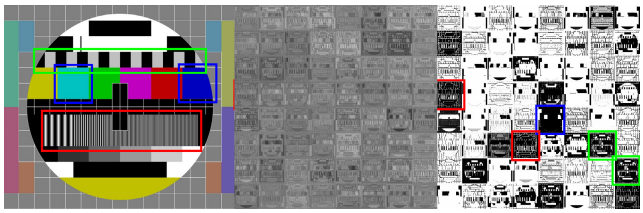


FIGURE 7. Image features taken from an RBM latent space, the left section of the image is the input data, the middle shows 64 features computed by 64 hidden units of an untrained RBM, the right section features an RBM trained over 30 epochs.

For the experiment A we trained RBM to show visualize how it can learn to images features without labeling input data then we compared how use of CLBP-RBM layer affects the recognition ability of convolutional network. For the experiment B we were reducing the number of convultional layers in CNN and testing how the accuracy decreases when the CLBP-RBM layer was used and when not. These tests showed that the preprocessing layer proposed by us improves the overall effectivness of CNN in terms of generaliztion ability and possible size reduction. For the experiment C we used two types of noise added to input data:

- random pixel noise,
- gradient distortion.

Then tested how the CNN network recognizes validataion data. For both types of noise the accuracy of the network decreases as the noise factor increases, but we showed that CLBP-RBM can significantly reduce the decrease in efficiency. The experiment D consisted of comparing the probability histograms computed by RBMs from different datasets, it showed that this type of measure can be used in order to check the similarity of input data.

A. TRAINING RBMs TO LEARN SIMPLE IMAGE FEATURES

The first step in investigating whether an RBM is able to recognize image features is to visualize its responses to hidden units. Single hidden units should react to particular features, like edges, corners, blobs, colors, etc. Visualization of the RBM filters is not useful in this case because they process the binary patterns that are not a part of the image directly. That is why we processed a testing image containing different features and colors and visualized the RBM responses in each hidden unit which is presented in Fig. 7. The middle part of Fig. 7 presents how the RBM processes the image when it has not been trained, the hidden units respond in a random way independently of the image features. The responses for trained RBMs in each channel differ depending on what particular unit is reactive to. For example, the regions marked with blue rectangles present the connection between the blue color in the image and the high response in filter number 37 (row 5, column 5), so this hidden unit is trained to recognize blue regions. Similar connections may be observed in regions marked with red and green rectangles but they are reactive on lines or blobs.

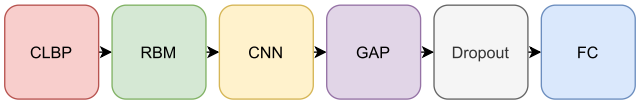


FIGURE 8. The CNN network with CLBP-RBM preprocessing.

B. CLBP-RBM PREPROCESSING VALIDATION

This section presents how a CLBP-RBM preprocessing layer affects the overall validation accuracy of CNN. To make use of unsupervised learning we utilized the STL-10 dataset [45] which is composed of three parts:

- 100000 unlabeled images for unsupervised learning,
- 5000 labeled images per category for supervised training,
- 8000 labeled images per category for testing.

For the CNN part of the network we chose 3 commonly used backbones:

- VGG16 [46],
- Resnet50 [30],
- DenseNet121 [47].

We also added a small backbone network composed of six convolutional layers followed by max pooling layers. It is referred to further as "our" network. For all the backbones after the convolutional layers, we added global average pooling [48] (GAP) and dropout [49] before the last fully connected layer (FC) as shown in Fig. 8. First step was to investigate best parameters for preproprocessing layer. For *kernelsize* and *numberofhiddenunits* we tested the accuracy of entire network and time od response in preprocessing layer. The visualization of results is shown in Fig. 9. All the results are relative to the lowest point (kernel size = 1, number of hidden units = 16). The best accuracy with the low response time is for kernel size = 2 and number of hidden units = 48. Therefore we use this set of parameters for further experiments.

