UDC 004.032.26(045)
DOI:10.18372/1990-5548.78.18261

[1]V. M. Sineglazov,
D. O. Kudriev

STRUCTURAL-PARAMETRIC SYNTHESIS OF CAPSULE NEURAL NETWORKS

[1]Aviation Computer-Integrated Complexes Department, Faculty of Air Navigation Electronics and Telecommunications, National Aviation University, Kyiv, Ukraine
[2]Department of Artificial Intelligence, Institute of Applied System Analysis, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine
E-mails: [1]svm@nau.edu.ua ORCID 0000-0002-3297-9060, [2]hallos977@gmail.com

Abstract*—This work is dedicated to the structural-parametric synthesis of capsule neural networks. A methodology for structural-parametric synthesis of capsule neural networks has been developed, which includes the following algorithms: determining the most influential parameters of the capsule neural network, a hybrid machine learning algorithm. Using the hybrid algorithm, the optimal structure and values of weight coefficients are determined. The hybrid algorithm consists of a genetic algorithm and a gradient algorithm (Adam). 150 topologies of capsule neural networks were evaluated, with an average evaluation time of one generation taking 10 hours. Chromosomes and weights are stored in the generation folder. The chromosome storage format is JSON, using the jsonpickle library for writing. Also, when forming a new generation, chromosome files from previous generations are used as a "cache". If a chromosome of the same structure exists, the accuracy is assigned immediately to avoid unnecessary training of neural networks. As a result of using the hybrid algorithm, the optimal topology and parameters of the capsule neural network for classification tasks have been found.*

Index Terms—Capsule neural network; structural and parametric synthesis; genetic algorithm; adaptive estimation of moment (Adam); classification problem.

I. INTRODUCTION

Artificial intelligence has truly become the hottest trend in the world of contemporary technology and business. In recent years, there has been rapid development in this field, significantly influencing various aspects of life, including medicine, education, manufacturing, finance, and many others. More and more companies and businesses are seeking to apply artificial intelligence in their processes and technologies.

The idea of capsule networks is quite intuitive and is aimed at addressing issues with convolutional neural networks, primarily the loss of important information between layers. The output of a capsule has a vector or matrix form, allowing it to convey grouped essential features to the next layer. However, capsule neural networks currently have certain drawbacks, primarily significant training slowdown due to the use of computationally expensive algorithms.

Some of the shortcomings of convolutional neural networks according to Geoffrey Hinton [1]

1) Convolutional networks are hierarchically too simple-neurons, layers. Intuitively, the human brain has more complex structures used for image recognition, so neural networks should also have them.

2) Convolutional networks lose a lot of information during the use of max pooling operation—where only the most active activation value is passed to the next layer as the dimensionality is reduced. All other information is lost. As Hinton states: "The pooling operation used in convolutional neural networks is a big mistake, and the fact that it works so well is a catastrophe."

3) Convolutional networks do not consider the "pose" of the objects they investigate – object transformations and transpositions. For a convolutional network, a digit slightly tilted at a 45-degree angle will appear foreign. This necessitates the use of a large amount of training data.

4) Another consequence of ignoring pose is that the neural network does not take into account how elements of the image are positioned relative to each other when transitioning from simpler elements to more complex ones.

Capsule networks address many issues of convolutional neural networks, performing significantly better on small-sized datasets and exhibiting superior feature learning for object recognition in images. While capsule neural networks have not yet surpassed convolutional neural networks in overall performance on complex and real-world datasets, they already yield impressive results on datasets like MNIST. Capsule

©National Aviation University, 2023

neural networks remain an active area of research in the field of modern computer vision.

The emergence of this new type of neural network necessitates the formalization of approaches and the application of algorithms to create capsule neural networks with optimal architecture and parameters for image processing tasks. The goal of this work is to identify the most crucial hyperparameters and develop an algorithm for finding the optimal network topology and training parameters.

II. Overview of Capsule Network Related Research

Capsule neural networks began to develop rapidly relatively recently. Although the idea of using capsules in neural networks had been around for some time, it wasn't until 2011 that an article [1] proposed capsules as a research direction. Only in 2017 did a work [2] introduce the "Routing by agreement" algorithm, giving rise to the architecture known as CapsNet.

In subsequent research, various modifications and alternative approaches to the original CapsNet architecture were proposed. In article [3], researchers suggested an alternative activation function for capsules and experimented with modifying the topology of the original network. Article [4] provided a formal description of the original dynamic routing approach as an optimization problem minimizing clustering loss and proposed a version.that was slightly modified The concept of group capsule networks was introduced in article [5], claiming equivariance for the output position and invariance for activations. The authors of the original CapsNet adapted the expectation-maximization algorithm for clustering similar capsule votes during routing in article [6]. Spectral capsule networks [7], based on this work, modified routing by decomposing capsule votes from previous layers into single values. Article [8] proposed a routing mechanism based on variational Bayesian methods for training a Gaussian mixture model. Stability to affine transformations was the focus of researchers in article [9], separating transformation matrices between all low-level capsules and each high-level capsule. Article [10] raised doubts about the effectiveness of the existing routing algorithm, suggesting that better results could be achieved without routing. On the other hand, researchers in article [11] demonstrated that the "Routing by agreement" mechanism is necessary for ensuring compositional structures in capsule-based networks. Nevertheless, a new architecture based on a variation of the original capsule idea, called homogeneous filter capsules without inter-layer routing, was proposed in article [12].

