

The method of intelligent classification based on deep associative neural networks*

Eugene Fedorov^{1,*}, Tetyana Utkina^{1,†}, Olga Nechyporenko^{1,†}, Maryna Leshchenko^{1,†}, Kostiantyn Rudakov^{1,†} and Ihor Zubko^{1,†}

¹ Cherkasy State Technological University, Cherkasy, 18000, Ukraine

Abstract

A approach for intelligently classifying the state of chicken eggs based on deep associative neural networks is proposed. This method aims to automate the recognition and interpretation of chicken egg ovoscopy visualization results during incubation. The model of the associative autoencoder offers several advantages over traditional methods. For instance, the input image is pre-sized, and the pairs count "convolutional - pooling/upsampling layer" is defined practically, depending on the image size, which improves the accuracy of classification. Additionally, the planes count is determined as the dividing quotient the cells count in the layer of input by two to the power of the doubled pairs count "convolutional - pooling/upsampling layer" to retain the total cells count in the layer after pooling/upsampling. This process halves the layer planes size in width and height, automating the structure definition of the model layers. The deep Boltzmann machine model offers several advantages over the traditional deep Boltzmann machine. These include pre-resizing the input image, determining the number of limited Boltzmann machines empirically to increase classification accuracy, and setting the neurons count in the hidden layers as double the neurons count in the visible layer to satisfy the Kolmogorov theorem on the representation of multidimensional continuous functions by a superposition of one-dimensional continuous functions. This model automates the definition of the model layer's architecture. The intelligent classification method of chicken eggs developmental state, based on deep associative neural networks, can be applied in intelligent systems for classifying the results of chicken eggs candling visualization during incubation in industrial poultry production.

Keywords

ovoscopy, intelligent classification, deep associative neural networks, convolutional autoencoder, deep Boltzmann machine

1. Introduction

When incubating eggs, it is crucial to consider several factors, such as maintaining the appropriate temperature and humidity, monitoring the composition of the ventilated air, and using high-quality, fresh, and fertilized eggs to ensure the production of healthy offspring. Before placing the eggs in the incubator, it is important to conduct ovoscopy to check for egg integrity and the possibility of development. Ovoscopy involves transilluminating the eggs to select high-quality ones without structural damage, which is essential for obtaining a healthy brood.

The ovoscopy process is the first stage before using the incubator. This process is usually repeated 2-3 times during the incubation period to check for defects in the shell and inside the eggs, the absence of an air chamber, and the presence of any embryo abnormalities. This helps in identifying and rejecting eggs with pathologies or other developmental disorders [1].

The cause of non-developing offspring may be due to abnormal egg shape, thinned or damaged shell (pits, protrusions, roughness, dark spots), calcareous growths on the shell, the presence of foreign objects or clots, the presence of two yolks at once, the yolk location not in the center or its


IDDMM'24: 7th International Conference on Informatics & Data-Driven Medicine, November 14 - 16, 2024, Birmingham, UK

* Corresponding author.

† These authors contributed equally.

✉ y.fedorov@chdtu.edu.ua (E. Fedorov); t.utkina@chdtu.edu.ua (T. Utkina); olne@ukr.net (O. Nechyporenko); mari.leshchenko@gmail.com (M. Leshchenko); k.rudakov@chdtu.edu.ua (K. Rudakov); i.zubko@chdtu.edu.ua (I. Zubko)

 0000-0003-3841-7373 (E. Fedorov); 0000-0002-6614-4133 (T. Utkina); 0000-0002-3954-3796 (O. Nechyporenko); 0000-0002-0210-9582 (M. Leshchenko); 0000-0003-0000-6077 (K. Rudakov); 0000-0002-3318-3347 (I. Zubko)

 © 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

displacement, the yolk remains in place when the egg is turned, displacement of the air chamber, or the absence of the embryo [2]. These signs indicate the need to exclude such eggs from the incubation process to avoid wasting resources on non-viable specimens (sterile or dead). Timely removal of these eggs from the incubator will reduce the other eggs risk being contaminated with harmful microorganisms, avoid excessive water evaporation, and eliminate the source of pollution.

To reduce the number of unsuitable eggs for incubation, the following conditions must be observed from the very beginning: collect eggs 3-4 times a day; do not incubate eggs laid after 18 hours; ensure hygiene monitoring and check for the absence of mechanical damage to the protective egg film when collecting with appropriate machines, automated lines, or by workers: to replace the litter as needed. Chicken eggs should be stored for no more than 5 days, while duck and turkey eggs should not be stored for more than 8 days, and goose eggs for no more than 10 days. Maintain the air temperature within the range of 10-15 degrees Celsius and the relative humidity between 70-80% during storage. Ensure that the temperature does not exceed +27 degrees Celsius, as this can lead to the development of the embryo, resulting in unsuccessful incubation. Similarly, the temperature should not fall below +8 degrees Celsius, as this could cause irreversible chemical changes in the eggs [3].

Throughout the entire development cycle, the embryo changes color and condition. As a result, ovoscopy is used to monitor the proper development of the fetus. Ovoscopy is conducted in a warm, darkened room while observing fire safety regulations. It is crucial that the eggs are exposed to heat rays for no more than 15-20 seconds during ovoscopy to prevent overheating, which can render them unsuitable for incubation.

A fresh egg, when examined with an ovoscope, should display the following characteristics: a consistent shell; a small air pocket at the rounded end of the egg; the yolk positioned centrally or slightly closer to the rounded end, with indistinct boundaries; when the egg is rotated, the yolk should rotate with some resistance; and there should be no foreign objects inside the egg.

The following defects should be observed during conducting ovoscopy on non-viable egg specimens [1-3]:

- light stripes on the shell indicate possible damage in the oviduct, with the crack sealed with additives;
- a marbled shell may indicate an uneven distribution of calcium, resulting in spots on the shell;
- the air chamber may be located on the side or at the sharp end of the egg, indicating delamination of the subshell membranes;
- a large air chamber suggests an old egg;
- if the yolk is not visible and the egg color is orange-red, it may indicate that the yolk has broken and mixed with the protein;
- movement of the yolk along and across the egg may indicate hailstones are torn off;
- if the yolk is stuck in one place, it could be due to improper storage, causing it to stick to the shell;
- two yolks may indicate a genetic failure;
- blood clots inside the egg may suggest hemorrhage in the oviduct;
- the presence of foreign objects inside the egg, such as grains of sand, feathers, or worm eggs, may be due to their entry into the oviduct;
- dark spots under the shell or a completely dark egg may indicate the development of a mold colony, known as “punches”.

Not all eggs selected based on external signs during candling will necessarily hatch, as only fertilized eggs will do so during the early stages of ovoscopy. Fertilized eggs can be identified after 5-7 days of incubation during the next stage of ovoscopy. There will be no signs of embryo development during candling.