We trained an RBM on the unlabeled data for 30 epochs, then the CNN and FC on training data for 150 epochs with cross-entropy loss function and RMSProp optimizer. Then compared the validation accuracy when CLBP-RBM preprocessing is used and when it is not. The measured value is a standard relative error given as:

$$\frac{accuracy\ CLB\_RBM - accuracy\ no\ CLBP\_RBM}{accuracy\ no\ CLBP\_RBM} \quad (16)$$

The results in table 1 illustrate that using the proposed preprocessing, the final neural network achieves higher validation accuracy, this was proven using four backbones, the accuracy metric for all of them was better when CLBP-RBM was used. The table also includes the size of the network and the time needed to process the image relative to CLBP-RBM time. This indicates that adding the preprocessing layer does not increase the overall processing time and that we can achieve better accuracy independently of the size of the CNN backbone. Fig. 10 presents learning curves on our backbone.

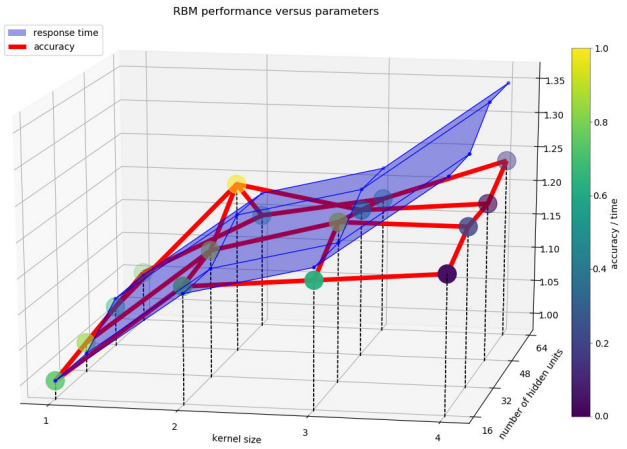


FIGURE 9. RBM performance versus hyperparameters.

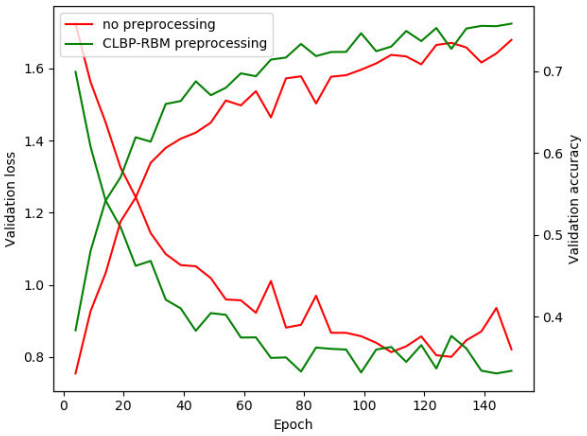


FIGURE 10. Validation learning curves with our backbone as CNN.

TABLE 1. Accuracy improvements achieved by different types of network backbones with CLBP-RBM preprocessing.

	backbone type			
	Our	VGG	Resnet	Densenet
Accuracy Improvement	2.6%	2.9%	4.7%	7.5%
Number of parameters	2.76M	16M	27.7M	7M
Processing time - CLBP-RBM relative	15x	22x	135x	300x

The network with CLBP-RBM preprocessing outperforms the version without it.

Based on the assumption that CLBP-RBM preprocessing enriches the input features for CNNs, we tested how the reduction of the number of convolutional layers affects the overall validation accuracy. The table 2 presents the results achieved by the network relative to "our" network with six convolutional layers. For a CNN network without preprocessing, the deletion part of convolutional layers affects the accuracy metric more significantly than for the network using our preprocessing, so our approach may lead to the possibility of using smaller networks performing the same

TABLE 2. Accuracy reduction versus CNN layers reduction.