The attention mechanism dynamically assigns more importance to specific features considered more relevant to solving a particular problem. This idea gained popularity in various deep learning applications and was implemented in natural language processing and computer vision. In article [13], researchers applied the attention mechanism to capsule routing with a feedforward function without iterations. However, they selected low-level capsules, multiplying their activations by a parameter vector learned through backpropagation, without measuring agreement. Thus, the original "Routing by agreement" idea was distorted. Article [14] slightly modified the original dynamic routing to calculate agreement between the pose of high-level capsules and votes of low-level capsules using an inverse scalar dot product mechanism. They proposed parallel iterative routing instead of sequential, performing the routing procedure simultaneously on all capsule layers. Capsules, together with a self-attention mechanism, were applied in article [15] for entity interactions in natural language processing tasks.

In article [16], researchers consolidated previous contributions and applied a self-attention mechanism to capsules for routing information to higher-level capsules, resulting in a lightweight architecture with a small number of trainable parameters (160K).

Nevertheless, currently, there are no studies that focus on finding optimal topologies and parameters of capsule neural networks using advanced methods such as genetic algorithms. In the article [3], only a small series of experiments were conducted, which did not lead to significant improvements in results.

III. CapsNet Topology

A capsule is a group of neurons whose output represents various properties of a single entity. Instead of using the pooling operation, which loses a lot of information, capsules can be used to not only show the probability of the existence of a certain feature but also its characteristics that the neural network learns to recognize. Each layer of a capsule network contains many capsules. The activities of neurons in an active capsule reflect various properties of a specific entity present in the image. These properties may include various instantiation parameters, such as pose, deformation, velocity, albedo, shading, texture, etc. An obvious way to represent existence is to use a separate logistic block, the output of which is the probability that the entity exists. However, the probability of the existence of an entity can also be determined by the length of the feature vector.

Active capsules on one level determine, through transformation matrices, the instantiation parameters of capsules on the next level.

The weight coefficients between capsules of a lower level and a capsule at the next level are iteratively updated for each data input using the "Routing by agreement" algorithm in such a way that the output of each capsule is directed to the capsule where its output value forms a cluster with the outputs of other capsules.

In other words, a capsule processing certain elements of an image is given data from capsules at a lower level if many capsules have a similar pose and activation (eyes, lips, nose positioned to form a face).

Thus, capsule neural networks, unlike convolutional neural networks, have the following characteristics

1) Capsule networks avoid using pooling operations, which prevents information loss.

2) Capsule networks employ vector representations of features rather than scalar ones.

3) Capsule networks take into account feature properties, reducing the need for extensive training data; the network can recognize features even with slightly altered appearances.

4) Capsule networks create an understanding of the existence of higher-level features by considering the spatial relationships between low-level features.

5) Capsule networks enable the use of "Routing by Agreement" algorithms between capsules at different levels, allowing more efficient data transmission.

Topologically, these differences are achieved by introducing two new types of layers compared to convolutional neural networks.

- Convolutional capsule layer (called PrimaryCaps in article [2]). This layer performs convolution operations similar to a convolutional layer in a convolutional neural network but adjusts the dimensions of the output feature maps to represent the number of capsules in the layer and their output lengths.
- Capsule layer (called DigitCaps in article [2]). This layer replaces the pooling layers in a convolutional neural network and determines the existence and properties of higher-level features.

Capsule neural networks share a similar topology with convolutional neural networks, but they use capsule convolutional layers instead of convolutional layers and capsule layers instead of pooling layers.

Despite these changes, the first layers of the neural network can still be a regular convolutional layers. Thus, capsule neural networks can include three types of layers:

- convolutional layers;
- capsule convolutional layers;
- capsule layers.

Convolutional layers, as in convolutional neural networks, apply convolution operations to process input data and obtain feature maps used to determine the existence of specific features.

Capsule convolutional layers are similar to convolutional layers but also incorporate an additional nonlinear vector activation function.

IV. CapsNet Specific Layers

Capsule neural networks use two new types of layers compared to convolutional neural networks - capsule layers and convolutional capsule layers.

Capsule layers – layers that replace pooling layers, they receive maps of features and output a representation of the properties of a feature of a higher level in the form of vectors, where its length is the probability that this feature exists.

Let's go over the mathematical model of a capsule layer (Fig. 1).

As input to the capsule j of the layer $l + 1$ are given vectors u_i from the layer l. They are multiplied by a weight matrix W_{ij}, which performs data transformation.

$$\hat{u}_{j|i} = W_{ij} u_i.$$

Next a weighted sum of all input vectors is calculated

$$s_j = \sum_i c_{ij} \hat{u}_{j|i},$$

where c_{ij} – weight coefficient. The more the input vector "agrees" with the output vector, the greater the value of c_{ij}, which belongs to the interval [0, 1]. These coefficients are calculated with the help of routing algorithms.

As a final step, a non-linear activation function is used, so-called "squashing" – one of the novelties introduced in [1], as the activation function is vector based, instead of scalar as in classic neural networks. After its application we have the capsule output vector.

$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\|^2}.$$

For one-dimensional vector, function graph is as shown on Fig. 2.

Convolutional capsule layers – layers that closely resemble regular convolutional layers. In these layers, a convolution operation is performed with a number of filters equal to the product of the number of capsules and their dimensionality. After obtaining feature maps, an additional "squashing" operation is applied to add extra block-wise nonlinearity.