Monitoring egg development allows for the assessment of the progress of the incubation process and the identification of eggs with frozen fetuses due to hypothermia, overheating, or sticking to the film for the timely removal of such specimens from the incubator. This will prevent the spread of infection and reduce energy losses for incubating non-viable specimens, as well as adjust the microclimate parameters in the incubator in order to increase the percentage of offspring hatchability [4].

Today, to increase the efficiency of monitoring the state of development of the embryo of poultry eggs during otoscopy, various methods of visualizing the incubation process are used: tomography, magnetic resonance and infrared visualization, microscopy, ultrasound, thermal difference, digital signal processing, etc. [3, 4].

Intelligent identification methods are used to conduct a qualitative assessment of the results of otoscopy of poultry eggs. This allows for high classification accuracy, improved quality of monitoring of the incubation process, and reduced costs in industrial poultry production.

Currently, deep neural networks [5,6] using parallel and distributed computing [7] have become widespread for intelligent image classification, surpassing pseudo-two-dimensional hidden Markov models in popularity.

The first class of deep non-associative neural networks are convolutional networks such as:

1. LeNet-5 [8], AlexNet [9], and VGG (Visual Geometry Group) [10] neural networks are based on convolutional pairs and pooling layers, as well as dense layers.
2. The ResNet family of neural networks [10] are built on the Residual block.
3. The DenseNet (Dense Convolutional Network) [11] neural network is based on a dense block, which comprises a Residual blocks.
4. The GoogLeNet (Inception V1) neural network [12] is based on the Inception block.
5. Inception V3 [13] is based on Inception and Reduction blocks.
6. Inception-ResNet-v2 [14] is based on Inception and Reduction blocks.
7. Xception [15] is based on the Depthwise separable convolution block.
8. The MobileNet neural network [16] is based on the Depthwise separable convolution block.
9. MobileNet2 neural network [17] is based on the Inverse Residual block.
10. The SR-CNN neural network [18] is based on the Squeeze-and-Excitation – Residual block.

The second class of deep non-associative neural networks are convolutional networks such as:

1. ViT (Visual Transformer) [19] uses normalization layers, Multi-Head Attention, and a two-layer MLP.
2. DeiT [20] utilizes distillation token, normalization layers, Multi-Head Attention, and a two-layer MLP.
3. DeepViT (Deep Visual Transformer) [21] is based on normalization layers, Re-Attention (replaces Multi-Head Attention), and a two-layer perceptron.
4. CaiT [22] relies on normalization layers, Multi-Head Attention or Class-Attention, and a two-layer MLP.
5. CrossViT [23] is based on normalization layers, Cross-Attention (replaces Multi-Head Attention), and a two-layer perceptron.
6. Compact Convolutional Transformer (CCT) [24] is based on convolutional and downsampling layers, normalization layers, Multi-Head Attention and a two-layer perceptron, and a pooling sequence layer.
7. Pooling-based Vision Transformer (PiT) [25] is based on depthwise convolutional and downsampling layers, normalization layers, Multi-Head Attention and a two-layer perceptron.
8. LeViT [26] is based on distillation token, convolutional and downsampling layers, normalization layers, LeViT Attention (replaces Multi-Head Attention) and a two-layer perceptron.

9. Convolutional vision Transformer (CvT) [27] is based on convolutional and downsampling layers, normalization layers, Multi-Head Attention and a two-layer perceptron.
10. MobileViT [28] is based on Inverse Residual blocks, convolutional layers, normalization layers, Multi-Head Attention and a two-layer perceptron.

Deep non-associative networks have the following drawbacks [29]:

- Difficulty in determining the parameters of the architecture of a deep associative neural network (patch size, size and count of layers, etc.);
- Insufficiently high training speed;
- Insufficiently high recognition accuracy.

Consequently, the issue of creating an effective deep associative neural network that addresses these concerns is relevant.

The first class of such networks are autoencoders such as:

- Convolutional autoencoder [30, 31] uses convolutional layers, upsampling/downsampling layers;
- Variational autoencoder [32, 33] considers the influence of Gaussian noise.

The second class of such networks is the deep Boltzmann machine [34]. In order to achieve our goal, we need to accomplish the following tasks:

1. To create a model to classify the state of chicken eggs using a convolutional autoencoder.
2. To develop a model to classify the state of chicken eggs using a deep Boltzmann machine.
3. To choose the quality criteria for the egg state classification method.
4. To determine the structure of the egg state classification method.
5. To perform a numerical research of the offered egg state classification approach.

The aim of the work is to improve the quality of classification of the state of chicken eggs by using deep associative neural networks.

2. Creating an ovoscopy model based on a convolutional autoencoder

Figure 1 shows a convolutional autoencoder for sample recovery, a dynamic non-recurrent network with a hierarchical architecture.

The input image in this type of convolutional autoencoder is pre-resized, and the “convolutional - pooling layer” pairs count is defined practically, based on the image size. Additionally, the planes count is determined automatically as the quotient of dividing the cells count in the layer of input by a power of two. This power is equal to twice the “convolutional - pooling/upsampling layer” pair count. This method allows for the preservation of the total cells count in the layer after pooling/upsampling, which effectively reduces/increases the layer planes size by two times in width and height.

In contrast to MLP, the input layer and output layer have the same neurons count and receive the same data. The hidden layers count is always odd, and the number of neurons in the hidden layers is fewer than the neurons count in the input/output layer. This decreases as we approach the central (code) hidden layer. Every autoencoder consists of an encoder and a decoder. When there's only one hidden layer and a linear activation function, the autoencoder becomes similar to PCANN.

The autoencoder implements an auto-associative memory (a pair of samples ($\mathbf{m}_x, \mathbf{m}_y$), $\mathbf{m}_y = \mathbf{m}_x$, representing its element (cell)) and restores (extracts) the stored sample \mathbf{m}_y by the key sample \mathbf{m}_x corresponding to the input vector \mathbf{x} . The most important property of a convolutional

autoencoder is that the same network with the same connection weights can store and reproduce several stored samples.

