DOI: https://doi.org/10.15276/hait.03.2021.3
UDC 004.93.1

Methodology of neural network compression for multi-sensor transducer network models based on edge computing principles

Ivan M. Lobachev[1)]
ORCID: https://orcid.org/0000-0002-4859-304X; lobachev.i.m@gmail.com. Scopus ID: 57192379296
Svitlana G. Antoshchuk[1)]
ORCID: https://orcid.org/0000-0002-9346-145X; asg@opu.ua. Scopus ID: 8393582500
Mykola A. Hodovychenko[1)]
ORCID: https://orcid.org/0000-0001-5422-3048; nick.godov@gmail.com. Scopus ID: 57188700773
[1)] Odessa National Polytechnic University, 1, Shevchenko Ave. Odessa, 65044, Ukraine

ABSTRACT

This paper focuses on the development of a methodology to compress neural networks that is based on the mechanism of pruning the hidden layer neurons. The aforementioned neural networks are created in order to process the data generated by numerous sensors present in a transducer network that would be employed in a smart building. The proposed methodology implements a single approach for the compression of both Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) that are used for the tasks of classification and regression. The main principle behind this method is based on the dropout mechanism, which is employed as a regulation mechanism for the neural networks. The idea behind the method proposed consists of selecting optimal exclusion probability of a hidden layer neuron, based on the redundancy of the said neuron. The novelty of this method is the usage of a custom compression network that is based on an RNN, which allows us to determine the redundancy parameter not just in a single hidden layer, but across several layers. The additional novelty aspect consists of an iterative optimization of the network-optimizer, to have continuous improvement of the redundancy parameter calculator of the input network. For the experimental evaluation of the proposed methodology, the task of image recognition with a low-resolution camera was chosen, the CIFAR10 dataset was used to emulate the scenario. The VGGNet Convolutional Neural Network, that contains convolutional and fully connected layers, was used as the network under test for the purposes of this experiment. The following two methods were taken as the analogous state of the art, the MagBase method, which is based on the sparcification principle as well as the method which is based on rarefied representation by employing the approach of rarefied encoding SFAC. The results of the experiment demonstrated that the amount of parameters in the compressed model is only 2.56 % of the original input model. This has allowed us to reduce the logical output time by 93.7 % and energy consumption by 94.8 %. The proposed method allows to effectively using deep neural networks in transducer networks that utilize the architecture of edge computing. This in turn allows the system to process the data in real time, reduce the energy consumption and logical output time as well as lower the memory and storage requirements of real-world applications.

Keywords: Smart Building; Internet of Things; Neural Network Compression; Network pruning; Sparse Representation; Recurrent Neural Network; Long Short-Term Memory

***For citation*: Lobachev I. M., Antoshchuk S. G., Hodovychenko M. A. Methodology of neural network compression for multi-sensor transducer network models based on edge computing principles. *Herald of Advanced Information Technology.* 2021; Vol. 4 No. 3: 232–243. DOI: https://doi.org/10.15276/hait.03.2021.3**

INTRODUCTION, FORMULATION OF THE PROBLEM

In recent years we have seen a wide application of machine learning principles, especially in Internet of Things (IoT) systems, this is explained in part by the availability of small, inexpensive computational devices [1].

Approaches that are based on traditional methods of machine learning, require manual attribute selection, this lowers their ability to generalize on new data sets that do not contain information about the distribution of the data [2]. As a result, the implementations based on deep neural networks became the dominating type in many areas, including smart buildings, agriculture, motion and gesture recognition etc. [3, 4]. The main reason behind this is that neural networks allow to automatically select attributes from a large volume or raw data.

This allows the system to exclude the human factor and obtain a high degree of solution efficiency in the neural network.

Among the most popular types of neural networks, one could highlight Convolutional Neural Networks (CNN), Fully Connected Neural Networks (FC), and Recurrent Neural Networks (RNN). Models based on neural networks can include one or more types of neural networks in varying combinations. Convolutional neural networks extract spatial attributes, while recurrent neural networks are geared more towards time attribute extraction from the data set. The usage of these attributes is the key

to the high efficiency of neural networks when dealing with tasks such as classification, regression and forecasting [5, 6].

Despite these advantages, the usage of models based on machine learning requires a large pool of resources, including energy, processing capability and data storage.

These requirements lower the feasibility of employment of neural networks in devices with limited resources [7].

Usually, extensive resource requirements become the bottleneck when developing IoT solutions, where the output of the neural network is needed in real-time.

One of the methodologies to decrease resource demand of neural networks is the compression of neural networks.

Compressed neural networks possess several advantages when compared to traditional neural networks:

– computational power demand: a vast number of floating-point operations (FLOPs) involved in the operation of DNN (Deep Neural Network) can exceed the limited processing capacity of the IoT nodes [11, 12], [13, 14].

Therefore, it would be useful to use DNN compression to reduce processing power requirements [15]:

– time consumption: the training and working time of a DNN model is significantly long, which makes it difficult to perform model inference in real time [16, 17]. DNN compression techniques thus provide a higher degree of quickness in both training and inference tasks;

– memory capacity: neural networks achieve a high performance when using large number of neurons, which in turn requires large memory consumption to hold and process the model [8, 9], [10]. As a result, compression could lower the memory requirements. This simplifies and makes it more economical to deploy compressed neural networks on devices with limited memory resources;

– power consumption: compression of neural network also lowers the energy consumption for data processing [19, 20], [21]. This improves the ease of deploying the compressed neural network models on battery-power IoT devices;

– privacy: the transfer of data from an edge node to the cloud leads to high probability of security breaches and confidentiality compromises [18]. Therefore, it would be beneficial to process data with neural networks in-place, which helps maintain confidentiality and ensures data security.

A feature of IoT systems that are used in the field of creating smart buildings is the use of a variety of sensors of various types, with different dimensions of data as well as update frequency for new data, which consider the spatial and temporal characteristics of their environment.

Thus, **the purpose of this study** is to develop a compression method that would provide a unified approach to the compression of various types of neural networks to process multisensory data in the development of smart buildings.

1. LITERATURE REVIEW

Compression techniques of deep neural networks can be broken down into five types, depending on the approach to the compression process used: network pruning, sparse representation, data precision, knowledge distillation, and other approaches. Table 1 lists the main categories and subcategories of compression techniques.

Network pruning. This compression approach is implemented by removing individual components of neural network: filters, channels, layers of individual neurons in order to get compressed model.

Compressed model uses less memory, consumes less energy and allows to get inference faster than uncompressed model with the same accuracy or with acceptable loss of accuracy.