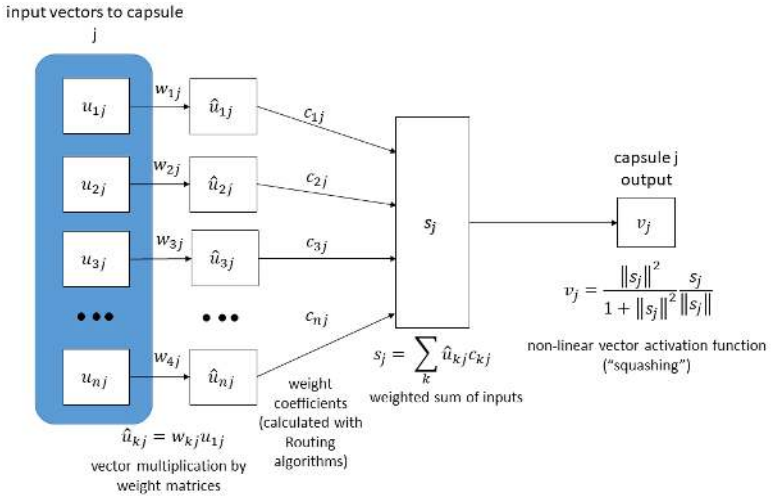

Fig. 1. Math model diagram of a capsule in a capsule layer

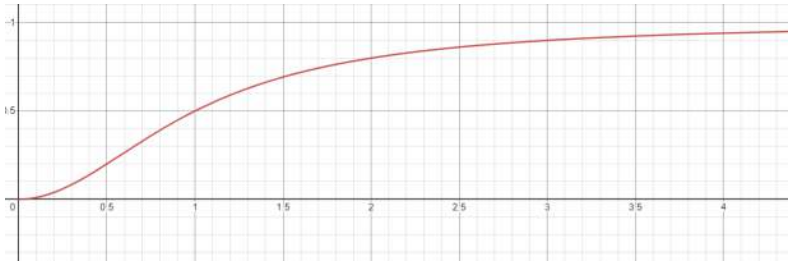

Fig. 2. "Squash" function graph for one dimensional vector

Mathematically, the convolutional capsule layer is equivalent to a convolutional layer, with the additional application of a squashing function along the dimensions of the resulting features.

$$y_i = \sigma(W_i x + b_i),$$

where matrix **W** depends on the parameters of the convolutional kernel, $i = 1 \ldots k$, $k = m \times n$, where m is the amount of capsules, n is the size of the capsule output vector.

All resulting feature maps are resized in order to have the following output vector dimensionality:

$$(H, W, n, m),$$

where H and W are height and width of resulting feature maps after convolution application. After that the "squash" function is used on the last layer.

V. Structural-Parametric Synthesis of Capsule Neural Networks

A. Choice of approach and architecture

The architecture of the capsule neural network was chosen with the incorporation of a generator because it serves as an organic regularization method for capsule neural networks (the prediction vector of a capsule already encodes object properties) and is widely applied in the literature dedicated to capsules [2], [3], [13] and [16]. The generator takes as input a vector formed by concatenating the prediction vectors of capsules. It utilizes a so-called "mask" that assigns zero values to the vectors of capsules corresponding to a class different from the true one. The implementation of capsules was chosen from the article [16] due to the following advantages:

- is ideologically similar to the original CapsNet described in [2];
- has the greatest novelty (paper was released in 2021);
- data routing, applied through the attention mechanism, accounts for capsule agreement;
- is computionally lighter than counterparts.

In the work, a capsule neural network with a generator is used as an approach to regularization, so the capsule neural network essentially consists of two neural networks. The capsule neural network is comprised of:

- convolutional layers;
- convolutional capsule layer;
- fully-connected capsule layers.

The convolutional layers can have a different number of kernels, varying strides and kernel sizes, and activation functions. The following activation functions will be considered: sigmoid, ReLU, Leaky

ReLU. The adjustable parameters are the convolutional kernel elements in the layer.

The convolutional capsule layer may have the same number of kernels as the convolutional layer that precedes it, with different strides and kernel sizes. The product of the number of capsules in the layer and their dimensions should equal the product of the dimensions of the output feature map after applying the convolution operation. The adjustable parameters also include the convolutional kernels in the layer. At the level of the capsule output vectors, a capsule compression function is used.

Capsule fully connected layers can have an arbitrary number of capsules and their sizes. The adjustable parameters are represented by affine transformation weight matrices **W** of vector values of capsules from previous layers and optimal capsule connection weight matrices **B**. "Squash" function is applied to the output vectors.

Generator neural network, on the other hand, can be comprised of:

- convolutional layers;
- deconvolutional layers;
- fully-connected layers.

In the work, a generator architecture will be employed in which a fully connected layer at the beginning transforms its shape into a small image with a large number of filters. This small image is then enlarged to the required size of the output image using layers of transposed convolutions. Additionally, an extra convolutional layer with three convolutional kernels converts the image to the RGB format.