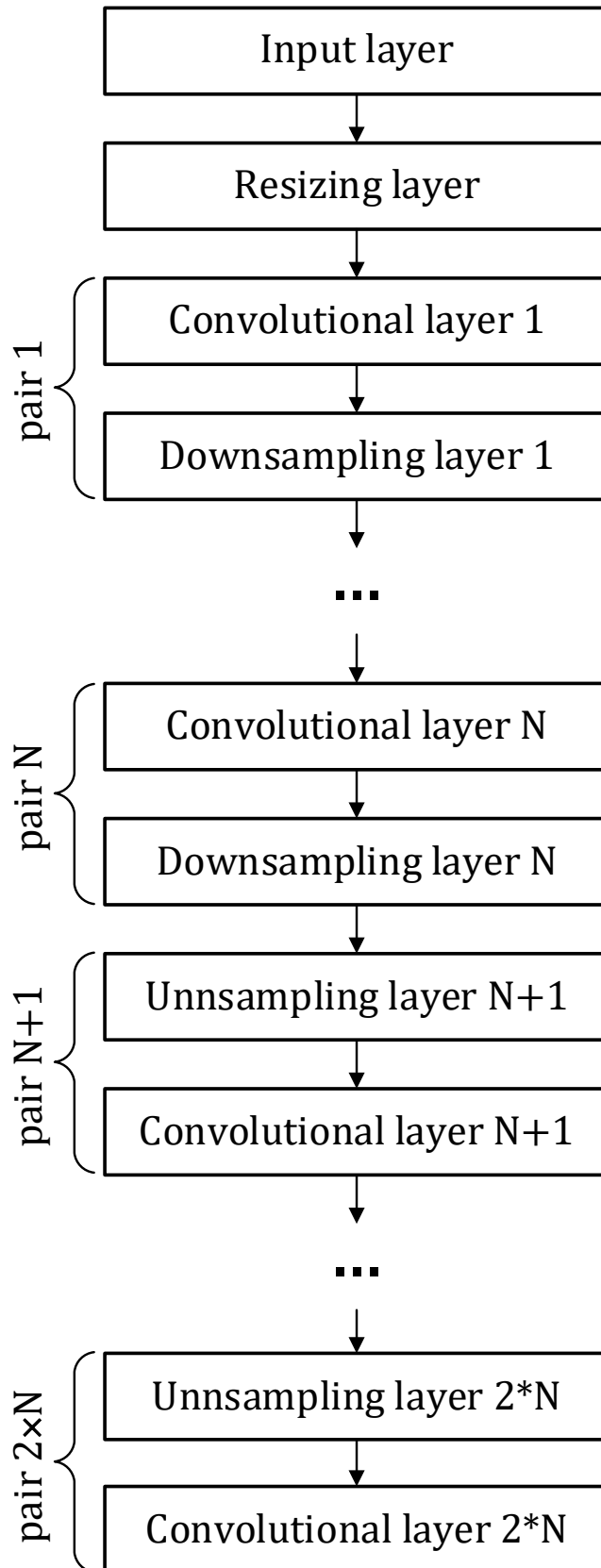


Figure 1: Convolutional autoencoder neural network

The convolutional autoencoder utilizes error correction learning (supervised learning), with the most common method being gradient descent, which forms the basis of the backpropagation (BP) technique for feedforward ANNs.

The model for the convolutional autoencoder is presented below.

Let v is the position, $v = (v_x, v_y)$, S_e is the pooling (or upsampling) layer, C_l is the convolutional layer, K_{S_e} and K_{C_l} are the cell planes count, A_l is the connection area of the layer plane S_e and C_l , \hat{L} are the convolutional layers count.

1. $l = 1$.
2. The output signal for the convolutional layer is being calculated.

$$\begin{aligned} u_{c_l}(m, i) &= ReLU(h_{c_l}(m, i)), \\ m &\in \{1, \dots, N_{c_l}\}^2, i \in \overline{1, K_{c_l}}, \\ K_{c_l} &= \begin{cases} 2^{2l}, & l \leq \hat{L}/2, \\ 2^{2(\hat{L}-l+1)}, & l > \hat{L}/2, \end{cases} \end{aligned} \quad (1)$$

$$h_{c_l}(m, i) = \begin{cases} b_{c_1}(i) + \sum_{v \in A_1} w_{c_1}(v, 1, i)x(m + v), & l = 1, \\ b_{c_l}(i) + \sum_{k=1}^{K_{S_{l-1}}} \sum_{v \in A_{l-1}} w_{c_l}(v, k, i)u_{S_{l-1}}(m + v, k), & l > 1, \end{cases} \quad (2)$$

where $w_{c_1}(v, 1, i)$, $w_{c_l}(v, k, i)$ is the connection weight, $u_{c_l}(m, i)$ is the cell output.

If $l \leq \hat{L}/2$, then to calculate the output signal for the downsampling layer (downsample by a factor of 2)

$$\begin{aligned} u_{S_e}(m, k) &= \max_{v \in \{0,1\}^2} \{u_{c_l}(2m + v, k)\}, \\ m &\in \{1, \dots, N_{S_l}\}^2, k \in \overline{1, K_{S_l}}, K_{S_l} = 2^{2l}, \end{aligned} \quad (3)$$

where $w_{S_l}(k, k)$ is the connection weight, $u_{S_l}(m, k)$ is the cell output.

If $l \geq \hat{L}/2$, then to calculate the output signal.

$$u_{S_l}(2m + v, k) = \begin{cases} u_{S_l}(m, k), & l = \frac{\hat{L}}{2}, \\ u_{c_l}(m, k) & l > \frac{\hat{L}}{2}, \end{cases} \quad (4)$$

$$\begin{aligned} v &\in \{0,1\}^2, m \in \{1, \dots, N_{S_l}\}^2, \\ k &\in \overline{1, K_{S_l}}, K_{S_l} = 2^{2(\hat{L}-l)} \end{aligned} \quad (5)$$

where $u_{S_l}(m, k)$ is the cell output.

If $l \leq \hat{L}$, then increment l , go to 2.

The output signal for the output convolutional layer is being calculated.

$$\begin{aligned} u_o(m, 1) &= \text{sigm}(h_o(m, 1)), \\ m &\in \{1, \dots, N_o\}^2, \\ h_o(m, 1) &= b_o(1) + \sum_{k=1}^{K_{S_L}} \sum_{v \in A_L} w_o(v, k, 1)u_{S_L}(m + v, k), \end{aligned} \quad (6)$$

where $w_o(v, k, 1)$ is the connection weight, $u_o(m, 1)$ is the cell output.

Note: an example of upsampling: the matrix $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ is transformed to the matrix $\begin{bmatrix} 1 & 1 & 2 & 2 \\ 3 & 3 & 4 & 4 \\ 3 & 3 & 4 & 4 \end{bmatrix}$.

Note: An example of pooling: the matrix $\begin{bmatrix} 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ 3 & 3 & 4 & 4 \\ 3 & 3 & 4 & 4 \end{bmatrix}$ is transformed to the matrix $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$.

3. Creating an ovoscopy model based on the deep Boltzmann machine

Figure 2 shows a generative Bernoulli-Bernoulli deep Boltzmann machine, also known as a deep Boltzmann machine (DBM). This is a type of recurrent neural network that comprises of one visible and multiple hidden layers. The input image is pre-resized, unlike traditional DBMs.