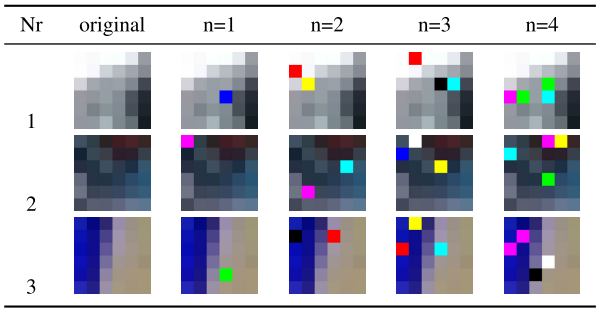
	no. convolutional layers reduction			
	1	2	3	4
CLBP-RBM	**2.50%**	**5.00%**	**19.00%**	**25.00%**
no preprocessing	4.10%	10.00%	25.00%	37.00%

tasks with higher accuracy. For example, the deletion of two convolutional layers reduced the accuracy by 5% while, when no preprocessing was used, the reduction was 10%, therefore designing the CNN network for resource-limited systems when the trade-off between size and accuracy of the network is significant may be simpler with the proposed preprocessing.

C. PATTERN RECONSTRUCTION

As mentioned in the previous section RBMs may be used to reconstruct noised data and we use this ability with binary patterns. Since the LBP transformation is not linear, we cannot inverse it to visualize the reconstructed patterns, but we can measure the Hamming distance between the CLBP descriptor taken from the original and reconstructed pattern and compare the distance between the CLBP from the original and from the noised pattern. In order to investigate the reconstruction ability. We trained RBMs on input vectors obtained from STL-10 dataset, since the time of response is not critical for this experiment we used kernel size = 4, which gives 256 visible units in an RBM. Then we chose random patterns from the dataset and added rndom noise in n pixels. Some examples are shown in the table 3.

TABLE 3. Example patterns used for reconstruction, the original and with added random noise.



Using this noising method we defined a metric to measure the reconstruction ability:

$$D(CLBP(OP),\ CLBP(NP)) - D(CLBP(OP),\ R), \quad (17)$$

where D is a Hamming distance [50], OP is the original pattern, NP is a noised pattern, R is a reconstructed pattern. Hamming distance is a natural method to compare binary vectors, thus this metric allows to observe the general improvement of reconstruction, the higher value the better reconstruction. The comparison was carried out 100 times on 50 randomly chosen patterns from a training dataset, then we averaged the value for each noise and each number of Gibbs steps. Results in table 4 show that the reconstruction efficiency increases

TABLE 4. Reconstruction results.

	noised pixels			
Gibbs Steps	1	2	3	4
1	**1.24**	**7.46**	**13.6**	**19.2**
2	-0.7	5.5	11.23	16.713
3	-1.77	4.21	10.04	15.44
4	-2.47	3.38	9.28	14.96

with an increase in noise. The ideal number of Gibbs steps for reconstruction is 1 because this results in the most significant improvement compared to other numbers of steps and needs the least computing.

Since the previous experiments seem to show that a CLBP-RBM layer may help in recognition of noised images, we added random noise to the images being classified in a validation set in STL-10. Noise was added randomly in each validation stage with three noise factor values. The noise factor is the proportion of noised pixels in the image. Examples of noised samples are shown in Fig. 11. The accuracy and loss over the epochs are presented in Fig. 13 and Fig. 12a.

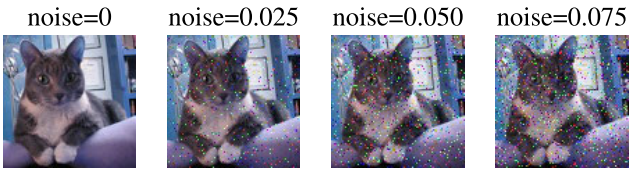


FIGURE 11. Examples of noised images with different noise factors.