VI. Structural-parametric Synthesis of Capsule Neural Networks Problem Statement

The mose decisive parameters of the capsule neural network are the ones below.

a) Layer count and their positioning:

- convolutional layers;
- convolutional capsule layers.

b) Capsule activation function *("squash"), can be of different types:*

- from paper [2] $v_j = \frac{\|s_j\|^2}{1+\|s_j\|^2}\frac{s_j}{\|s_j\|^2}$,
- from paper [16] $v_j = \left(1-\frac{1}{e^{\|s_j\|}}\right)\frac{s_j}{\|s_j\|^2}$.

c) Convolutional layer parameters:

- kernel size;
- stride;
- neuron activation function;
- kernel count.

d) Convolutional parameters in convolutional capsule layers: similar to parameters of convolutional layers.

e) Capsule count and dimensionality in capsule and convolutional capsule layers.

g) Capsule implementation approach:

- routing by agreement;
- expectation maximization;
- attention routing;
- homogenous vector capsules;
- self-attention routing.

Loss function for one of the capsules in the capsule neural network is the following:

$$L_c = T_c \max\left(0, m^+ - \|v_c\|\right)^2 + \lambda(1-T_c)\max\left(0, \|v_c\| - m^-\right)^2,$$

where T_c is the correct prediction flag, 1 when prediction is correct, 0, when incorrect; v_c is the capsule output vector norm; m^+ is the correct prediction margin (usually 0.9); m^- incorrect prediction margin (usually 0.1).

Thus, the entire loss function is the following:

$$L = \sum\nolimits_i L_i,$$

where $i = 1\ldots m$ are indeces of capsules in the last layer, which is responsible for classification.

In addition, in CapsNet architectures with generator usage, loss function has an additive with the error of reconstruction of the original image, which was the generator output.

$$L = \sum\nolimits_i L_i + \gamma\left\|z_{true} - z_{pred}\right\|, \qquad (1)$$

where z_{true} is the input to the capsule neural network; z_{pred} is the reconstruction created by the generator; γ is the regularization impact coefficient.

We are given a finite set $J = \{(R_j, Y_j)\}$ $j = 1, ..., P$ of pairs "attribute-value", where R_j, Y_j is the input and output neural network vectors.

The problem is to determine X optimal values of parameters of the capsule neural network, which would minimize the loss function (1).

Vector **X** can be defined in the following way:

$$\mathbf{X} = (\overline{x}_{11}, \overline{x}_{12}, \ldots, \overline{x}_{1x_2}, x_2, \overline{x}_3, \overline{x}_4, \overline{x}_5, \overline{x}_{61}, \overline{x}_{62}, \ldots, \overline{x}_{6x_7}, x_7, x_8, \overline{x}_9)^{\mathrm{T}},$$

where **X** is the vector, which defines the topology and parameters of the network, that need to be found, when solving the problem of structural-

parametric synthesis; $\overline{x}_{1i}$ is the vector of convolutional layer i properties; $\overline{x}_{1i} = (f_{1i}, k_{1i}, s_{1i}, a_{1i})^{\mathrm{T}}$, where f_{1i} is the kernel count in the convolutional layer, k_{1i} is the kernel size, s_{1i} is the stride length, a_{1i} is the neuron activation function; x_2 is the convolutional layer count; $\overline{x}_3$ is the convolutional capsule layer parameter vector.

$$\overline{x}_3 = (f_3, k_3, s_3, N_3, D_3)^{\mathrm{T}},$$

where f_3 is the kernel count in convolutional capsule layer; $f_3 \in \{32,64,96,128,160,192,224,256\}$; k_3 is the kernel size, $k_3 \in \{3,5,7,9\}$; s_3 is the stride length, $s_3 \in \{1,2,3,4,5\}$; N_3 is the capsule count, $N_3 \in \mathbb{N}$; D_3 is the capsule size, $D_3 \in \mathbb{N}$, $N_3 \times D_3 = f_3$; $\overline{x}_4$ is the capsule layer parameters vector,

$$\overline{x}_4 = (N_4, D_4)^{\mathrm{T}},$$

where N_4 is the capsule count, $N_4 = m$, where m is the amount of classes in the training dataset; D_4 is the capsule size, $D_4 \in \{8,16,32,48,64\}$; $\overline{x}_5$ is the output dimensions of a fully-connected layer at the beginning of the generator after reshaping

$$\overline{x}_5 = (x_5, z_5)^{\mathrm{T}},$$

where $\overline{x}_5$ is the height and width, $x_5 \in \{2,4,8\}$, z_5 is the channel count, $z_5 \in \{128,256,512\}$; $\overline{x}_{6i}$ is the deconvolutional layer parameters vector i in a generator, $i = 1 \ldots x_7$

$$\overline{x}_{6i} = (f_{6i}, k_{6i}, a_{6i})^{\mathrm{T}},$$

where f_{6i} is the kernel count, $f_{6i} \in \{32,64,96,128,160,192,224,256\}$, k_{6i} is the kernel size, $k_{6i} \in \{2,4,8\}$, a_{6i} is the neuron activation function, $a_{6i} \in$ {'relu', 'sigmoid', 'leaky_relu'}; x_7 is the amount of deconvolutional layers; x_8 is the kernel size of the final convolutional layer in the neural network, $x_8 \in \{1,2,3,4,5\}$; $\overline{x}_9$ is the weight parameters.

VII. Method (Algorithm) of Structural-Parametric Synthesis of Capsule Neural Networks

The structural-parametric synthesis of a capsule neural network consists of two stages – structural synthesis and parametric synthesis.