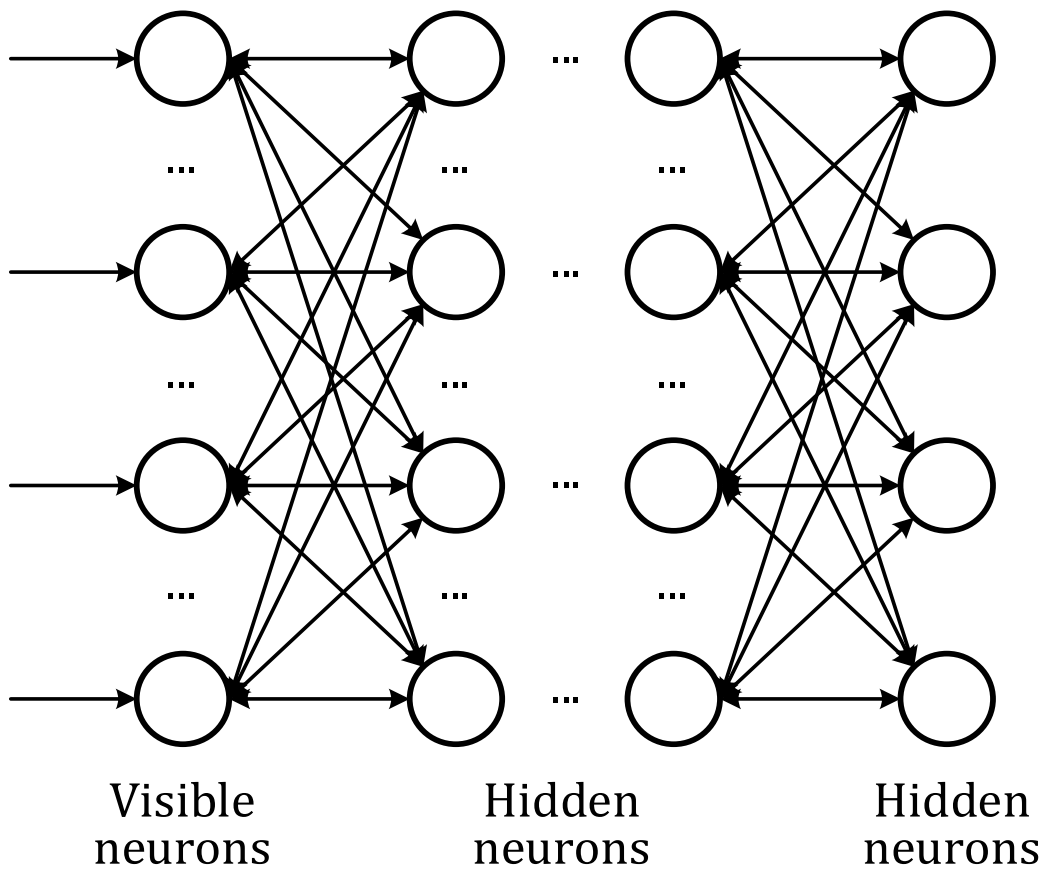


Figure 2: The deep Boltzmann machine (DBM)

Additionally, the number of restricted Boltzmann machines (RBM) is defined practically, which helps improve the model classification accuracy. Furthermore, the neurons count in the hidden layers is twice the neurons count in the visible layer, which satisfies Kolmogorov's theorem on the representation of a multidimensional continuous function by a superposition of one-dimensional continuous functions.

Generative DBM is a composition of only generative RBMs.

The most crucial property of DBM is that the same neural network with the same connection weights can store and reproduce several different memorized samples.

The components of DBM are stochastic neurons whose state is defined in the form:

$$x_j = \begin{cases} 1, & \text{with } P_j, \\ 0, & \text{with } 1 - P_j, \end{cases} \quad (7)$$

$$P_j = \frac{1}{1 + \exp(-\Delta E_j)} \quad (8)$$

where P_j is the probability, ΔE_j is the neural network energy increment.

The Deep Boltzmann Machine model is presented as follows.

During the bottom-up pass (recognition phase), the following steps are executed: 1-6.

Bottom-up pass (recognition phase) (steps 1-6)

1. Initialization of binary vectors of hidden neurons states $\mathbf{x}_1^{(l)} = (x_1^{(l)}, \dots, x_{N^{(l)}}^{(l)})$, $l \in \overline{1, L}$.
2. $l = 1$, $\mathbf{x}^{(0)} = \mathbf{x}^{in}$.

Positive phase (step 3)

3. The state of hidden neurons of the l layer $j \in \overline{1, N^{(l)}}$ is being calculated.

$$P_j = \begin{cases} \frac{1}{1 + \exp(-b_j^{(l)}(n) - \sum_{i=1}^{N^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)} - \sum_{i=1}^{N^{(l+1)}} w_{ij}^{(l+1)} x_i^{(l+1)})}, & l < L, \\ \frac{1}{1 + \exp(-b_j^{(l)}(n) - \sum_{i=1}^{N^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)})}, & l = L. \end{cases} \quad (9)$$

$$x_j^{(l)} = \begin{cases} 1, & P_j \geq U(0,1), \\ 0, & P_j < U(0,1). \end{cases} \quad (10)$$

Negative phase (steps 4-6)

4. The state of visible or hidden neurons of the $l - 1$ layer $j \in \overline{1, N^{(l-1)}}$ is being calculated.

$$P_j = \begin{cases} \frac{1}{1 + \exp(-b_j^{(l-1)}(n) - \sum_{i=1}^{N^{(l)}} w_{ij}^{(l)}(n) x_i^{(l)})}, & l = 1, \\ \frac{1}{1 + \exp(-b_j^{(l-1)}(n) - \sum_{i=1}^{N^{(l)}} w_{ij}^{(l)}(n) x_i^{(l)} - \sum_{i=1}^{N^{(l-2)}} w_{ij}^{(l-2)}(n) x_i^{(l-2)})}, & 1 < l \leq L. \end{cases} \quad (11)$$

$$x_j^{(l-1)} = \begin{cases} 1, & P_j \geq U(0,1), \\ 0, & P_j < U(0,1). \end{cases} \quad (12)$$

5. If $l < L$, then $l = l + 1$, go to step 3.

6. The state of hidden neurons of the L layer $j \in \overline{1, N^{(L)}}$ is being calculated.

$$P_j = \frac{1}{1 + \exp(-b_j^{(L)}(n) - \sum_{i=1}^{N^{(L-1)}} w_{ij}^{(L-1)}(n) x_i^{(L-1)})}, \quad (13)$$

$$x_j^{(L)} = \begin{cases} 1, & P_j \geq U(0,1), \\ 0, & P_j < U(0,1). \end{cases} \quad (14)$$

Top-down traversal (spawning phase) (steps 7-11)

7. $l = L - 1$.