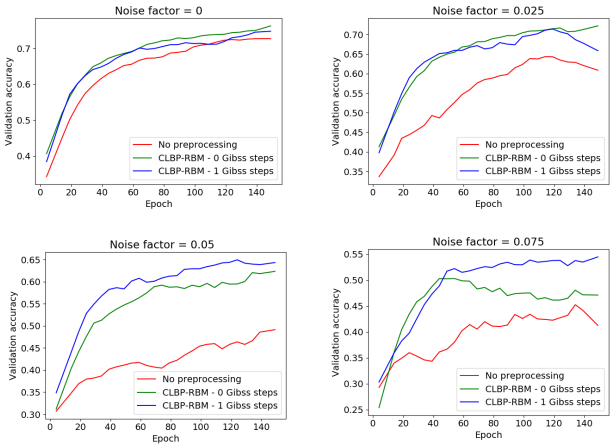


FIGURE 12. Validation accuracy curves for different noise factors.

The results show the impact of the noise effects on the generalization ability of the entire network. The accuracy decreases as the noise factor increases, but the influence of a CLB-RBM layer is also significant. We can observe that the CLB-RBM layer improves the accuracy independent of the noise factor. For the highest noise factor = 0.75, the accuracy was 18% higher with RBM layer than without.

Another type of perturbation of an input image is the fast gradient sign method [51] which relies on noising the image based on the gradient computed by the network. The formula

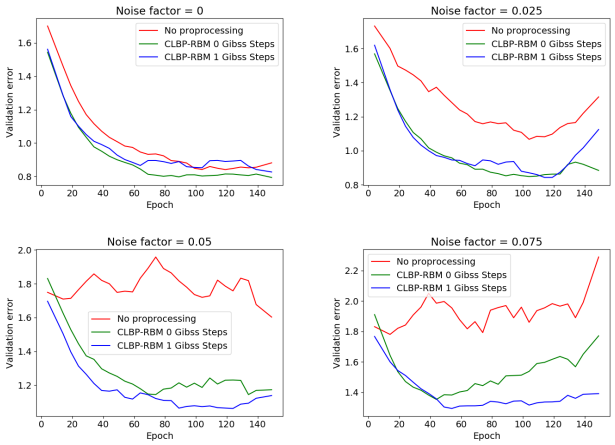


FIGURE 13. Validation loss curves for different noise factors.

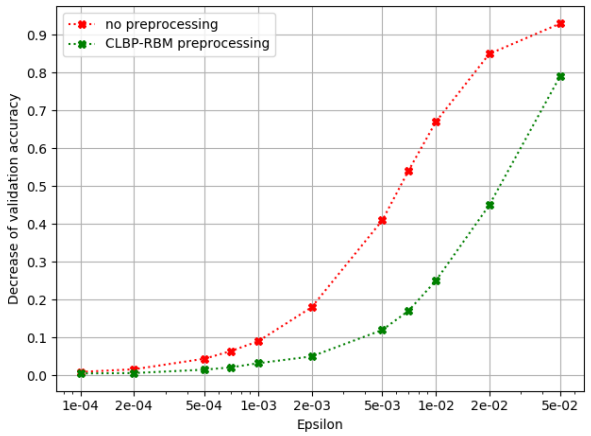


FIGURE 14. Decrease of accuracy on validation dataset versus epsilon in adversarial attacks.

for obtaining the distorted image is as follows:

$$x = x + \epsilon(\nabla_x J(\theta, x, y)) \tag{18}$$

where x is an input image assuming the value of pixels are in range [0, 1], J is a loss function, y is an output label, ϵ is noise factor.

In this case, the distortion is performed on an already preprocessed image (i.e., after the CLBP-RBM layer), however, an RBM may reconstruct the data with regard to the algorithm 1 assuming v is the noised input. We tested how the fast gradient method affects the generalization ability in the network with CLB-RBM preprocessing compared to a network without. The results are shown in Fig. 14. This distortion affects both networks, but for the network that uses CLB-RBM preprocessing the decrease of accuracy is significantly smaller. That demonstrates that 'our' method may also be applied for this use case.

