Applying Convolutional Neural Network for Cancer Disease Diagnosis Based on Gene Expression Data

Sergii Babichev[a,b], Ihor Liakh[c], Vasyl Morokhovych[c], Andrii Honcharuk[d], Anatolii Balanda[d], Oleksandr Zaitsev[d]

[a] *Kherson State University, University street, 27, Kherson, 73000, Ukraine*
[b] *Jan Evangelista Purkyne University in Usti nad Labem, Pasteurova, 15, Usti nad Labem, 400 96, Czech Republic*
[c] *Uzhhorod National University, University street, 14, Uzhhorod, 88000, Ukraine*
[d] *Military Academy named after Eugene Bereznyak, Yria Il'enka street, 81, Kyiv, 04050, Ukraine*

Abstract

Applying deep learning techniques, such as convolutional or recurrent neural networks, to process gene expression data for developing complex disease diagnostic systems is one of modern bioinformatics's current focuses. Deep learning algorithms can identify specific patterns in the hierarchical representation of data and craft distinct functions that allow for precise identification of the subjects being studied. In this paper, we present our research findings on applying a convolutional neural network (CNN) in diagnosing various types of cancer based on gene expression data. The experimental data were sourced from The Cancer Genome Atlas (TCGA) and comprised 3269 samples. These samples can be categorized into nine classes based on the type of cancer. We introduced an ordered search-by-grid algorithm to pinpoint the optimal set of hyperparameters for the CNN. We assessed the model's efficacy using classification quality metrics, considering type I and II errors. Furthermore, we introduced an integrated F1-score index, drawing from the Harrington desirability function. The obtained results demonstrate the high efficacy of our proposed approach in diagnosing cancer based on gene expression data. The simulation results have shown that the single-layer CNN is more efficient for this type of data by all classification quality criteria. The number of correctly identified samples was 955 out of 981. The classification accuracy was 97.3%.

Keywords 1

Gene expression profiles, cancer disease, Harrington desirability function, convolution neural network, classification quality criteria

1. Introduction and literature review

The appropriateness of using deep learning methods for gene expression data processing is determined by the structure of the experimental data and its large volume. Typically, experimental data contains thousands of objects and more than ten thousand attributes. One of the primary advantages of deep learning methods is their ability to process complex and unstructured data. Deep learning algorithms can identify specific patterns in the hierarchical representation of data and formulate functions that allow for high-precision identification of the objects under investigation. Another significant advantage of models based on deep learning methods is their high accuracy and efficiency. Moreover, models based on deep learning can formulate appropriate functions directly from raw data, enabling the discovery of hidden patterns and intricate relationships in the data that are challenging to uncover using traditional methods. Deep learning-based models possess a scalability feature. This means these models can be efficiently scaled to process large volumes of data, benefiting from parallel or distributed computing architectures, significantly accelerating training and inference processes. In

IDDM'2023: 6th International Conference on Informatics & Data-Driven Medicine, November, 17–19, 2023, Bratislava, Slovakia
EMAIL: sergii.babichev@ujep.cz (A 1); ihor.lyah@uzhnu.edu.ua (A 2); vs.mor75@gmail.com (A 3); papandreas1972@gmail.com (A 4); anatol_ssu@ukr.net (A 5); a.zaysev@gmail.com (A 6)
ORCID: 0000-0001-6797-1467 (A 1); 0000-0001-5417-9403 (A 2); 0000-0002-4939-6566 (A 3); 0009-0001-6190-958X (A 4); 0000-0002-3047-3090 (A 5); 0000-0003-2475-3800 (A 6)

the case of gene expression data, the correct application of deep learning methods enhances the effectiveness of diagnostic systems for complex objects due to the higher accuracy in identifying the studied objects on the one hand and increasing objectivity in determining the state of an object through parallel data processing on the other hand. All the above points highlight the relevance of the current research.

At present, there are several deep learning (DL) methods that can be applied to gene expression data to extract hidden patterns and make predictions regarding the state of the respective object [1]. Figure 1 illustrates a block diagram of the most common deep learning methods focused on processing gene expression data and analyzing genomic sequences, as well as possible directions for their application.