Positive phase (step 8)

8. The state of hidden or visible neurons of the l layer $j \in \overline{1, N^{(l)}}$ is being calculated.

$$P_j = \begin{cases} \frac{1}{1 + \exp\left(-b_j^{(l)}(n) - \sum_{i=1}^{N^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)} - \sum_{i=1}^{N^{(l+1)}} w_{ij}^{(l+1)} x_i^{(l+1)}\right)}, & l > 0, \\ \frac{1}{1 + \exp\left(-b_j^{(l)}(n) - \sum_{i=1}^{N^{(l+1)}} w_{ij}^{(l+1)} x_i^{(l+1)}\right)}, & l = 0. \end{cases} \quad (15)$$

$$x_j^{(l)} = \begin{cases} 1, & P_j \geq U(0,1), \\ 0, & P_j < U(0,1). \end{cases} \quad (16)$$

Negative phase (steps 9-11)

9. The state of hidden neurons of the $l + 1$ layer $j \in \overline{1, N^{(l+1)}}$ is being calculated.

$$P_j = \begin{cases} \frac{1}{1 + \exp\left(-b_j^{(l+1)}(n) - \sum_{i=1}^{N^{(l)}} w_{ij}^{(l+1)} x_i^{(l)}\right)}, & l = L - 1, \\ \frac{1}{1 + \exp\left(-b_j^{(l+1)}(n) - \sum_{i=1}^{N^{(l)}} w_{ij}^{(l+1)} x_i^{(l)} - \sum_{i=1}^{N^{(l+2)}} w_{ij}^{(l+2)} x_i^{(l+2)}\right)}, & 0 < l \leq L - 1. \end{cases} \quad (17)$$

$$x_j^{(l+1)} = \begin{cases} 1, & P_j \geq U(0,1), \\ 0, & P_j < U(0,1). \end{cases} \quad (18)$$

10. If $l > 0$, then go to step 8.

11. The state of visible neurons of zero layer $j \in \overline{1, N^{(0)}}$ is being calculated.

$$P_j = \frac{1}{1 + \exp\left(-b_j^{(0)}(n) - \sum_{i=1}^{N^{(1)}} w_{ij}^{(1)} x_i^{(1)}\right)}, \quad (19)$$

$$x_j^{(0)} = \begin{cases} 1, & P_j \geq U(0,1), \\ 0, & P_j < U(0,1). \end{cases} \quad (20)$$

The result is a pattern $\mathbf{m}_y = (x_1^{(0)}, \dots, x_{N^{(0)}}^{(0)})$.

4. Selection of quality criteria for classifying the state of chicken eggs.

The next criteria were elected to evaluate the training of the suggested mathematical models of deep associative neural networks:

Accuracy criterion:

$$\text{Accuracy} = \frac{1}{I} \sum_{i=1}^I [\mathbf{d}_i = \hat{\mathbf{y}}_i] \rightarrow \max_W, \quad (21)$$

$$\hat{y}_{ij} = \begin{cases} 1, & j = \arg \max_z y_{iz}, \\ 0, & j \neq \arg \max_z y_{iz}. \end{cases}$$

Categorical cross-entropy criterion:

$$CCE = -\frac{1}{I} \sum_{i=1}^I \sum_{j=1}^K d_{ij} \ln y_{ij} \rightarrow \min_W, \quad (22)$$

where y_i – is i -th model output vector, $y_{ij} \in [0,1]$, d_i – is i -th test vector, $d_{ij} \in \{0,1\}$, I – is the learning set power, K – is the classes count, W – is weights vector.

5. Determination of the structure of the method for classifying the condition of chicken eggs

Figure 3 shows the structural diagram of the method for classifying the state of chicken eggs for the proposed deep associative neural networks. The convolutional autoencoder is proposed to be trained based on stochastic metaheuristics to improve the quality of classification [35].

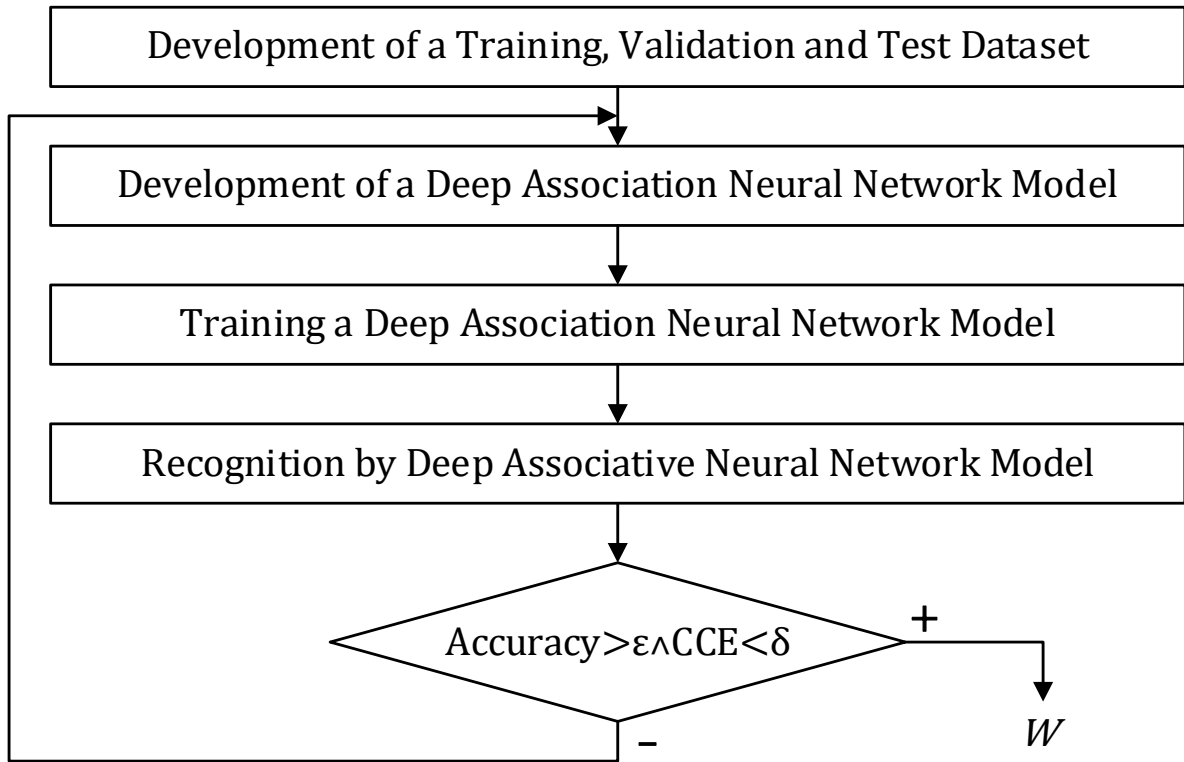


Figure 3: Block diagram of the method for classifying the state of chicken eggs

6. Numerical research

Numerical research was carried out using the Chicken benchmark [37], which consisted of RGB images sized 1080x800. Out of the total 380 photos, 80% were randomly chosen for the learning set, while the remaining 20% were used for the test and validation sets. Since the proposed deep associative neural networks do not employ recurrent connections, their parameters were determined using a GPU. The “Keras” module was utilized to realise the suggested deep associative neural networks, and “Google Colab” was selected as the environment of software.

Table 1 shows the structure of the convolutional autoencoder model, with a resizing layer added to the beginning of the neural network.