D. RBM ENERGY FOR MEASURING THE SIMILARITY OF TWO DATASETS

For measuring the similarity of data from different datasets, we used the previously mentioned STL-10, and two others:

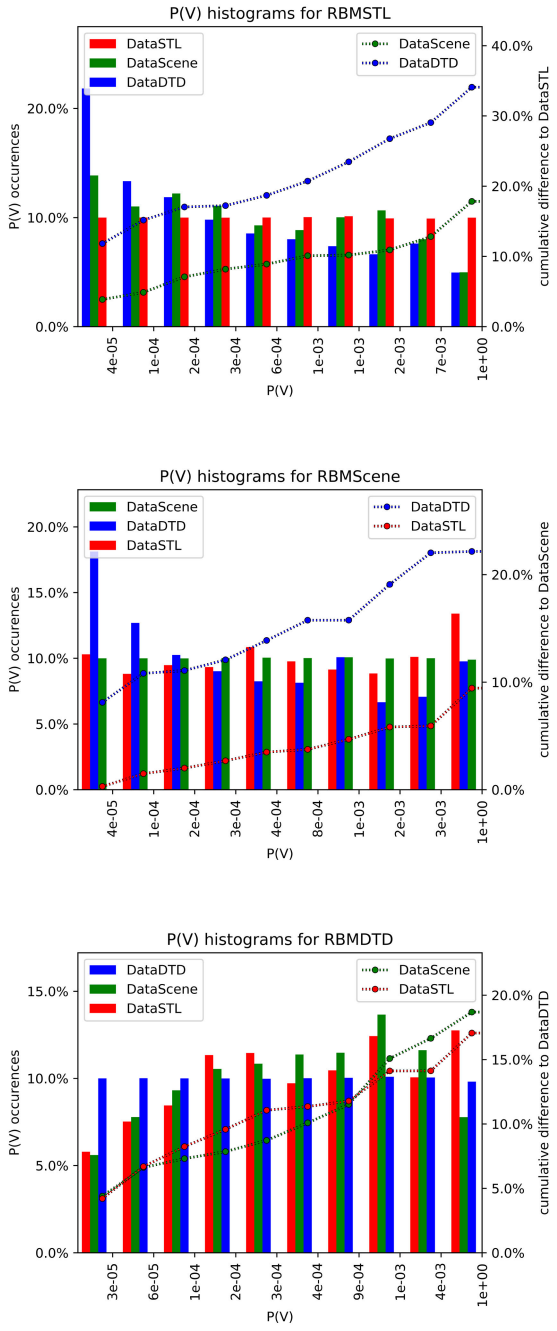


FIGURE 15. Histograms for $P(V)$ taken from different datasets.

- DTD [52] - composed of images presenting different textures,
- Indoor Scene [53] - composed of images from different room scenes.

For humans, the DTD images differ from the other two while the Indoor Scene and STL-10 seem to be similar, although they present different objects they are taken from the real world as pictures, so the features, in general, should be similar.

Table 5 includes the distances between those datasets.

TABLE 5. Distances to datasets based on RBM probabilities histogram.

RBM \ Data	STL	Scene	DTD
STL	X	29	108
Scene	18	X	60
DTD	50	54	X

The first two rows in Table 5 show the distances for RBMs trained on STL-10 and Scene dataset. For these two rows, the distance to the DTD dataset is much higher than the distances to other rows. Furthermore, for an RBM trained on the DTD dataset, the distances to Scene and STL-10 datasets are similar but significantly higher than it was for the other two RBMs. This implies that the RBMs should not be transferred between Scene or STL-10 datasets to the DTD dataset and vice versa.

Fig. 15 presents the histograms for all trained RBMs and different datasets. For visualization purposes the number of bins is 10, no IQR is used and the histograms are flattened. Despite the comparisons are simplified, viusalizations lead to the similar conclusions as we had for Table 5.

IV. CONCLUSION

This paper aims to describe and investigate the potential of Restricted Boltzmann Machines as Local Binary Pattern processors in terms of a preprocessing tool for Convolutional Neural Networks. First, we introduced the improved Local Binary Pattern, which enhances the original LBP by an additional 8 bits that describe the color and simplified intensity of processed pixels, in further tests, we showed that an RBM is capable of recognizing these additional features. Fig. 7 shows an example of how hidden units are trained to recognize the basic image features. These are then efficient representations for the CNN to classify complex objects. The enhanced LBP is an efficient solution to provide additional information about image features.