To measure the importance and contribution of neural network component to the final network performance, we could compare network accuracy when component is removed from the model [17]. Pruning is applied step by step to the neural network to exclude only components that will not or minimally decrease network performance.

Pruning techniques can be further divided into four subcategories, based on components to be pruned: channel pruning, filter pruning, layer pruning and connection pruning.

Pruning methods could minimize the amount of used storage and requirements for the computing power of the node on which the compressed neural network is deployed.

Sparse representation. Sparse representation uses the sparsity which is present in the weight matrices of the neural network model. In sparse representation techniques, the weights there are zero or close to zero are excluded from the matrix, which lowers the power and computational power requirements of the compressed model. Connections in the network with similar weights are multiplexed, where in-place of multiple weights with a single connection is replaced with a single weight with the multiplex connection.

Sparse representation techniques could be divided into quantization, multiplexing, and weight sharing. The main idea behind the sparse representa-

tion methods is to reduce the weight matrix without losing the performance of the DNN model.

Data precision. Data precision techniques reduces the number of bits required for storing the weights in the weight matrices W. For example, the FLOPs in the default DNN model requires 32 bits, which can be replaced with integer datatype, that requires only 8 bits. Similarly, we can use binary, 6 bits, 16 bits for replacing 32 bits FLOPs in the DNN model. We categorized the existing literature on DNN compression using bits precision into three sub-categories, namely, estimation using integer, low bits representation, and binarization.

Table 1. **Overview of compression techniques categories**

Compression techniques	Subcategories
Network pruning	
	Layer pruning [24,25]
Sparse representation	
	Weight sharing [29]
Data precision	
	Low bit representation [32]
Knowledge distillation	
	Domain adaptation [35]
Other methods	[36,37], [38]

Source: compiled by the authors

Knowledge distillation. In a DNN model, the term knowledge distillation is defined as the process of transferring the generalization ability of the complex model (teacher) to the compact model (student) to improve its performance. Knowledge distillation provides a mechanism to overcome the accuracy tradeoff due to the DNN compression. The training of the student model using knowledge distillation improves its generalizability so that it can mimic teacher-like behavior to predict the probabilities of a class label.

We could further divide knowledge distillation techniques into logit transfer techniques, teacher-assistant techniques, and domain adaptation methods.

Other methods. DNN compression methods that do not fit into any of the above four categories are classified as other. These include DNN compression techniques that perform DNN modeling in such a way that they can be easily deployed on mobile devices. Typically, these methods consist of allocating tasks to reduce memory usage or leverage parallel processing mechanisms.

Analysis of neural network compression methods has shown the relevance of developing a method that provides a unified approach to the compression of machine learning models that use a combination of different types of neural networks to process multisensory data in the field of smart building systems.

2. COMPRESSION METHOD DESIGN

The proposed method is based on the use of the well-known deep neural network regularization method known as "dropout" or exclusion. The dropout operation assigns to each neuron of the network of hidden layers the probability of its exclusion. During the screening process, elements of hidden layers can be excluded based on a probability parameter, resulting in a network structure with fewer elements. The main parameter of this operation is the probability of exclusion, which must be selected in such a way as to form an optimal network structure that preserves the accuracy of work and minimizes resource consumption. The proposed method is aimed at finding the optimal exclusion probability for each element of the hidden layer.

To obtain the parameter of the optimal probability of exclusion in the proposed method, the network parameters themselves are used. For this, the parameter of redundancy of the node of the hidden layer of the neural network is introduced. From the point of view of the compression process of the model, the element with higher redundancy has a higher probability of being excluded.

A new idea within this method is the usage of a special neural network, which plays the role of an optimizer (network-optimizer), which takes as input the weights of each layer of the original network, finds the redundancy value, and estimates the probability of dropout for each neuron of the hidden layer.

Within the compression process, the original neural network and network-optimizer are enhanced iteratively to reduce the value of loss function.

The proposed technique could be described as a sequence of steps:

1) dropout operations are added to the hidden layers of the source neural network, which randomly eliminate nodes with probability $p^{(l)}$. The input and output layers of the neural network has a fixed size and do not affect by compression method;

2) initialization of network-optimizer. Each layer intended for compression represented as the weight matrix $W^{(l)}$, from which the network-optimizer receives the values of the redundancy parameter, after which the optimal probabilities of dropout each element of the hidden layer $p^{(l)}$, which

are then used in the dropout operation in the original neural network;

3) an iterative process of enhancing of the network-optimizer and the source network conducted. The network-optimizer is enhanced to get more accurate dropout probabilities, which will produce a more accurate compressed structure of the original neural network. The original neural network is enhanced to achieve a more efficient structure.

2.1. Dropout routine

The basic idea is that we regard neural networks with dropout operations as Bayesian neural networks with Bernoulli variational distributions.

For the full connected layers, the dropout operation can be described as

$$\begin{aligned} z_{[j]}^{(l)} &\sim Bernoulli\left(p_{[j]}^{(l)}\right), \\ \widetilde{W}^{(l)} &= W^{(l)} diag\left(z^{(l)}\right), \\ Y^{(l)} &= X^{(l)}\widetilde{W}^{(l)} + b^{(l)}, \\ X^{(l+1)} &= f\left(Y^{(l)}\right), \end{aligned} \tag{1}$$

where: $l = 1, \dots, L$ – is the layer number in the network; $W^{(l)} \in R^{d^{(l-1)} \times d^{(l)}}$ – weight matrix for each layer l; $b^{(l)} \in R^{d^{(l)}}$ – bias vector; $X^{(l)} \in R^{1 \times d^{(l-1)}}$ – the input; $f(\cdot)$ – activation function.

As shown in (1), each hidden unit is controlled by a Bernoulli variable. In the default dropout method, the success probabilities of $p_{[j]}^{(l)}$ usually is assigned to the same constant p for all neurons of hidden layers. Proposed method estimated individual probabilities for each neuron to compress the neural network structure.

In convolutional neural networks, default operation is convolution. Convolution can be represented as a linear operation as shown in (1).

For some layer l, we define $K^{(l)} = \left\{K_k^{(l)}\right\}$ for $k = 1, \dots, c^{(l)}$ as the convolutional neural network kernels, where $K_k^{(l)} \in R^{h^{(l)} \times \omega^{(l)} \times c^{(l-1)}}$ is the kernel of network with height $h^{(l)}$, width $\omega^{(l)}$ and channel $c^{(l-1)}$. The input matrix of layer l defines as $\widehat{X}^{(l)} \in R^{\hat{h}^{(l-1)} \times \widehat{\omega}^{(l-1)} \times c^{(l-1)}}$ with height $\hat{h}^{(l-1)}$, width $\widehat{\omega}^{(l-1)}$ and channel $c^{(l-1)}$.