Structural synthesis is achieved by finding the optimal number of layers, their types, mutual arrangement, as well as the parameters characteristic to these layers.

Parametric synthesis aims to find optimal weight coefficients for the neural network obtained during structural synthesis, which will lead to the best overall performance.

The hybrid algorithm for structural-parametric synthesis performs the task of discovering optimal network topologies and determining optimal weight coefficients.

It combines the actions of two algorithms – the genetic algorithm and the optimization algorithm using gradient descent.

The sequence of actions is presented below.

1) Initialize neural network population.

2) Perform capsule neural network parameters optimization for m training epochs (m – is a hyperparameter which is set before training) .

3) Use gradient descent algorithm on neural networks with the best parameters which would give the highest prediction accuracy.

VIII. Results and Analysis

A. Results of applying the genetic algorithm for topology discovery

The population size was set at 30 individuals. The total number of generations, including the initial one, was set to 5. Training with the Adam optimizer took place over 10 epochs, after which the highest accuracy achieved during the network training period was assigned to the corresponding individual and its chromosome.

Thus, 150 topologies of capsule neural networks were evaluated. The average evaluation time for one generation was 10 hours. Chromosomes and weights are stored in the generation folder. The chromosome storage format is JSON, utilizing the jsonpickle library for recording. Additionally, when forming a new generation, chromosome files from previous generations are used as a "cache." If a chromosome of the same structure exists, the accuracy is immediately assigned to individuals to avoid unnecessary training of neural networks (Fig. 3).

As evident from the evolutionary trajectory, the overall quality of neural networks improved, except for the last generation where the minimum quality decreased. There is an explanation for this. Since the input image is standardized beforehand, its pixel values lie in the range [-1, 1]. In the initial version of the genetic algorithm, the activation function in the genes of the last convolutional layer could take values such as hyperbolic tangent, ReLU, Sigmoid, or Leaky ReLU. Only hyperbolic tangent has output values in the range [-1, 1]. After correcting this

mistake, many neural networks began to receive correct values for the generator loss function. The architectures of some of them were likely adapted to generator loss functions that remained unchanged, negatively impacting the training results.

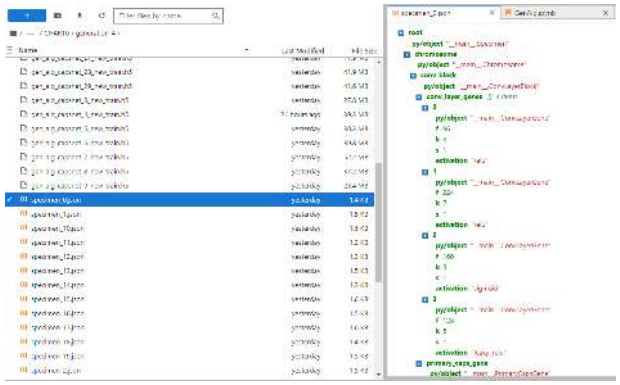

Fig. 3. File system appearence, after algorithm execution for 5 generations

Evolutionary trajectory can be seen on Fig. 4.

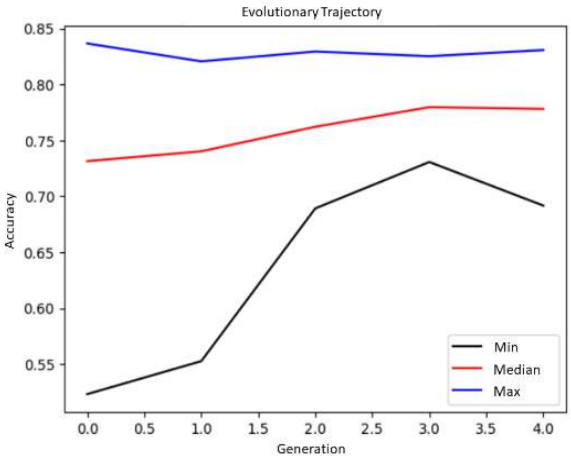

Fig. 4. Evolutionary trajectory

Five neural networks with the highest accuracy were attained (Table I).

TABLE I. NEURAL NETWORKS WITH THE HIGHEST ACCURACY

Generation	Specimen number	Accuracy	Network parameters count	Generator parameters count
0	8	0.837	3 798 064	5 645 059
4	4	0.831	5 011 312	5 270 243
2	10	0.830	2 870 192	2 906 275
3	5	0.825	3 548 816	7 887 747
2	14	0.824	3 548 816	2 013 859

Overall, even through comparative search, neural networks were found that achieved better results than those discovered manually. The baseline model had an accuracy of 0.831, and the subsequent learning curve was observed over 20 epochs (Fig. 5).

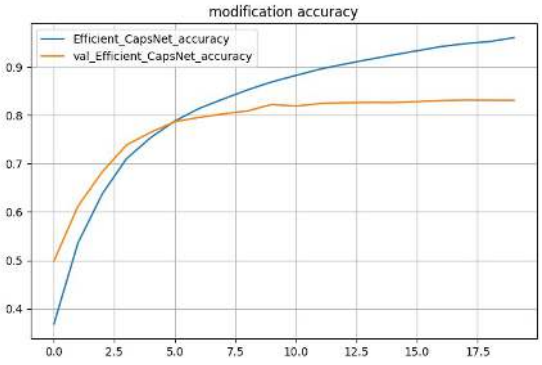

Fig. 5. Baseline neural network training curve

Below are the topologies of the best neural networks.