The primary aim of this project was to achieve an improved generalization ability with the use of an RBM as a preprocessor for the CNN by using an unlabeled datset and unsupervised learning. We addressed that with a CLBP-RBM layer, which can be trained in a fully unsupervised manner. This layer used in image preprocessing increased overall accuracy for the entire model on the validation dataset by 2.6%-7.5%. We tested several commonly used CNN backbones plus one small custom backbone created by us. They differ in the number of convolutional layers and the number of parameters. This implies that a CLBP-RBM layer is a good choice for any CNN backbone architecture. The time measurements show that the preprocessing layer does not greatly affect the overall processing time. Another test revealed that by using a CLBP-RBM we can reduce the number of convolutional layers without a significant decrease in accuracy. This feature can be extremely useful for systems with limited resources. An additional advantage of the preprocessing layer proposed by us is its potential use in denoising of input images. We performed a range of tests with recognition of noised images. The results of these tests lead to the conclusion that the CLBP-RBM layer improves image recognition quality. Thanks to their recurrent structure and reconstruction capability, they may also denoise the corrupted patterns by running a number of Gibbs steps. This proved useful in corrupted images when the noise factor was very high. Furthermore, the proposed denoising method was effective in adversarial attack problems, we showed that our method achieves significantly better accuracy in this type of perturbation than a network without any preprocessing.

We were also challenged with a problem to measure the fit of trained RBMs in transfer learning tasks. This is achievable since the free energy of an RBM can be used to compute the marginal probability of a given input vector. This way we can create a histogram of these probabilities taken from a subset of data and then measure the distance to the reference histogram that was created during training the RBM on the original data. In our opinion, this is an essential feature of the preprocessing step since it introduces the ''awareness'' of the model of the input data. In other words, the model can estimate if it is fitted well to the input data or has to be retrained. Most of the images used in training for the presented models depict natural objects, therefore they have a similar binary pattern distribution. This way, if the preprocessing model is fitted well to the data, time can be saved in the retraining procedure.

Since we have shown that a CLBP-RBM preprocessing layer is applicable in classification tasks, we believe that it may be useful in other image processing problems, such as object detection or segmentation because the first layers in CNNs for those rely on similar feature extraction methods. Therefore, there are potentially many other areas in image processing where this preprocessing may be successfully applied.

REFERENCES

[1] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, ''Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions,'' *J. Big Data*, vol. 8, no. 1, Dec. 2021.

[2] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, ''A survey on deep learning for big data,'' *Inf. Fusion*, vol. 42, pp. 146–157, Jul. 2018.

[3] L. Liu, O. Wang, X. Wang, P. Fieguth, J. Chen, and X. Liu, ''Deep learning for generic object detection: A survey,'' *Int. J. Comput. Vis.*, vol. 128, pp. 261–318, Oct. 2018.

[4] S. Herath, M. Harandi, and F. Porikli, ''Going deeper into action recognition: A survey,'' *Image Vis. Comput.*, vol. 60, pp. 4–21, Apr. 2017.

[5] S. Wang, W. Zhou, and C. Jiang, ''A survey of word embeddings based on deep learning,'' *Computing*, vol. 102, no. 3, pp. 717–740, Mar. 2020.

[6] Y. Li, Z. Zhang, B. Liu, Z. Yang, and Y. Liu, ''ModelDiff: Testing-based DNN similarity comparison for model reuse detection,'' in *Proc. 30th ACM SIGSOFT Int. Symp. Softw. Test. Anal.*, Jul. 2021, pp. 139–151.

[7] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Upper Saddle River, NJ, USA: Prentice-Hall, 1988.

[8] K. Pearson, ''On lines and planes of closest fit to systems of points in space,'' *London, Edinburgh, Dublin Phil. Mag. J. Sci.*, vol. 2, no. 11, pp. 559–572, 1901.