From the input $\hat{X}^{(l)}$ we get dimensional section $h^{(l)} \times \omega^{(l)} \times c^{(l-1)}$ with stride s, and turn these sections into vectors. These vectors are the rows of input representation $X^{(l)} \in R^{n \times (h^{(l)} \omega^{(l)} c^{(l-1)})}$, where n – number of vectors. The vectorized kernels form the columns of the weight matrix $W^{(l)} \in R^{(h^{(l)} \omega^{(l)} c^{(l-1)}) \times c^{(l)}}$.

With this transformation, dropout operations can be applied to convolutional neural networks according to (1). The composition of pooling and activation functions can be regarded as the nonlinear function $f(\cdot)$ in (1).

In convolution neural network we exclude kernels instead of distinct neurons. Therefore, proposed technique manages to prune kernels from the convolutional network.

For the recurrent neural network, we take a multi-layer LSTM network as an example. The LSTM dropout operation can be described as

$$\begin{aligned} z_{[j]}^{(l)} &\sim Bernoulli\left(p_{[j]}^{(l)}\right), \\ \begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} &= \\ \begin{pmatrix} sigm \\ sigm \\ sigm \\ tanh \end{pmatrix} W^{(l)} &\begin{pmatrix} h_t^{(l-1)} \odot z^{(l-1)} \\ h_{t-1}^{(l)} \odot z^{(l)} \end{pmatrix}, \\ c_t^{(l)} &= f \odot c_{t-1}^{(l)} + i \odot g, \\ h_t^{(l)} &= o \odot tanh \tanh\left(c_t^{(l)}\right), \end{aligned} \tag{2}$$

where: $l = 1, \dots, L$ – is the layer number and $t = 1, \dots; T$ – is the step in the recurrent neural network; Element-wise multiplication is denoted by $\odot$; operators $sigm$ и $tanh$ defines sigmoid function and hyperbolic tangent respectively; the vector $h_t^{(l)} \in R^{n^{(l)}}$ is the output of step t at layer l; the vector $h_t^{(0)} = x_t$ is the input for the whole network at step t; the matrix $W^{(l)} \in R^{4n^{(l)} \times (n^{(l-1)} + n^{(l)})}$ is the weight matrix at layer l.

As shown in (2), proposed method uses vector of Bernoulli random variables $z^{(l)}$ to estimate dropout among distinct time steps in each network layer, while individual Bernoulli variables are used for different steps in the LSTM dropout. Proposed method tries to reduce the number of hidden dimensions in the blocks of LSTM. In case of using GRUs, dropout routine can be carried out similarly.

2.2. Design of network-optimizer

A neuron in the hidden layer, which is connected to model parameter with high redundancy value, would have a higher chance to be excluded.

Network-optimizer is designed in such way, that it takes the weights of neural network $\left\{W^{(l)}\right\}$ as inputs, estimates redundancy value in these weights and generates dropout probabilities $\left\{p^{(l)}\right\}$ for neurons of hidden elements that can be used to compress network structure.

A default approach is to train a distinct neural network for each layer of original neural network. Though, redundancy value is shared between different layers, the proposed technique obtains LSTM as the architecture for network-optimizer to learn redundancy value among multiple layers.

According to the (2), the weight in layer l of the neural network can be represented as a single matrix $W^{(l)} \in R^{d_f^{(l)} \times d_{drop}^{(l)}}$, where $d_{drop}^{(l)}$ – defines the dimension that dropout operation is applied and $d_f^{(l)}$ defines the dimension of features within each excluded element.

The weight matrix of LSTM at layer l can be represented as $W^{(l)} \in R^{4\cdot(n^{l-1})+n^l)\times n^{(l)}}$, where $d_{drop}^{(l)} = n^{(l)}$ and $d_f^{(l)} = 4 \cdot \left(n^{(l-1)} + n^{(l)}\right)$. Since we take weights from the original network layer by layer $W = \{W^{(l)}\}$, with $l = 1, \dots, L$ as the input of the network-optimizer. Instead of using a default LSTM as the architecture of optimizer, we apply a modified l-step model described as

$$\begin{pmatrix} v_i^T \\ v_f^T \\ v_o^T \\ v_g^T \end{pmatrix} = W_c^{(l)} W^{(l)} W_i^{(l)}, \begin{pmatrix} u_i \\ u_f \\ u_o \\ u_g \end{pmatrix} = W_h h_{l-1},$$

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} sigm \\ sigm \\ sigm \\ tanh \end{pmatrix} \left(\begin{pmatrix} v_i \\ v_f \\ v_o \\ v_g \end{pmatrix} + \begin{pmatrix} u_i \\ u_f \\ u_o \\ u_g \end{pmatrix} \right), \quad (3)$$

$$c_i = f \odot c_{l-1} + i \odot g,$$

$$h_l = o \odot tanh \tanh(c_l),$$

$$p^{(l)} = p_t = sigm\left(W_o^{(l)} h_i\right),$$

$$z_{[j]}^{(l)} \sim Bernoulli\left(p_{[j]}^{(l)}\right),$$

We define d_c as the dimension of the LSTM hidden state. Then $W^{(l)} \in R^{d_f^{(l)} \times d_{drop}^{(l)}}$, $W_c^{(l)} \in R^{4 \times d_f^{(l)}}$, $W_i^{(l)} \in R^{d_{drop}^{(l)} \times d_c}$, $W_h \in R^{4d_c \times d_c}$ and $W_o^{(l)} \in R^{d_{drop}^{(l)} \times d_c}$. The set of parameters of network-optimizer is defined as ϕ, where $\phi = \left\{w_c^{(l)}, W_i^{(l)}, W_h, W_o^{(l)}\right\}$. The matrix $W^{(l)}$ is the input matrix for step l in the network-optimizer, which is also the parameters of layer of the original network in (1) or (2).

In comparison with the default LSTM version that requires vectors as inputs, the modified LSTM model keeps the structure of the original matrix and uses less parameters to obtain the redundancy among the dropout elements.

Further, $W_c^{(l)}$ and $W_i^{(l)}$ transform original weight matrix $W^{(l)}$ with different sizes into fixed size matrix. The vector $z^{(l)}$ serves as template and probability $p^{(l)}$ is the exclude probabilities for the layer in the network used in (1) and (2), which is also the dropout learnt through observing the redundancy of the source network.

2.3. Network compressing routine

In formulas (1) and (2) custom dropout routines were described, which are used on the source network that should be compressed and network-optimizer used to get dropout probabilities.

Here we will discuss the details of the compressing routine. It enhances the source neural network and the network-optimizer in a step-by-step manner and enables the network-optimizer to stepwise compress the original neural network with soft deletion.