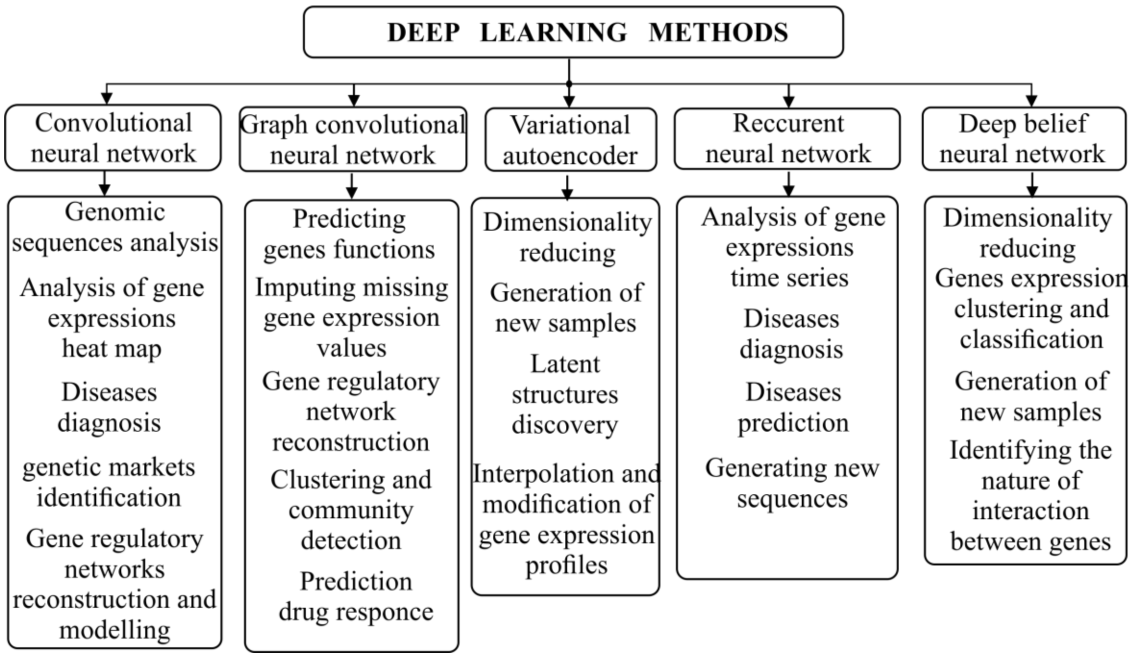
DEEP LEARNING METHODS
Convolutional neural network
Graph convolutional neural network
Variational autoencoder
Reccurent neural network
Deep belief neural network
Genomic sequences analysis
Analysis of gene expressions heat map
Diseases diagnosis
genetic markets identification
Gene regulatory networks reconstruction and modelling
Predicting genes functions
Imputing missing gene expression values
Gene regulatory network reconstruction
Clustering and community detection
Prediction drug responce
Dimensionality reducing
Generation of new samples
Latent structures discovery
Interpolation and modification of gene expression profiles
Analysis of gene expressions time series
Diseases diagnosis
Diseases prediction
Generating new sequences
Dimensionality reducing
Genes expression clustering and classification
Generation of new samples
Identifying the nature of interaction between genes

Figure 1: Block diagram of existing deep learning methods and their application directions for analyzing gene expression data and genomic sequences

As can be seen from Figure 1, the main deep learning (DL) methods can be listed as follows:

1. Convolutional Neural Networks (CNN) [2-4]. They are used for analyzing gene expression data that can be represented as a vector (one-dimensional CNNs) or as images or heat maps (two-dimensional CNNs). Among the advantages of CNNs are their ability to detect hidden dependencies and to form a vector of useful features from genomic data. Depending on the problem formulation, the following application directions for CNNs can be distinguished: analysis of genomic sequences, analysis of gene expression heat maps, disease diagnosis, identification of genetic markers, and reconstruction and modelling of gene regulatory networks (GRN).
2. Recurrent Neural Networks (RNN) [5,6]. They are a powerful tool for analyzing and processing gene expression data, including time series of gene expression values. Typically, when using gene expression data, RNNs are used for solving the following tasks: time series analysis, disease prediction, and generating new sequences.
3. Graph Convolutional Networks (GCN) [7,8]. GCN is a method for processing gene expression data represented in the form of graphs, where genes are represented as nodes and relationships between genes (such as expression interconnections or regulatory interactions) are depicted as edges. In this case, the following application directions for GCNs are possible: predicting gene functions, imputing missing gene expression values, reconstructing gene regulatory networks, clustering, and community detection, and predicting drug responses.
4. Variational Autoencoders (VAE) [9,10]: VAEs are generative models that are capable of discerning patterns of gene interactions based on the low-dimensional representation of gene expression data and generating new samples with similar expression profiles. In the context of

gene expression data processing, VAEs can be applied to address the following tasks: data dimensionality reducing, generation of new samples, the discovery of latent structures, and interpolation and modification of expression profiles.

5. Deep Belief Network (DBN) [11,12]: DBNs consist of several layers of Restricted Boltzmann Machines (RBMs) and can represent the distribution of gene expression data in the form of a hierarchical structure. The potential applications of DBNs in this context may include data dimensionality reduction, clustering and classification, generation of new samples, and identifying the nature of interactions between genes.

Within the framework of current research, the problem of improving the efficiency of diagnostic systems of complex diseases based on gene expression data is being addressed. The solution to this problem involves identifying co-expressed genes in the first stage and classifying objects based on the formed subsets of gene expression profiles in the second stage. This fact limits the number of deep learning (DL) methods that can be applied to solve the stated problem. For instance, classifying objects based on gene expression data can be solved using convolutional or recurrent neural networks. In this case, there is a challenge in determining the optimal network structure and hyperparameter vector that govern the network's performance. Identifying subsets of co-expressed gene expression profiles is possible using a deep belief network. Still, in addition to determining the optimal structure and network hyperparameters, there's a challenge in proving its advantage compared to classic gene expression profile clustering algorithms currently used in this domain. Graph convolutional neural networks can also be used in classification systems. However, their application requires a gene regulatory network reconstruction process in the preliminary stage to represent it as a graph. This, in turn, requires identifying a subset of co-expressed gene expression profiles by applying a clustering procedure to the gene expression data. Implementing this process is possible through model hybridization by using different DL methods at relevant data processing stages. This requires thorough research to assess the efficiency of the appropriate method and determine the optimal network structure and hyperparameter vector.