Generation 0. Specimen 8.

Convolutional layer $f = 64$, $k = 3$, $s = 1$, $a =$ 'Leaky ReLU'.
Convolutional layer $f = 256$, $k = 3$, $s = 1$, $a =$ 'ReLU'.
Convolutional layer $f = 160$, $k = 5$, $s = 1$, $a =$ 'Leaky ReLU'.
Convolutional layer $f = 192$, $k = 7$, $s = 1$, $a =$ 'ReLU'.
Convolutional capsule layer $f = 192$, $k = 7$, $s = 5$, $N = 72$, $D = 24$.
Capsule layer $N = 10$, $D = 64$.
Fully connected layer $x = 2$, $z = 256$.
Deconvolutional layer $f = 256$, $k = 8$, $s = 2$, $a =$ 'Sigmoid'.
Deconvolutional layer $f = 256$, $k = 2$, $s = 2$, $a =$ 'Leaky ReLU'.
Deconvolutional layer $f = 256$, $k = 2$, $s = 2$, $a =$ 'Leaky ReLU'.
Deconvolutional layer $f = 256$, $k = 2$, $s = 2$, $a =$ 'Leaky ReLU'.
Final convolutional layer $f = 3$, $k = 3$, $s = 1$, $a =$ 'Sigmoid'.

Generation 4. Specimen 4.

Convolutional layer $f = 64$, $k = 3$, $s = 1$, $a =$ 'Leaky ReLU'.
Convolutional layer $f = 256$, $k = 3$, $s = 1$, $a =$ 'ReLU'.
Convolutional layer $f = 224$, $k = 7$, $s = 1$, $a =$ 'ReLU'.
Convolutional layer $f = 192$, $k = 3$, $s = 1$, $a =$ 'Leaky ReLU'.
Convolutional layer $f = 224$, $k = 5$, i = 2, $a =$ 'ReLU'.
Convolutional capsule layer $f = 224$, $k = 7$, $s = 1$, $N = 56$, $D = 16$.
Capsule layer $N = 10$, $D = 64$.
Fully connected layer $x = 2$, $z = 256$.
Deconvolutional layer $f = 256$, $k = 8$, $s = 2$, $a =$ 'Sigmoid'.
Deconvolutional layer $f = 192$, $k = 2$, $s = 2$, $a =$ 'Leaky ReLU'.
Deconvolutional layer $f = 64$, $k = 4$, $s = 2$, $a =$ 'Leaky ReLU'.
Deconvolutional layer $f = 96$, $k = 2$, $s = 2$, $a =$ 'Sigmoid'.

Final convolutional layer $f = 3$, $k = 2$, $s = 1$, a = 'tanh'.

Generation 2. Specimen 10.

Convolutional layer f = 192, k = 3, s = 1, a = 'ReLU'.
Convolutional layer $f = 64$, $k = 3$, $s = 1$, a = 'Sigmoid'.
Convolutional layer $f = 224$, $k = 7$, $s = 1$, a = 'ReLU'.
Convolutional layer $f = 192$, $k = 3$, $s = 1$, a = 'Leaky ReLU'.
Convolutional layer $f = 224$, $k = 5$, $s = 2$, a = 'ReLU'.
Convolutional capsule layer $f = 224$, $k = 7$, $s = 1$, $N = 56$, $D = 16$.
Capsule layer $N = 10$, $D = 64$.
Fully connected layer $x = 2$, $z = 128$.
Deconvolutional layer $f = 64$, $k = 2$, $s = 2$, a = 'Leaky ReLU.
Deconvolutional layer $f = 160$, $k = 4$, $s = 2$, a = 'Leaky ReLU'.
Deconvolutional layer $f = 224$, $k = 8$, $s = 2$, a = 'Sigmoid'.
Deconvolutional layer $f = 96$, $k = 2$, $s = 2$, a = 'Sigmoid'.
Final convolutional layer $f = 3$, $k = 2$, $s = 1$, a = 'ReLU.

Generation 3. Specimen 5.

Convolutional layer $f = 64$, $k = 3$, $s = 1$, a = 'Leaky ReLU'.
Convolutional layer $f = 256$, $k = 3$, $s = 1$, a = 'ReLU'.
Convolutional layer $f = 160$, $k = 5$, $s = 1$, a = 'Leaky ReLU'.
Convolutional layer $f = 224$, $k = 3$, $s = 1$, a = 'Leaky ReLU'.
Convolutional layer $f = 192$, $k = 5$, $s = 2$, a = 'ReLU'.
Convolutional capsule layer $f = 224$, $k = 7$, $s = 1$, $N = 56$, $D = 16$.
Capsule layer $N = 10$, $D = 64$.
Fully connected layer $x = 2$, $z = 256$.
Deconvolutional layer $f = 256$, $k = 8$, $s = 2$, a = 'Sigmoid'.
Deconvolutional layer $f = 192$, $k = 2$, $s = 2$, a = 'Leaky ReLU'.
Deconvolutional layer $f = 224$, $k = 8$, $s = 2$, a = 'Sigmoid'.
Deconvolutional layer $f = 96$, $k = 2$, $s = 2$, a = 'Sigmoid'.
Final convolutional layer $f = 3$, $k = 2$, $s = 1$, a = 'ReLU'.

Generation 2. Specimen 14.