We define the source neural network as $F_W(x|z)$ and we call it opponent. It gets x as network input and produces predictions based on dropout z and parameters W, that refer to a weight $W = \{W^{(l)}\}$. We assume that $F_W(x|z)$ was trained beforehand. We define the network-optimizer by $z \sim \mu_\phi(W)$. It takes the weights of the opponent as inputs and generates the probability distribution of the mask vector z based on its own parameters ϕ. In order to enhance the optimizer to exclude hidden elements in the opponent, proposed technique follows the function

$$L = E_{Z\sim\mu_\phi}\left[L\left(y, F_w(z)\right)\right] = \sum_{z\sim\{0,1\}^{|z|}} \mu_\phi(W) \cdot L\left(y, F_W(x|z)\right), \quad (4)$$

where $L(\cdot,\cdot)$ is the function of the opponent. This function can be described as the expected loss of the network over the dropout generated by the optimizer.

Proposed technique enhances the optimizer and opponent in a stepwise way. It reduces loss function value as described in (4) by using the gradient descent on optimizer and opponent step by step. Since

dropout operations could be described as discrete sampling routines, we could not use backpropagation algorithm directly.

As a result, we apply likelihood estimator to get the gradient value over ϕ

$$\begin{aligned}\nabla_\phi L &= \sum_z \nabla_\phi \mu_\phi(W)\cdot L\big(y,F_w(x|z)\big)\\ &= \sum_z \mu_\phi(W)\nabla_\phi \, log\, log\, \mu_\phi(W)\\ &\cdot L\big(y,F_w(x|z)\big)\\ &= E_{z\sim\mu_\phi}\Big[\nabla_\phi \, log\, log\, \mu_\phi(W)\\ &\cdot L\big(y,F_w(x|z)\big)\Big]\sum_{z\sim\{0,1\}^{|z|}}\mu_\phi(W)\\ &\cdot L\big(y,F_W(x|z)\big).\end{aligned} \tag{5}$$

An estimator for (5) can be

$$\widehat{\nabla_\phi L} = \nabla_\phi \log \mu_\phi(W)\cdot L\big(y,F_w(x|z)\big),\quad z\sim\mu_\phi. \tag{6}$$

The gradient over $W^{(l)} \in W$ is

$$\begin{aligned}\nabla_{W^{(l)}} L &= \sum_z \mu_\phi(W)\\ &\cdot \nabla_{W^{(l)}} L\big(y,F_w(x|z)\big)\\ &= E_{z\sim\mu_\phi}\big[\nabla_{W^{(l)}} L(x|z)\big].\end{aligned} \tag{7}$$

In similar way, an estimator for (7) can be

$$\widehat{\nabla_{W^{(l)}} L} = \nabla_{W^{(l)}} L\big(y,F_w(x|z)\big), z\sim\mu_\phi. \tag{8}$$

Although the estimator (6) is an unbiased estimator, it will produce higher variance. A higher variance of the estimator can make the convergence slower. This means that variance reduction techniques are typically required to make the optimization feasible in practical tasks.

First variance lowering approach is to subtract a c from signal $L\big(y,F_w(x|z)\big)$ in (5) which keeps the gradient. We are tracking the average of the signal $L\big(y,F_w(x|z)\big)$ defined by c, and subtract c from the gradient (6).

Second variance lowering approach is to keep track of the average of the signal variance v, and divides the learning signal by $\big(1,\sqrt{v}\big)$.

Combing the aforementioned two variance lowering approaches, the final estimator (6) for gradient becomes

$$\begin{aligned}&\widehat{\nabla_\phi L}\\ &= \nabla_\phi \, log\, log\, \mu_\phi(W)\\ &\cdot \frac{L\big(y,F_w(x|z)\big)-c}{\big(1,\sqrt{v}\,\big)}, z\sim\mu_\phi.\end{aligned} \tag{9}$$

where c and v are the moving average of mean and the moving average of variance of signal $L\big(y,F_w(x|z)\big)$ respectively.

In comparison with other compressing techniques that deleted weights without recovery capability, the proposed method used so-called "soft" deletion by iteratively lowering exclusion probabilities of neurons with a factor $\gamma \in (0,1)$. During the experiments, the factor γ was 0.5.

Proposed method could not make optimal dropout decision at the beginning, soft deletion technique provides possibility to recover excluded element. This approach reduces the risk network degradation.

Within the compression routine, threshold of dropout, defined as τ increases from 0 with the step of ∇. The neurons of hidden layers with dropout, that is less than the threshold, will be given decay on probability, i.e., $\hat{p}_{[j]}^{(l)} \leftarrow \gamma \cdot p_{[j]}^{(l)}$.

In conclusion, the operation in optimizer (3) can be described as

$$z_{[j]}^{(l)} \sim Bernoulli\left(p_{[j]}^{(l)} \cdot \gamma^{lp_{[j]}^{(l)} \le \tau}\right) \tag{10}$$

where: l – is the function; $\gamma \in (0,1)$ is the factor and $\tau \in [0,1)$ – is the threshold. Since the operation of lowering dropout probability with the predefined factor γ is differentiable, we can still optimize the opponent and the network-optimizer through (8) and (9).

The compression process will stop when the percentage of left number of parameters in $F_W(x|z)$ is smaller than a user-defined value $\alpha \in (0,1)$.

Final step of compression is fine-tuning the resulting network with a mask $\hat{z}$, which is decided by the value τ. Mask generation (10) will be described as

$$\hat{z}_{[j]}^{(l)} = lp_{[j]}^{(l)} > \tau. \tag{11}$$

3. EXPERIMENTAL RESULTS

For an experimental evaluation of the developed technique, we will test its work in relation to the compression of the convolutional neural network VGGNet in the problem of image recognition on the CIFAR10 dataset.

The experiment was conducted on Intel Edison as a hardware platform. Intel Edison uses a Intel Atom SoC with a 500 MHz frequency and has a 1GB of

RAM and a 32Gb flash card. All experiments were carried out using only the power of the CPU. The training of neural networks took place on a desktop using a GeForce RTX 3060 video card. The compression process of the trained networks also took place on a desktop. All compressed neural networks worked using the Theano framework using only the processor's power. Matrix multiplication operations were optimized with BLAS and Sparse BLAS algorithms. No additional optimization was conducted.

For comparison with the proposed neural network compression method, MagBase [39] and SFAC [40] algorithms were taken.

MagBase is a magnitude-based network pruning algorithm. The algorithm prunes weights in convolutional kernels and fully connected layer based on the magnitude. It retrains the network connections after each pruning step and can recover the pruned weights. For convolutional and fully connected layers, MagBase searches the optimal thresholds separately.