Table 1
Convolutional Autoencoder Model Structure

Layer Type	Filters	Filter Size / Stride	Output
Resize	-	-	256x256
Convolutional	4	3x3 / 1	256x256
Downsampling	4	2x2 / 2	128x128
Convolutional	16	3x3 / 1	128x128
Downsampling	16	2x2 / 2	64x64
Upsampling	16	2x2 / 2	128x128
Convolutional	16	2x2 / 1	128x128
Upsampling	4	2x2 / 2	256x256
Convolutional (output)	4	2x2 / 1	256x256

Table 2 shows the structure of the deep Boltzmann machine model, with a resizing layer added to the beginning of the neural network.

Table 2
Deep Boltzmann Machine Model Structure

Layer Type	Output
Resize	256x256
Hidden	512x512
Hidden	512x512
Hidden	512x512

Figure 4 shows how the loss (measured by categorical entropy) changes with the iterations count for the convolutional autoencoder.

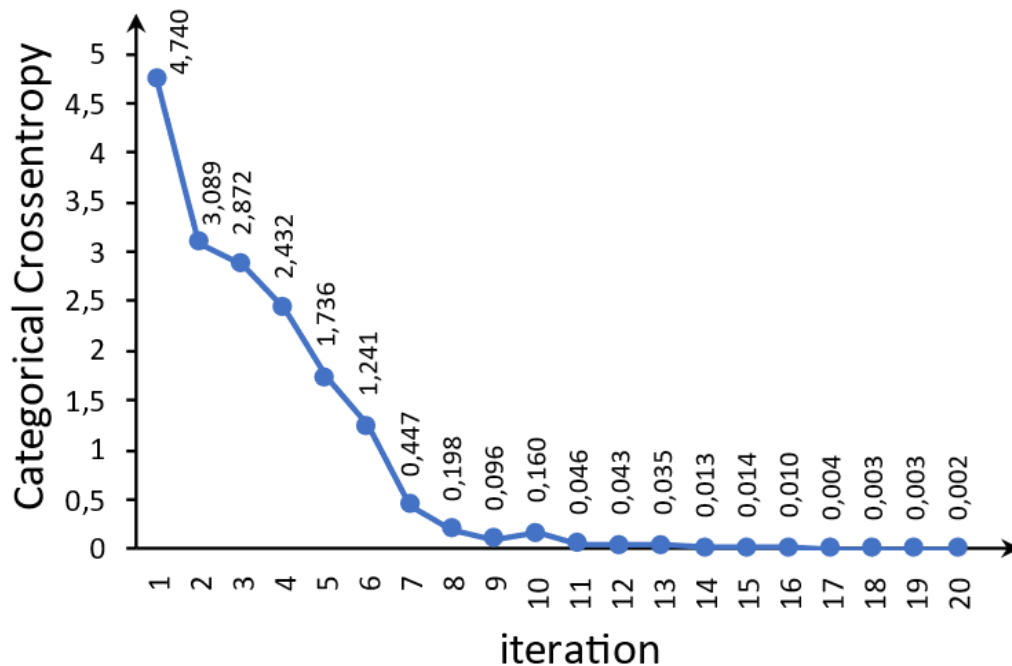


Figure 4: Dependence of categorical entropy on the iterations count for a convolutional autoencoder

Figure 5 shows the relationship between the categorical entropy and the “convolutional - pooling/upsampling layer” pairs count for the convolutional autoencoder.

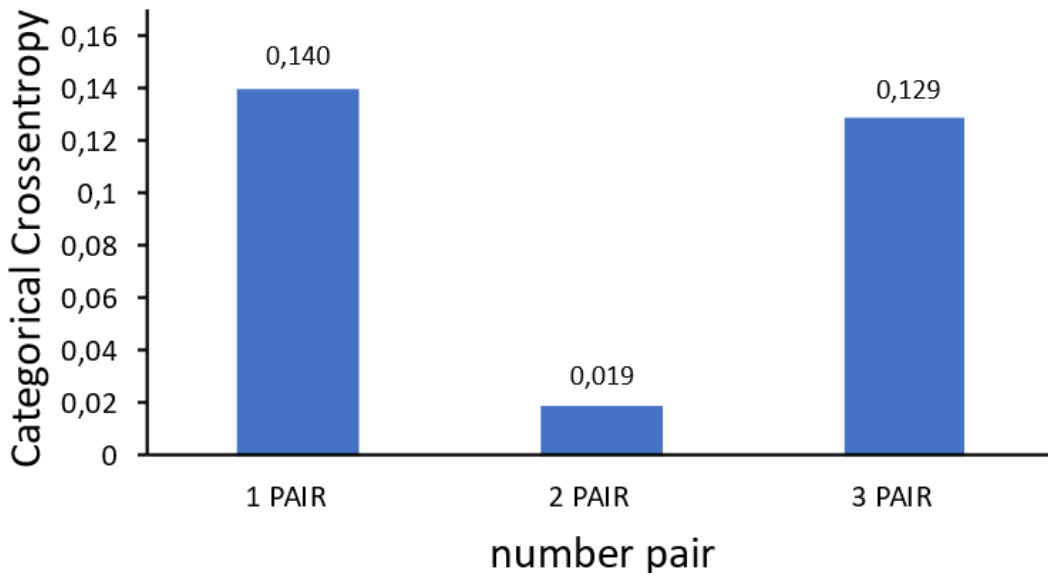


Figure 5: Dependence of categorical entropy on the convolutional – pooling/upsampling layer pairs count for the convolutional autoencoder

Figure 6 shows the impact of the iterations count on the loss (measured by categorical entropy) for the deep Boltzmann machine model.