Convolutional layer $f = 64$, $k = 3$, $s = 1$, a = 'Leaky ReLU'.
Convolutional layer $f = 256$, $k = 3$, $s = 1$, a = 'ReLU'.
Convolutional layer f = 160, k = 5, s = 1, a = 'Leaky ReLU'.
Convolutional layer $f = 224$, $k = 3$, $s = 1$, a = 'Leaky ReLU'.
Convolutional layer $f = 192$, $k = 3$, $s = 1$, a = 'Leaky ReLU'.
Convolutional layer $f = 224$, $k = 5$, $s = 2$, a = 'ReLU'.
Convolutional capsule layer $f = 224$, $k = 7$, $s = 1$, $N = 56$, $D = 16$.
Capsule layer $N = 10$, $D = 64$.
Fully connected layer $x = 2$, $z = 256$.
Deconvolutional layer $f = 192$, $k = 4$, $s = 2$, a = 'Sigmoid'.
Deconvolutional layer $f = 32$, $k = 2$, $s = 2$, a = 'Leaky ReLU'.
Deconvolutional layer $f = 224$, $k = 8$, $s = 2$, a = 'Sigmoid'.
Deconvolutional layer $f = 96$, $k = 2$, $s = 2$, a = 'Sigmoid'.
Final convolutional layer $f = 3$, $k = 2$, $s = 1$, a = 'ReLU'.

B. Results of applying Adam to finish weight optimization

Additional optimization was done for 40 training epochs (thus, the total epoch count is 50) (Table II).

TABLE II. BEST NEURAL NETWORK ACCURACY AFTER TRAINING CONCLUSION

Generation	Specimen number	Accuracy	Accuracy after fine-tuning
0	8	0.837	0.849
4	4	0.831	0.861
2	10	0.830	0.850
3	5	0.825	0.862
2	14	0.824	0.861

Neural network 5 from generation 3 had the highest prediction accuracy. Below, the graphs of its additional training are presented (Figs 6 and 7).

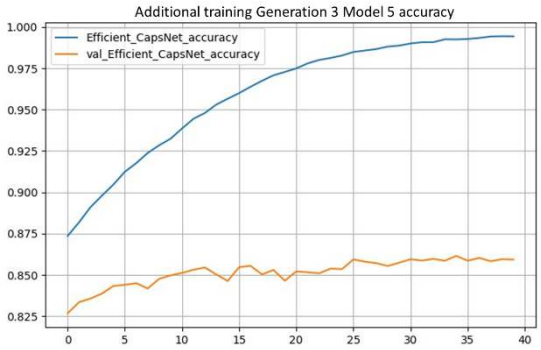

Fig. 6. Accuracy

Fig. 7. Loss functions

Thus, a neural network with improved performance compared to manually discovered architectures was obtained. However, the results are not very satisfactory, as convolutional neural networks show significantly better performance on this dataset. There are several reasons why the capsule neural network did not perform well.

1) Dataset – CIFAR-10 is a highly noisy dataset with significantly different backgrounds, which confuses capsules that should primarily encode the properties of objects in the class. Capsule neural networks have historically not outperformed convolutional networks on this dataset.

2) Dependency on generator regularization - capsule neural networks heavily rely on the regularization of the generator network. Due to an

activation function error in the last layer and the absence of a separate metric indicating the effectiveness of the generator in creating images, which would be considered in the genetic algorithm, the generators were underdeveloped.

3) Insufficient number of generations – the hybrid algorithm's application process is very time-consuming, and it would have been desirable to train it for at least 20 generations, but there was a lack of time.

IX. CONCLUSION

In this paper, the task of structural-parametric synthesis of capsule neural networks was successfully accomplished. A detailed review of contemporary mathematical implementations of capsules was conducted, and the optimal approach was chosen. A hybrid algorithm, consisting of genetic and gradient descent algorithms, was applied to find the optimal topology of the capsule neural network and its adjustable parameters. The algorithm evaluated 150 topologies of capsule neural networks, 5 of which were further fine-tuned, yielding results superior to those achieved by manually crafted capsule neural networks.

REFERENCES

[1] G. Hinton, A. Krizhevsky, and S. Wang, "Transforming Auto-Encoders," *Artificial Neural Networks and Machine Learning: ICANN 2011, 21st International Conference on Artificial Neural Networks*, Espoo, Finland, June 14-17, 2011, Proc., Part I. 44–51. https://doi.org/10.1007/978-3-642-21735-7_6.

[2] S. Sabour, N. Frosst, and G. E. Hinton, *Dynamic Routing Between Capsules*. arXiv:1710.09829. https://doi.org/10.48550/arXiv.1710.09829

[3] Edgar Xi, Selina Bing, and Yang Jin, *Capsule network performance on complex data*. arXiv: 1712.03480. https://doi.org/10.48550/arXiv.1712.03480

[4] Dilin Wang and Qiang Liu, *An optimization view on dynamic routing between capsules*, 2018. URL: https://openreview.net/forum?id=HJjtFYJDf

[5] Jan Eric Lenssen, Matthias Fey, and Pascal Libuschewski, *Group equivariant capsule networks*. arXiv: 1806.05086. https://doi.org/10.48550/arXiv.1806.05086

[6] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst, *Matrix capsules with EM routing*, 2018. URL: https://openreview.net/pdf?id=HJWLfGWRb