SFAC is a sparse-coding and factorization-based algorithm. The algorithm simplifies the fully connected layer by finding the optimal code-book and code based on a sparse coding technique. For the convolutional layer, the algorithm compresses the model with matrix factorization methods. We greedily search for the optimal code-book and factorization number from the bottom to the top layer.

The task is image recognition through a low-resolution camera. During this experiment, we use CIFAR102 as our training and testing dataset. The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. It is a standard testing benchmark dataset for the image recognition tasks. While not necessarily representative of seeing objects in the wild, it offers a more controlled environment for a comparison.

VGGNet was used as network-opponent. This architecture was chosen to illustrate that proposed approach could compress deep and large network structures. Network structure is shown in Table 2.

Final compression structure for proposed method and analogues is also described in Table 2. Proposed method carried out more efficient compression. Network-optimizer uses enhanced LSTM network to learn redundancy values across network-opponent layers.

Alternative methods used redundancy information only within distinct network layer. Proposed method uses global redundancy values among hidden layers to compress network in a more efficient manner. Also, there is a performance loss by network, compressed by SFAC method. It's safe to assume that performance degradation is a result of absence of fine-tuning routine.

The compromise between network accuracy and memory consumption is shown in Fig 1. We could see that proposed technique reaches a better performance with the usage of standard weight matrix representation, while alternative methods use sparse representation.

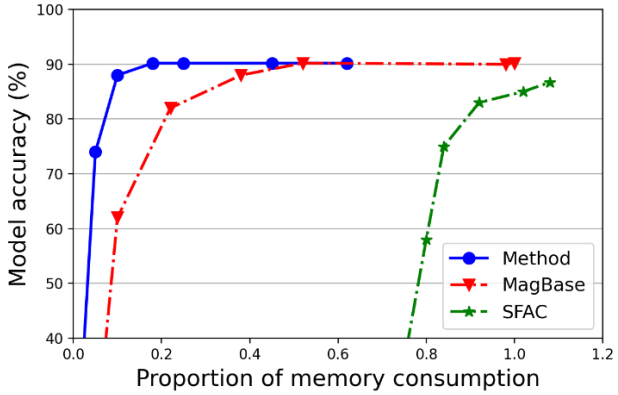

Fig. 1. **Compromise between accuracy and memory consumption**

Source: **compiled by the authors**

The compromise between accuracy and execution time is show in Fig. 2. Proposed approach achieves better performance in comparison with alternative methods. Compressed network takes 83,4ms for inference with the same accuracy, which is 93,7 % quicker than original network.

MagBase algorithm uses less execution time compared with SFAC in this experiment. Therefore, factorizing 2d kernel into two 1d kernels helps less in reducing computation time. SFAC fails to compress the original network into a small size while keeping the original performance, because SFAC avoids the fine-tuning.

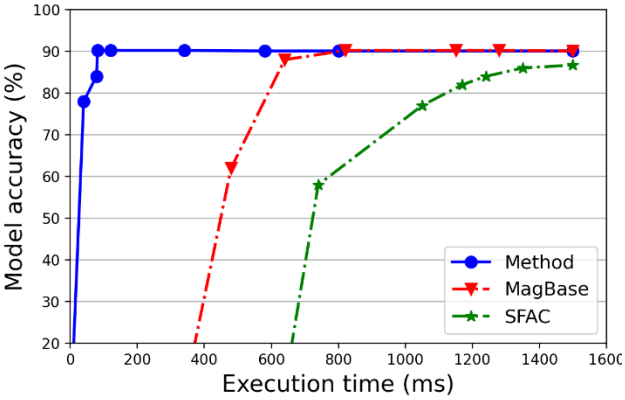

Fig. 2. **Compromise between accuracy and execution time**

Source: **compiled by the authors**

The compromise between accuracy and power consumption is shown in Fig. 3. Proposed technique lowers power consumption by 94,8 % in comparison with the original network. It facilitates development of a long-lasting deep neural network models in nodes with energy deficit.

Table 2. **Experimental results over VGGNet**

	VGGNet		Proposed method		MagBase	SFAC
Layers	Neurons of hidden layers	Parameters	Neurons of hidden layers	Percent of parameters left	Percent of parameters left	Percent of parame-ters left
Convolutional 1(3x3)	64	1600	26	42.5 %	53.5 %	93.6 %
Convolutional 2(3x3)	64	35 600	41	31.8 %	40.3 %	57.7 %
Convolutional 3(3x3)	128	72 700	55	30.7 %	52.7 %	85.7 %
Convolutional 4(3x3)	128	145 600	64	22.4 %	67.2 %	56.2 %
Convolutional 5(3x3)	256	292 700	101	21.2 %	71.7 %	85.5 %
Convolutional 6(3x3)	256	584 300	94	15.3 %	65.1 %	56.2 %
Convolutional 7(3x3)	256	584 300	85	13.1 %	61.8 %	56.3 %
Convolutional 8(3x3)	512	1 173 500	118	8.2 %	36.8 %	85.7 %
Convolutional 9(3x3)	512	2 356 200	92	4.2 %	10.3 %	56.2 %
Convolutional 10(3x3)	512	2 353 700	61	2.1 %	3.7 %	56.8 %
Convolutional 11(2x2)	512	1 045 100	125	3.2 %	3.2 %	84.7 %
Convolutional 12(2x2)	512	1 045 100	117	5.4 %	1.6 %	84.5 %
Convolutional 13(2x2)	512	1 045 100	142	6.3 %	2.2 %	84.1%
Fully connected 1	4096	2 094 700	23	0.18 %	2.3 %	95.3 %
Fully connected 2	4096	16 776 700	364	0.05 %	0.34 %	127 %
Fully connected 3	10	40 500	10	9.2 %	18.2 %	90.6 %
Total		29 647 400		2.56 %	7.07 %	108 %
Accuracy	90.2 %		90.2 %		90.2 %	86.7 %

Source: **compiled by the authors**

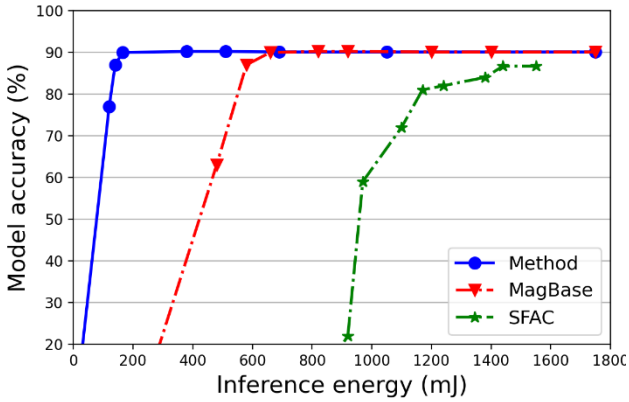

Fig. 3. **Compromise between accuracy and power consumption**

Source: **compiled by the authors**

CONCLUSIONS

In this paper, a method for the compression of neural networks is proposed, which is based on the mechanism of pruning neurons of the hidden layers. Pruning is based on a modified dropout mechanism, in which, instead of selecting a single probability of excluding neurons, the optimal parameter of the exclusion probability is selected for each neuron. To find the optimal probability of excluding each neuron, the redundancy parameter of the hidden layer neuron is used, which is estimated using a special recurrent neural network, which makes it possible to consider the spatial connections between neurons on different layers of the compressible network.