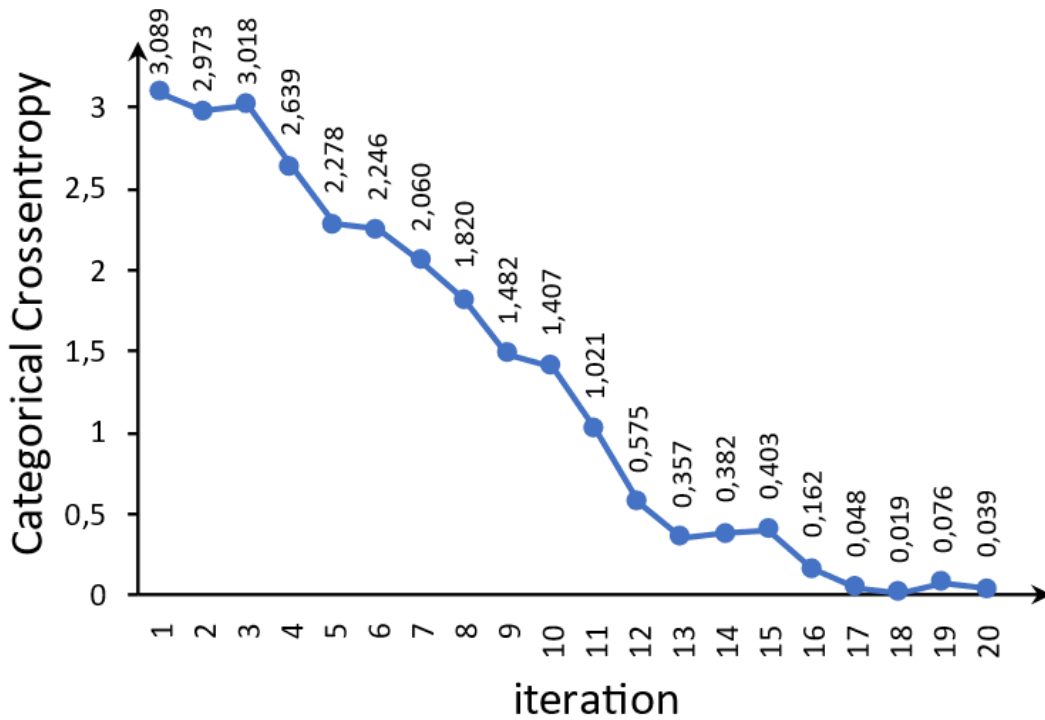


Figure 6: Dependence of categorical entropy on the iterations count for a deep Boltzmann machine model

Figure 7 shows the effect of the RBM count on the loss (measured by categorical entropy) for the deep Boltzmann machine model.

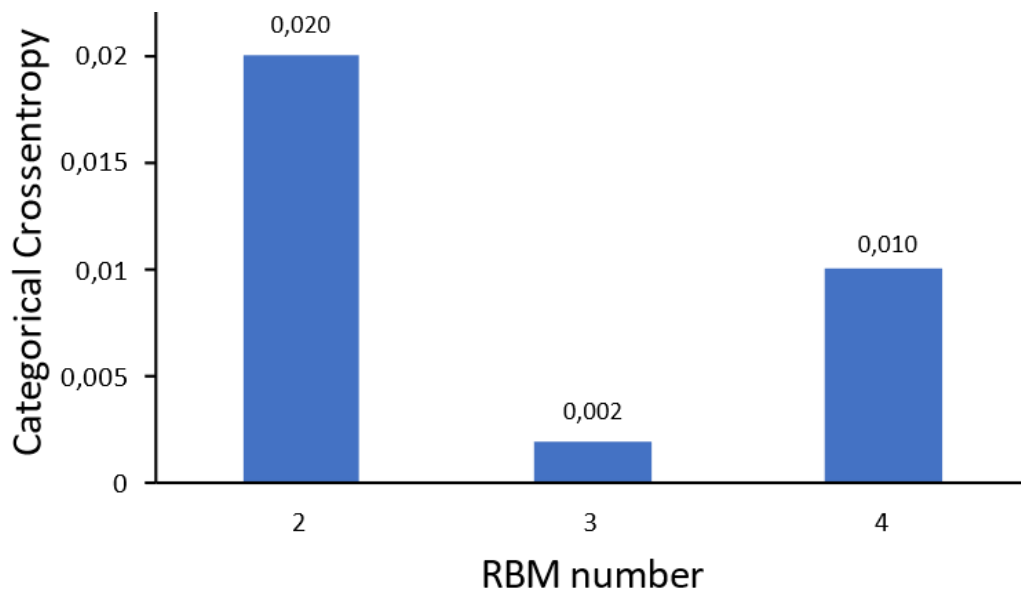


Figure 7: Dependence of categorical entropy on the RBM count for a deep Boltzmann machine model

According to the numerical research results, the following recommendations can be made:

- The minimum number of iterations for a convolutional autoencoder is 11 (Fig. 4).
- The optimal “convolutional – pooling/upsampling layer” pairs count for a convolutional autoencoder are 2 (Fig. 5).
- The minimum iterations count for a deep Boltzmann machine is 17 (Fig. 6, Fig. 7).
- The best RBM count for a deep Boltzmann machine is 3 (Fig. 7).

The proposed method improves the speed and accuracy of classifying the state of chicken eggs, thereby enhancing the quality of egg inspection during incubation (ovoscopy).

7. Conclusions

1. Different image classification methods were explored to enhance the accuracy of identifying the condition of chicken eggs; deep associative neural networks are the most effective method currently available.
2. The model of associative autoencoder offers several advantages compared to the traditional version. It pre-resizes the input image, determines the "convolutional - pooling/upsampling layer" pairs count empirically based on the image size to improve classification accuracy, and calculates the planes count by dividing the cells count in the layer of input by two to the power of the double the number of pairs, ensuring the total cells count in the layer remains constant after pooling/upsampling. This reduces the layer planes size by half in width and height, automating the process of determining the model's layer architecture.
3. The modified deep Boltzmann machine model offers several advantages over the traditional model. First, the input image is pre-resized, which is beneficial for processing. Second, the number of limited Boltzmann machines is determined empirically, leading to improved classification accuracy. Third, the neurons count in the hidden layers is set to double the neurons count in the visible layer, in accordance with the Kolmogorov theorem. This automated approach simplifies the determination of the model's layer structure.
4. The developed method uses deep associative neural networks for intelligent classification of the state of chicken eggs. This method can be applied in intelligent systems to classify the state of eggs.

The proposed method improves the speed and accuracy of classifying the state of chicken eggs, thereby enhancing the quality of egg inspection during incubation (ovoscopy).

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

- [1] C. Yeo, H. Park, K. Lee, C. Song, Avian Embryo Monitoring During Incubation using Multi-Channel Diffuse Speckle Contrast Analysis, *Biomed. Opt. Express*, volume 7 (1), 2016.
- [2] M. Hashemzadeh, N. Farajzadeh, A Machine Vision System for Detecting Fertile Eggs in the Incubation Industry. *Intl. Journal of Computational Intelligence Systems* (2016) 850–862. doi:10.1080/18756891.2016.1237185.
- [3] S.-Y. Tsai, C.-H. Li, C.-C. Jeng, C.-W. Cheng, Quality Assessment during Incubation using Image Processing. *Sensors* 20, 5951 (2020). doi:10.3390/s20205951.
- [4] H. Yu, G. Wang, Z. Zhao, H. Wang, Z. Wang, Chicken Embryo Fertility Detection Based on PPG and Convolutional Neural Network, *Infrared Physics & Technology* 103, 103075 (2019). doi:10.1016/j.infrared.2019.103075.
- [5] E. Fedorov, O. Nechyporenko, T. Utkina, Forecast method for natural language constructions based on a modified gated recursive block, *CEUR Workshop Proceedings*, volume 2604, 2020, pp.199–214.
- [6] E. Fedorov, V. Lukashenko, V. Patrushev, A. Lukashenko, K. Rudakov, S. Mitsenko, The method of intelligent image processing based on a three-channel purely convolutional neural network, *CEUR Workshop Proceedings*, volume 2255, 2021, pp. 336–351.
- [7] G. Shlomchak, G. Shvachych, B. Moroz, E. Fedorov, D. Kozenkov, Automated control of temperature regimes of alloyed steel products based on multiprocessors computing systems, *Metalurgija*, volume 58(3-4), 2019, pp. 299–302.
- [8] Berezhsky O., Liashchynskyi P., Pitsun O., Izonin I. Synthesis of Convolutional Neural Network architectures for biomedical image classification // *Biomedical Signal Processing and Control*. - Vol. 95, Part B, 2024, 106325 <https://doi.org/10.1016/j.bspc.2024.106325>.
- [9] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, *Advances in Neural Information Processing Systems* 25 (2012) 1097–1105.
- [10] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778. doi:10.1109/CVPR.2016.90.
- [11] G. Huang, Zh. Liu, L. van der Maaten, K.Q. Weinberger, Densely Connected Convolutional Networks, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2017). doi:10.1109/CVPR.2017.243.
- [12] Ch. Szegedy, W. Liu, Ya. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going Deeper with Convolutions, 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2015) 1-9. doi:10.1109/CVPR.2015.7298594.
- [13] Ch. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the Inception Architecture for Computer Vision, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2015) 2818-2826. doi:10.1109/CVPR.2016.308.
- [14] Ch. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, *Proceedings of the AAAI Conference on Artificial Intelligence* 31(1). (2016) 1-12. doi:10.1609/aaai.v31i1.11231.
- [15] F. Chollet, Xception: Deep Learning with Depthwise Separable Convolutions, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2017) 1800-1807. doi:10.1109/CVPR.2017.195.