[9] G. H. Golub and C. F. Van Loan, *Matrix Computation*. Baltimore, MD, USA: Johns Hopkins Univ. Press, 1996.

[10] G. K. Uyanik and N. Guler, ''A study on multiple linear regression analysis,'' *Proc. Social Behav. Sci.*, vol. 106, no. 9, pp. 234–240, 2013.

[11] G. BakIr, B. Taskar, T. Hofmann, B. Scholkopf, A. Smola, and S. Vishwanathan, *Predicting Structured Data*. Cambridge, MA, USA: MIT Press, 2007.

[12] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Netw.*, vol. 2, no. 1, pp. 53–58, Jan. 1989.
[13] J. J. Hopfield and J. J. Tank, "Computing with neural circuits: A model," *Science*, vol. 233, pp. 624–633, Aug. 1986.
[14] D. Ackley, G. Hinton, and T. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognit. Sci.*, vol. 9, pp. 147–169, Jan. 1985.
[15] P. Smolensky, *Information Processing in Dynamical Systems: Foundations of Harmony Theory*. Cambridge, MA, USA: MIT Press, 1986.
[16] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.
[17] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.
[18] S. Nie, Z. Wang, and Q. Ji, "A generative restricted Boltzmann machine based method for high-dimensional motion data modeling," *Comput. Vis. Image Understand.*, vol. 136, pp. 14–22, Jul. 2015.
[19] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted Boltzmann machines for collaborative filtering," in *Proc. 24th Int. Conf. Mach. Learn. (ICML)*, 2007, pp. 791–798.
[20] K. Shimagaki and M. Weigt, "Selection of sequence motifs and generative hopfield-potts models for protein families," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 100, no. 3, Sep. 2019.
[21] A. Decelle, S. Hwang, J. Rocchi, and D. Tantari, "Inverse problems for structured datasets using parallel TAP equations and restricted Boltzmann machines," *Sci. Rep.*, vol. 11, no. 1, pp. 1–13, Dec. 2021.
[22] C. A. S. Assis, A. C. M. Pereira, E. G. Carrano, R. Ramos, and W. Dias, "Restricted Boltzmann machines for the prediction of trends in financial time series," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2018, pp. 1–8.
[23] P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. ICLR*, 2014, pp. 1–14.
[24] L. Girin, S. Leglaive, X. Bie, J. Diard, T. Hueber, and X. Alameda-Pineda, "Dynamical variational autoencoders: A comprehensive review," *Found. Trends Mach. Learn.*, vol. 15, nos. 1–2, pp. 1–175, 2021.
[25] K. He, X. Chen, S. Xie, Y. Li, P. Dollar, and R. Girshick, "Masked autoencoders are scalable vision learners," 2021, *arXiv:2111.06377*.
[26] A. Asperti, D. Evangelista, and E. L. Piccolomini, "A survey on variational autoencoders from a green AI perspective," *Social Netw. Comput. Sci.*, vol. 2, no. 4, pp. 1–23, Jul. 2021.
[27] M. A. Ranzato, C. Poultney, S. Chopra, and Y. Lecun, "Efficient learning of sparse representations with an energy-based model," in *Proc. Adv. Neural Inf. Process. Syst. J.*, 2006, pp. 1–8.
[28] G. Hinton and R. Salakhutdinov, "Deep Boltzmann machines," in *Proc. 12th Int. Conf. Artif. Intell. Statist.*, 2009, pp. 449–455.
[29] R. Salakhutdinov and H. Larochelle, "Efficient learning of deep Boltzmann machines," *J. Mach. Learn. Res. Proc. Track*, vol. 9, pp. 693–700, Mar. 2010.
[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[31] M. Capra, B. Bussolino, A. Marchisio, M. Shafique, G. Masera, and M. Martina, "An updated survey of efficient hardware architectures for accelerating deep convolutional neural networks," *Future Internet*, vol. 12, no. 7, p. 113, Jul. 2020.
[32] X. Hu, L. Chu, J. Pei, W. Liu, and J. Bian, "Model complexity of deep learning: A survey," *Knowl. Inf. Syst.*, vol. 63, no. 10, pp. 2585–2619, Oct. 2021.
[33] S. Sobczak, R. Kapela, K. McGuinness, A. Swietlicka, D. Pazderski, and N. E. O'Connor, "Restricted Boltzmann machine as an aggregation technique for binary descriptors," *Vis. Comput.*, vol. 37, no. 3, pp. 423–432, Mar. 2021.