For experimental verification of the developed method, the problem of image recognition using a low-resolution camera on the CIFAR10 dataset was taken; the convolutional neural network VGGNet, which contains convolutional and fully connected layers, was used as a test network. As analogous methods, we took a method based on the principle of network pruning (MagBase), as well as a method based on sparse representation using the sparse coding method (SFAC).

Experiments showed that proposed method manages to obtain compressed structure with 2,56 % of original network parameters number. This compression results in lowering the network inference time (by 93,7 %) and power consumption by 94,8 %.

Possible directions for the continuation of the work are the further study of dependencies between the structure of the neural network and the efficiency of its operation, which will further reduce the inference time and energy consumption.

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

REFERENCES

1. Hussain, F., Hussain, R., Hassan, S. & Hossain, E. “Machine learning in iot security: current solutions and future challenges”. *IEEE Communications Surveys & Tutorials*. 2020; Vol. 22 No. 3: 1686–1721, https://www.scopus.com/authid/detail.uri?authorId=55555819300. DOI: https://doi.org/10.1109/COMST.2020. 2986444.

2. Lobachev, I. M., Antoshchuk, S. G. & Hodovychenko, M. A. “Distributed deep learning framework for smart building transducer network“. *Applied Aspects of Information Technology*. 2021: Vol. 4 No. 2: 127–139, https://www.scopus.com/authid/detail.uri?authorId=57192379296. DOI: https://doi.org/10.15276/aait.02.2021.1.

3. Gupta, A., Gupta, H., Biswas, H. & Dutta, T. “An unseen fault classification approach for smart appliances using ongoing multivariate time series”. *IEEE Transactions on Industrial Informatics*. 2020; Vol. 17 No. 6: 3731–3738. DOI: https://doi.org/10.1109/TII.2020.3016590.

4. Antoshchuk, S. G., Lobachev, I. M. & Maleryk, R. P. “Method of the Sensor Network Resources Allocation in the Conditions of Edge Computing”. *Herald of Advanced Information Technology*. 2018; Vol. 1 No.1:28–35, https://www.scopus.com/authid/detail.uri?authorId=8393582500. DOI: https://doi.org/10.15276/ hait.01.2018.3.

5. Tu, Y. & Lin, Y. “Deep neural network compression technique towards efficient digital signal modulation recognition in edge device”. *IEEE Access*. 2019; Vol. 7: 58113–58119, https://www.scopus.com/authid/detail.uri?authorId=57201253730. DOI: https://doi.org/10.1109/ACCESS.2019.2913945.

6. Saraswat, S., Gupta, A., Gupta, H. & Dutta T. “An incremental learning-based gesture recognition system for consumer devices using edge-fog computing”. *IEEE Transactions on Consumer Electronics*. 2020; Vol.66 No.1: 51–60, https://www.scopus.com/authid/detail.uri?authorId=57201912456. DOI: https://doi.org/10.1109/TCE.2019.2961066,

7. Akmandor, A., Yin, H. & Jha N. “Smart, secure, yet energy-efficient, internet-of-things sensors”. *IEEE Transactions on Multi-Scale Computing Systems*. 2018; Vol. 4 No. 4: 914–930, https://www.scopus.com/authid/detail.uri?authorId=57200168624. DOI: https://doi.org/10.1109/TMSCS.2018.2864297.

8. Palit, I., Lou, Q., Perricone, R., Niemier, M. & Hu, X. “A uniform modeling methodology for benchmarking DNN accelerators”. *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2019. 1–7, https://www.scopus.com/authid/detail.uri?authorId=35318523500. DOI: https://doi.org/10.1109/ ICCAD45719.2019.8942095.

9. Marco, V., Taylor, B., Wang, Z. & Elkhatib, Y. “Optimizing deep learning inference on embedded systems through adaptive model selection”. *ACM Transactions on Embedded Computing Systems*. 2020; Vol.19 No.1: 1–28, https://www.scopus.com/authid/detail.uri?authorId=57195337351. DOI: https://doi.org/10.1145/3371154.

10. Xiang, Y. & Kim, H. “Pipelined data-parallel cpu/gpu scheduling for multi-dnn real-time inference”. *IEEE Real-Time Systems Symposium (RTSS)*. 2019. p. 392–405, https://www.scopus.com/authid/detail.uri?authorId=57211500577. DOI: https://doi.org/10.1109/RTSS46320.2019.00042.

11. Yang, T., Chen, Y., Emer, J. & Sze, V. “A method to estimate the energy consumption of deep neural networks”. *51st Asilomar Conference on Signals, Systems, and Computers*. 2017. p.1916–1920, https://www.scopus.com/authid/detail.uri?authorId=57200278848. DOI: https://doi.org/10.1109/ACSSC.2017.8335698.

12. Wu, Y., Emer, J. & Sze, V. “Accelergy: An architecture-level energy estimation methodology for accelerator designs”. *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2019. 125–138, https://www.scopus.com/authid/detail.uri?authorId=57213421811. DOI: https://doi.org/10.1109/ICCAD45719.2019.8942149.

13. Blalock, D., Ortiz, J., Frankle, J. & Guttag, J. “What is the state of neural network pruning?” – Available from: https://arxiv.org/abs/ 2003.03033. – [Accessed March 2021], https://www.scopus.com/authid/detail.uri?authorId=56764124300.

14. He, Y., Zhang, X. & Sun, J. “Channel pruning for accelerating very deep neural networks”. – Available from: https://arxiv.org/abs/ 1707.06168. – [Accessed March 2021], https://www.scopus.com/authid/detail.uri?authorId=56333360900.

15. Liu, C. & Liu, Q. “Improvement of pruning method for convolution neural network compression”. *ICDLT '18: Proceedings of the 2018 2nd International Conference on Deep Learning Technologies*. 2018. p. 57–60, https://www.scopus.com/authid/detail.uri?authorId=57204393134. DOI: https://doi.org/10.1145/3234804.3234824.