- [16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. An-dreetto, H. Adam, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. (2017) 1–9. doi:10.48550/arXiv.1704.04861.
- [17] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-Ch. Chen, MobileNetV2: Inverted Residuals and Linear Bottlenecks. In: 2018 IEEE Conference on Computer Vision and Pat-tern Recognition (CVPR), 2018, pp. 4510-4520. doi:10.1109/CVPR.2018.00474.
- [18] L. Geng, Yu. Hu, Z. Xiao, J. Xi, Fertility Detection of Hatching Eggs Based on a Convolutional Neural Network. Applied Sciences 9 (7), 1408, 2019. doi:10.3390/app9071408.
- [19] A. Dosovitskiy, L. Beyer, A. Kolesnikov and others, An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In: 9th International Conference on Learning Representations, 2021, pp. 1–22. doi:10.48550/arXiv.2010.11929.
- [20] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, H. Jégou, Training Data-Efficient Image Transformers & Distillation through Attention, Proceedings of Machine Learning Research. (2020) 1–22. doi:10.48550/arXiv.2012.12877.
- [21] D. Zhou, B. Kang, X. Jin, L. Yang, X. Lian, Z. Jiang, Q. Hou, J. Feng, DeepViT: To-wards Deeper Vision Transformer. (2021) 1-21. doi:10.48550/arXiv.2103.11886.
- [22] H.Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, H. Jégou, Going Deeper with Image Transformers, 2021 IEEE/CVF International Conference on Computer Vision (ICCV). (2021) 1-30. doi:10.1109/ICCV48922.2021.00010.
- [23] M.A. Elaziz, A. Dahou, A.O. Aseeri, A.A. Ewees, M.A.A. Al-qaness, R.A. Ibrahim. Cross vision transformer with enhanced Growth Optimizer for breast cancer detection in IoMT environment // Computational Biology and Chemistry. - Vol. 111, 2024, 108110 <https://doi.org/10.1016/j.compbiolchem.2024.108110>.
- [24] A. Hassani, S. Walton, N. Shah, A. Abuduweili, J Li, H. Shi, Escaping the Big Data Paradigm with Compact Transformers. (2021) 1-16. doi: 10.48550/arXiv.2104.05704.
- [25] B. Heo, S. Yun, D. Han, S. Chun, J. Choe, S. J. Oh, Rethinking Spatial Dimensions of Vision Transformers, IEEE International Conference on Computer Vision. (2021) 11916-11926. doi:10.1109/ICCV48922.2021.01172
- [26] B. Li, J. Wang, Zh. Guo, Yu. Li. Automatic detection of schizophrenia based on spatial-temporal feature mapping and LeViT with EEG signals // Expert Systems with Applications. - Vol. 224, 2023, 119969 <https://doi.org/10.1016/j.eswa.2023.119969>.
- [27] L. Yang, H. Wang, W. Meng, H. Pan. CvT-UNet: A weld pool segmentation method integrating a CNN and a transformer // Heliyon. - Vol. 10, Issue 15, 2024, e34738 <https://doi.org/10.1016/j.heliyon.2024.e34738>.
- [28] S. Mehta, M. Rastegari, Light-weight, General-purpose, and Mobile-friendly Vision Transformer, International Conference on Learning Representations. (2021) 1-18. doi:10.48550/arXiv.2110.02178.
- [29] S.S. Shekhawat, S. Shringi, H. Sharma, Twitter sentiment analysis using hybrid Spider Monkey optimization method. Evolutionary Intelligence, volume 14, 2021, pp. 1307–1316. doi:10.1007/s12065-019-00334-2.
- [30] P. Park, P.Di Marco, H. Shin, J. Bang, Fault detection and diagnosis using combined autoencoder and long short-term memory network, Sensors, volume 19, 2019, pp.1-17. doi:10.3390/s19214612.
- [31] J. Shang, J. Yu, A residual autoencoder-based transformer for fault detection of multivariate processes, Applied Soft Computing, volume 163, 2024, pp. 1-16. doi:10.1016/j.asoc.2024.111896.
- [32] J. Xu, Z. Cai, Gaussian mixture deep dynamic latent variable model with application to soft sensing for multimode industrial processes, Appl. Soft Comput, volume 114, 2022, pp.1-14. doi:10.1016/j.asoc.2021.108092.
- [33] R. Xie, N.M. Jan, K. Hao, L. Chen, B. Huang, Supervised variational autoencoders for soft sensor modeling with missing data, IEEE Trans. Ind. Inform, volume 16, 2019, pp. 2820–2828. doi:10.1109/TII.2019.2951622.

- [34] M. Vera, L.R. Vega, P. Piantanida, Information flow in Deep Restricted Boltzmann Machines: An analysis of mutual information between inputs and outputs, *Neurocomputing*, volume 507, 2022, pp. 235-246. doi:10.1016/j.neucom.2022.08.014.
- [35] T. Neskrodieva, E. Fedorov, M. Chychuzhko, V. Chychuzhko, Metaheuristic method for searching quasi-optimal route based on the ant algorithm and annealing simulation, *Radioelectronic and Computer Systems*, volume 1, 2022, pp. 92–102. doi:10.32620/reks.2022.1.07.
- [37] A. Karapetyan, E. Fedorov, K. Rudakov, A. Chepynoha, Methods of Parametric Identification Based on Monkey Behavior, III International Scientific Symposium “Intelligent Solutions” (IntSol-2023), Kyiv, Ukraine, 2023, pp. 89-100.
- [36] E. Fedorov, 2024. Chicken benchmark. URL: <https://github.com/fedorovee75/ArticleChicken>.