[34] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy, "MLP-mixer: An all-MLP architecture for vision," 2021, *arXiv:2105.01601*.
[35] A. Shaeiri, R. Nobahari, and M. H. Rohban, "Towards deep learning models resistant to large perturbations," 2020, *arXiv:2003.13370*.
[36] R. Abdur, H. Najmul, W. Tanzillah, and A. Shafiul, "Face recognition using local binary patterns (LBP)," *Global J. Comput. Sci. Technol. Graph. Vis.*, vol. 13, no. 4, pp. 1–9, May 2013.
[37] S. Moore and R. Bowden, "Local binary patterns for multi-view facial expression recognition," *Comput. Vis. Image Understand.*, vol. 115, no. 4, pp. 541–558, 2011.
[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.
[39] A. Decelle and C. Furtlehner, "Restricted Boltzmann machine, recent advances and mean-field theory," 2020, *arXiv:2011.11307*.
[40] O. Krause, A. Fisher, and C. Igel, "Algorithms for estimating the partition function of restricted Boltzmann machines," *Artif. Intell.*, vol. 278, Jan. 2019, Art. no. 103195.
[41] G. Hinton, "A Practical guide to training restricted Boltzmann machines," Tech. Rep. UTML TR 2010-003, Univ. Toronto, Toronto, ON, Canada, 2010.
[42] D. W. Scott, "On optimal and data-based histograms," *Biometrika*, vol. 66, no. 3, pp. 605–610, 1979.
[43] O. Pele and M. Werman, "The quadratic-chi histogram distance family," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 749–762.
[44] V. Srinivasan, Y. Han, and S. Ong, "Image reconstruction by a Hofield neural network," *Image Vis. Comput.*, 1993.
[45] A. Coates, H. Lee, and Y. Andrew, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. AISTATS*, 2011, pp. 215–223.
[46] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Proc. Int. Conf. Learn. Represent.*, 2014, pp. 1–14.
[47] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.
[48] M. Lin, Q. Chen, and S. Yan, "Network in network," 2014, *arXiv:1312.4400*.
[49] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2012.
[50] R. W. Hamming, "Error detecting and error correcting codes," *Bell Syst. Tech. J.*, vol. 29, no. 2, pp. 147–160, Apr. 1950.
[51] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.
[52] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 3606–3613.
[53] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 413–420.
[54] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.

SZYMON SOBCZAK received the Master of Science degree in automatic control and robotics from the Poznań University of Technology, in 2016, after writing his thesis on the generalization abilities of deep neural networks. He currently works in industry as a Software Engineer and an Artificial Intelligence Researcher focusing mainly on video and audio processing and automatic event detection. His research interests include deep neural networks and unsupervised learning, particularly in terms of optimization for resources-limited systems. He is interested mainly in the application of machine learning in video processing and vision systems.

RAFAL KAPELA received the M.S. and Ph.D. degrees in control and robotics from the Poznań University of Technology, Poland, in 2000 and 2008, respectively. In August 2009, he had the opportunity to join a new team of quality assurance engineers at Mentor Graphics, Poznan. His role there included quality assurance of software intended for design for test purposes. He was working simultaneously at both PUT and MGC, until May 2011. At that point, he became an FP7 Intra-European Fellowship Marie Curie Fellow for the VISION Project which was conducted in CLARITY at Dublin City University, Ireland. Following this, he returned to Poland, where he continues his research on image processing algorithms and FPGA systems. His research interests include multimedia, video annotation and compression, signal processing systems, and artificial intelligence.