16. Howard, A., Zhu, M., Chen B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. & Adam, H. “Mobilenets: Efficient convolutional neural networks for mobile vision applications”. –Available from: https://arxiv.org/abs/ 1704.04861. – [Accessed March 2021], https://www.scopus.com/authid/detail.uri?authorId=57191429941.

17. Molchanov, P., Mallya, A., Tyree, S., Frosio, I. & Kautz, J. “Importance estimation for neural network pruning”. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019. p. 11264–11272, https://www.scopus.com/authid/detail.uri?authorId=57191194479. DOI: https://doi.org/10.1109/CVPR.2019.01152.

18. Bhattacharya, S. & Lane, N. “Sparsification and separation of deep learning layers for constrained resource inference on wearables”. *SenSys '16: Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*. 2016. p. 176–189, https://www.scopus.com/authid/detail.uri?authorId=23135333200. DOI: https://doi.org/10.1145/2994551.2994564.

19. Huynh, L., Lee, Y. & Balan, R. “Deepmon: Mobile gpu-based deep learning framework for continuous vision applications”. *MobiSys '17: Proceedings of the 15th Annual International Conference on Mobile Systems, Applications and Services*. 2017. p. 82–95, https://www.scopus.com/authid/detail.uri?authorId=54972593600. DOI: https://doi.org/10.1145/3081333.3081360.

20. Denton, E., Zaremba, W., Bruna, J., LeCun, Y. & Fergus, R. “Exploiting linear structure within convolutional networks for efficient evaluation”. – Available from: https://arxiv.org/abs/1404.0736. – [Accessed March 2021], https://scopus.com/authid/detail.uri?authorId=55338354300.

21. Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M., & Dally, W. “Eie: efficient inference engine on compressed deep neural network”. *ACM SIGARCH Computer Architecture News*. 2016; Vol. 44 No. 3: 243–254, https://www.scopus.com/authid/detail.uri?authorId=57188982996. DOI: https://doi.org/10.1145/3007787.3001163.

22. Louizos, C., Ullrich, K. & Welling, M. “Bayesian compression for deep learning”. – Available from: https://arxiv.org/abs/1705.08665. – [Accessed March 2021], https://www.scopus.com/authid/detail.uri?authorId=56159413400.

23. He X., Zhou, Z. & Thiele, L. “Multi-task zipping via layer-wise neuron sharing”. – Available from: https://arxiv.org/abs/1805.09791. – [Accessed March 2021], https://www.scopus.com/authid/detail.uri?authorId=57203161263.

24. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. “Mnasnet: Platform-aware neural architecture search for mobile”. – Available from: https://arxiv.org/abs/1807.11626. – [Accessed March 2021], https://www.scopus.com/authid/detail.uri?authorId=57213829311.

25. Chauhan, J., Rajasegaran, J., Seneviratne, S., Misra, A., Seneviratne, A. & Lee, Y. “Performance characterization of deep learning models for breathing-based authentication on resource-constrained devices”. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*. 2018; Vol. 2 No. 4: 1–28, https://www.scopus.com/authid/detail.uri?authorId=57202425975. DOI: https://doi.org/10.1145/3287036.

26. Han, S., Mao H. & Dally W. “Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding”. – Available from: https://arxiv.org/abs/1510.00149. – [Accessed March 2021], https://www.scopus.com/authid/detail.uri?authorId=57188982996.

27. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H. & Kalenichenko, D. “Quantization and training of neural networks for efficient integer-arithmetic-only inference”. – Available from: https://arxiv.org/abs/1712.05877. – [Accessed March 2021], https://www.scopus.com/authid/detail.uri?authorId=6504282916.

28. Zhang, J., Ye, B., & Luo, X. “Mbfn: A multi-branch face network for facial analysis”, *ACAI 2019: Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*. 2019. p.542–549. DOI: https://doi.org/10.1145/3377713.3377719, https://www.scopus.com/authid/detail.uri?authorId=57192154642.

29. Georgiev, P., Bhattacharya, S., Lane N. & Mascolo, C. “Lowresource multi-task audio sensing for mobile and embedded devices via shared deep neural network representations”. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*. 2017: Vol. 1 No. 3: 1–19, https://www.scopus.com/authid/detail.uri?authorId=56411711300. DOI: https://doi.org/10.1145/3131895,

30. Lee, S. & Nirjon, S. “Neuro. zero: a zero-energy neural network accelerator for embedded sensing and inference systems”. *SenSys '19: Proceedings of the 17th Conference on Embedded Networked Sensor Systems*. 2019. p.138–152, https://www.scopus.com/authid/detail.uri?authorId=57204766313. DOI: https://doi.org/10.1145/3356250.3360030.

31. Yang, K., Xing, T., Liu, Y., Li, Z., Gong, X., Chen, X. & Fang, D. “cDeepArch: A compact deep neural network architecture for mobile sensing”. *15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. 2019; Vol. 27 No. 5: 2043–2055, https://www.scopus.com/authid/detail.uri?authorId=7409380308. DOI: https://doi.org/10.1109/SAHCN.2018.8397117.

32. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R. & Bengio, Y. “Quantized neural networks: Training neural networks with low precision weights and activations”. – Available from: https://arxiv.org/abs/1609.07061. – [Accessed March 2021], https://www.scopus.com/authid/detail.uri?authorId=56737272100.

33. Yun, S., Park, J., Lee, K. & Shin, J. “Regularizing class-wise predictions via self-knowledge distillation”. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020. p.13876–13885, https://www.scopus.com/authid/detail.uri?authorId=57214146085. DOI: https://doi.org/10.1109/cvpr42600.2020.01389.

34. Mirzadeh, S., Farajtabar, M., Li, A., Levine, N., Matsukawa, A. & Ghasemzadeh, H. “Improved knowledge distillation via teacher assistant”. – Available from: https://arxiv.org/abs/1902.03393. – [Accessed March 2021], https://scopus.com/authid/detail.uri?authorId=57215120808.

35. Yang, J., Zou, H., Cao, S., Chen, Z. & Xie, L. “MobileDA: Towards edge domain adaptation”, *IEEE Internet of Things Journal*. 2020; Vol. 7 No. 8: 6909–6918, https://www.scopus.com/authid/detail.uri?authorId=57195741598. DOI: https://doi.org/10.1109/JIOT.2020.2976762.

36. Radu, V., Lane, N., Bhattacharya, S., Mascolo, C., Marina, M. & Kawsar, F. “Towards multimodal deep learning for activity recognition on mobile devices”. *UbiComp '16: Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*. 2016. p. 185–188, https://www.scopus.com/authid/detail.uri?authorId=56045234400. DOI: https://doi.org/10.1145/2968219.2971461.