[7] Mohammad Taha Bahadori, *Spectral capsule networks*, 2018. URL: https://openreview.net/pdf?id=HJuMvYPaM

[8] Fabio De Sousa Ribeiro1, Georgios Leontidis, and Stefanos D Kollias, *Capsule routing via variational bayes*. arXiv: 1905.11455. https://doi.org/10.48550/arXiv.1905.11455

[9] Jindong Gu and Volker Tresp, *Improving the robustness of capsule networks to image affine transformations*. arXiv: 1911.07968. https://doi.org/10.48550/arXiv.1911.07968

[10] Inyoung Paik, Taeyeong Kwak, and Injung Kim, *Capsule networks need an improved routing algorithm*. arXiv: 1907.13327. https://doi.org/10.48550/arXiv.1907.13327

[11] Sai Raam Venkatraman, Ankit Anand, S Balasubramanian, and R Raghunatha Sarma, *Learning compositional structures for deep learning: Why routing-by-agreement is necessary*. arXiv: 2010.01488. https://doi.org/10.48550/arXiv.2010.01488

[12] Adam Byerly, Tatiana Kalganova, and Ian Dear, *No Routing Needed Between Capsules*. arXiv: 2001.09136. https://doi.org/10.48550/arXiv.2001.09136

[13] Jaewoong Choi, Hyun Seo, Suii Im, and Myungjoo Kang, *Attention routing between capsules*. arXiv: 1907.01750. https://doi.org/10.48550/arXiv.1907.01750

[14] Yao-Hung Hubert Tsai, Nitish Srivastava, Hanlin Goh, and Ruslan Salakhutdinov. Capsules with inverted dot-product attention routing. arXiv: 2002.04764. https://doi.org/10.48550/arXiv.2002.04764

[15] Dunlu Peng, Dongdong Zhang, Cong Liu, and Jing Lu, "Bg-sac: Entity relationship classification model based on self-attention supported capsule networks," *Appl. Sof Comput*. 91, 106186, 2020. https://doi.org/10.1016/j.asoc.2020.106186

[16] V. Mazzia, F. Salvetti, & M. Chiaberge, "Efficient-CapsNet: capsule network with self-attention routing," *Sci Rep* 11, Article number 14634, 2021. https://doi.org/10.1038/s41598-021-93977-0.

Received September 29, 2023

Sineglazov Victor. ORCID 0000-0002-3297-9060. Doctor of Engineering Science. Professor.
Head of the Department Aviation Computer-Integrated Complexes, Faculty of Air Navigation Electronics and Telecommunications, National Aviation University, Kyiv, Ukraine.
Education: Kyiv Polytechnic Institute, Kyiv, Ukraine, (1973).
Research area: Air Navigation, Air Traffic Control, Identification of Complex Systems, Wind/Solar power plant, artificial intelligence.
Publications: more than 700 papers.
E-mail: svm@nau.edu.ua

Kudriev Denys. Graduate Student.
Department of Artificial Intelligence, Institute of Applied System Analysis, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.
Education: National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine, (2022).
Research interests: artificial neural networks, capsule neural networks, programming.
E-mail: hallos977@gmail.com

В. М. Синєглазов, Д. О. Кудрєв. Структурно-параметричний синтез капсульних нейронних мереж
Дану роботу присвячено структурно-параметричному синтезу капсульних нейронних мереж. Розроблено методологію структурно-параметрично синтезу капсульних нейронних яка включає наступні алгоритми: визначення найбільш впливових параметрів НМ, гібридний алгоритм машинного навчання. За допомогою гібридного алгоритму визначається оптимальна структура та значення вагових коефіцієнтів. Гібридний алгоритм складається з генетичного алгоритму та градієнтного алгоритму (Adam). було оцінено 150 топологій капсульних нейронних мереж. Середній час оцінки одного покоління складав 10 годин. Хромосоми та ваги зберігаються у папку покоління. Формат збереження хромосом – json, механізм запису бібліотека jsonpickle. Також при утворення нового покоління, файли хромосом з інших поколінь використовуються як "кеш", якщо існує хромосома такого самого вигляду, то особині одразу присвоюється точність, для уникнення зайвих тренувань нейромереж. В результаті використання гібридного алгоритму знайдено оптимальну топологію та параметри капсульної нейронної мережі для вирішення задачі класифікації.

Ключові слова: капсульна нейронна мережа; структурно-параметричний синтез; генетичний алгоритм; адаптивна оцsнка момента (Adam); задача класифікації.

Синєглазов Віктор Михайлович. ORCID 0000-0002-3297-9060. Доктор технічних наук. Професор.
Завідувач кафедрою авіаційних комп'ютерно-інтегрованих комплексів. Факультет аеронавігації електроніки і телекомунікацій, Національний авіаційний університет, Київ, Україна.
Освіта: Київський політехнічний інститут, Київ, Україна, (1973).
Напрям наукової діяльності: аеронавігація, управління повітряним рухом, ідентифікація складних систем, вітроенергетичні установки, штучний інтелект.
Кількість публікацій: більше 700 наукових робіт.
E-mail: svm@nau.edu.ua

Кудрєв Денис Олексійович. Магістрант.
Кафедра штучного інтелекту, Інститут прикладного системного аналізу, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.
Освіта: Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна, (2022).
Напрям наукової діяльності: штучні нейронні мережі, капсульні нейронні мережі, програмування.
E-mail: hallos977@gmail.com