37. Mathur, A., Lane, N., Bhattacharya, S., Boran, A., Forlivesi, C. & Kawsar F. “Deepeye: Resource efficient local execution of multiple deep vision models using wearable commodity hardware”. *MobiSys '17: Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*. 2017. p. 68–81, https://www.scopus.com/authid/detail.uri?authorId=23135333200. DOI: https://doi.org/10.1145/3081333.3081359.

38. Svyatkovskiy, A., Kates-Harbeck, J. & Tang, W. “Training distributed deep recurrent neural networks with mixed precision on gpu clusters”. – Available from: https://arxiv.org/abs/1912.00286. – [Accessed March 2021], https://www.scopus.com/authid/detail.uri?authorId=36640625700.

39. Guo, Y., Yao, A. & Chen, Y. “Dynamic network surgery for efficient dnns”. – Available from: https://arxiv.org/abs/1608.04493. – [Accessed March 2021], https://www.scopus.com/authid/detail.uri?authorId=57194152792.

40. Bhattacharya S. & Lane N. “Sparsification and separation of deep learning layers for constrained resource inference on wearables”. *SenSys '16: Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*. 2016. p. 176–189, https://www.scopus.com/authid/detail.uri?authorId=23135333200. DOI: https://doi.org/10.1145/2994551.2994564.

Conflicts of Interest: the authors declare no conflict of interest

Received 22.01.2021
Received after revision 27.08.2021
Accepted 24.09.2021

DOI: https://doi.org/10.15276/hait.03.2021.3
УДК 004.93.1

Методологія компресії нейронних мереж для моделей многосенсорних трансдюсерних мереж на основі периферійних обчислень

Іван Михайлович Лобачев[1)]
ORCID: https://orcid.org/0000-0002-4859-304X; lobachev.i.m@gmail.com. Scopus ID: 57192559239
Світлана Григорівна Антощук[1)]
ORCID: https://orcid.org/0000-0002-9346-145X; asg@opu.ua. Scopus ID: 8393582500
Микола Анатолійович Годовиченко[1)]
ORCID: https://orcid.org/0000-0001-5422-3048; nick.godov@gmail.com. Scopus ID: 57188700773
[1)] Одеський національний політехнічний університет, пр. Шевченка, 1. Одеса, 65044, Україна

АНОТАЦІЯ

У цій статті основна увага приділяється розробці методу компресії нейронних мереж, який заснований на механізмі виключення нейронів прихованих шарів. Вищезазначені нейронні мережі створюються для обробки даних, що генеруються численними сенсорами, присутніми в трансдюсерних мережах, які використовуються в області створення розумних будинків. Запропонований метод реалізує єдиний підхід до компресії як згорткових нейронних мереж, так і рекурентних нейронних мереж, які використовуються для задач класифікації і регресії. Основний принцип цього методу заснований на механізмі виключення, який використовується в якості механізму регуляризації нейронних мереж. Ідея запропонованого методу полягає у виборі оптимальної ймовірності виключення нейрона прихованого шару на основі параметра надмірності. Новизна цього методу полягає у використанні спеціальної мережі-оптимізатора, яка представляє собою рекурентну нейронну мережу, що дозволяє обчислювати параметр надмірності не тільки на одному прихованому шарі, але і на кількох шарах. Додатковий аспект новизни полягає в ітеративній оптимізації мережі-оптимізатора для постійного поліпшення обчислення параметрів надмірності вхідної нейронної мережі. Для експериментальної оцінки запропонованого методу була обрана задача розпізнавання зображень камерою низького розширення, для емуляції сценарію використовувався набір даних CIFAR10. В якості експериментальної нейронної мережі була обрана згорткова нейронна мережа VGGNet, яка містить згорткові і повнозв'язні шари. В якості методів-аналогів був узятий метод MagBase, який заснований на принципі спарцифікаціі, а також метод, заснований на розрідженому представленні з використанням підходу розрідженого кодування SFAC. Результати експерименту показали, що кількість параметрів в скомпресованій моделі складає всього 2,38 % від оригінальної моделі. Це дозволило скоротити час логічного висновку на 93,7 % і споживання енергії на 94,8 %. Запропонований метод дозволяє ефективно використовувати глибокі нейронні мережі в трансдюсерних мережах, що використовують архітектуру периферійних обчислень. Це, в свою чергу, дозволяє системі обробляти дані в реальному часі, скоротити споживання енергії і час логічного висновку, а також зменшити вимоги до пам'яті та сховища для реальних додатків.

Ключові слова: Розумна будівля; інтернет речей; компресія нейронних мереж; проріджування мережі; розріджене представлення нейронної мережі; рекурентна нейронна мережа; короткочасна довгострокова пам'ять

ABOUT THE AUTHORS

Ivan M. Lobachev, PhD Student of Information Systems Department. Odessa National Polytechnic University, 1, Shevchenko Ave. Odessa, 65044, Ukraine
ORCID: https://orcid.org/0000-0002-4859-304X; lobachev.i.m@gmail.com. Scopus ID: 57192379296
***Research field*:** Deep learning; wireless sensor networks; smart cities; embedded systems; quantum computing; data mining

Іван Михайлович Лобачев, аспірант кафедри Інформаційних систем. Одеський національний політехнічний університет, пр. Шевченка, 1. Одеса, 65044,Україна

Svitlana G. Antoshchuk, Dr. Sci. (Eng), Professor, Head of Computer Systems Institute. Odessa National Polytechnic University, 1, Shevchenko Ave. Odessa, 65044, Ukraine
ORCID: https://orcid.org/0000-0002-9346-145X; asg@opu.ua. Scopus ID: 8393582500
***Research field*:** Pattern recognition; deep learning; object tracking; face recognition; graphic images formation and processing

Світлана Григорівна Антощук, доктор технічних наук, професор, директор інституту Комп'ютерних систем. Одеський національний політехнічний університет, пр. Шевченка, 1. Одеса, 65044,Україна

Mykola A. Hodovychenko, Ph.D, Associate Prof. of the Department of Project-based Learning in IT. Odessa National Polytechnic University, 1, Shevchenko Ave. Odessa, 65044, Ukraine
ORCID: https://orcid.org/0000-0001-5422-3048; nick.godov@gmail.com. Scopus ID: 57188700773
***Research field*:** Deep learning; data mining; smart cities; video processing; motion tracking; project-based Learning; patter recognition

Микола Анатолійович Годовиченко, кандидат технічних наук, доцент кафедри Проектного навчання в ІТ. Одеський національний політехнічний університет, пр. Шевченка, 1. Одеса, 65044,Україна