The choice of CNN for gene expression data processing is determined by their ability to automatically and adaptively learn spatial hierarchies of features from input data. CNNs can capture complex patterns and interactions between genes, aiding in tasks such as classification, clustering, and prediction of gene functions or disease associations. This ability to learn and generalize from the data makes CNNs a powerful tool for extracting meaningful insights from gene expression datasets, potentially leading to new biological discoveries and advancements in personalized medicine.

Numerous studies have focused on utilizing CNNs for diagnosing various objects. For instance, in [13], the authors introduced a fault diagnosis model for rolling bearings based on a multi-dimensional input convolutional neural network (MDI-CNN). The model presented by the authors featured multiple input layers. This design enabled them to combine both original and processed signals, leveraging the strengths of the CNN to automatically learn the characteristics of the original signal. This, in turn, enhanced the recognition accuracy and anti-jamming capability. In [14], the authors shared research findings on predicting cancer types using hybrid CNN + BiLSTM models, which analyzed microarray gene expressions. This approach enabled them to distinguish between different forms of cancers. The results they obtained surpassed those of existing CNN and RNN classifiers in terms of classification quality criteria.

However, it's worth noting that while there are certain advantages in this field, the challenge of effectively implementing deep learning techniques for gene expression data remains unresolved. A primary issue is the objective selection of hyperparameters for specific deep learning models, considering the relevant quality criteria. In this study, we build upon previous research on gene expression data processing [15,16] and the use of CNNs for disease diagnosis based on such data [17,18].

The primary objective of our research is to devise a method for determining the optimal list of hyperparameters for CNN when analyzing gene expression data.

2. Convolutional neural network

The general architecture of the multilayer CNN is depicted in Figure 2 [13]. Usually, it includes the following main components:

1. *Input layer*: Accepts input data, which can be represented as a one-dimensional data vector (a vector of gene expression values that define the state of the object) or a two-dimensional matrix (a heatmap of gene expression values, images, etc.). Depending on the type of input data, one-dimensional (1D) or two-dimensional (2D) convolutional layers are formed.
2. *Convolutional layers*: Used to detect local features in the input data. Each convolutional layer consists of a set of filters that perform the convolution operation on the input data. Convolution is the basic operation in CNN. Typically, in the convolutional layer, the feature map of the previous layer is a convolution using convolutional kernels, and the nonlinear activation function creates the output feature map. The computational process in this case can be expressed as follows [13]:

$$X_j^l = f\left(\sum_{i \in M_i} X_i^{l-1} * \omega_{ij}^l + b_j^l\right) \quad (1)$$

where: X_j^l and X_i^{l-1}) are the *j*-th and *i*-th features of the data at levels *l* and *l*-1 respectively; M_i is the set of input feature maps (determined by the filter applied to the input data at the corresponding convolutional level); ω_{ij}^l is the convolutional kernel connecting the *i*-th feature map of input data with the *j*-th feature map at the convolution level *l*; b_j^l is the bias; *f(·)* is a nonlinear activation function, $*$ stands for the convolution operation.

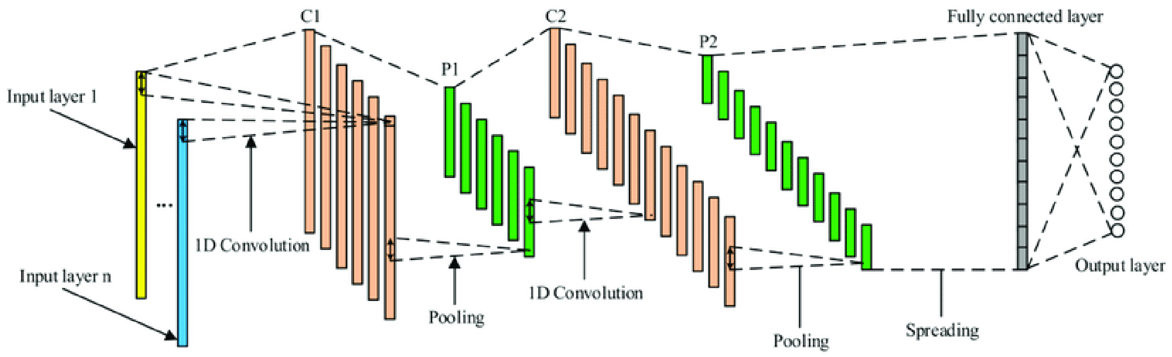

Figure 2: The general architecture of the multilayer convolutional neural network

3. *Pooling layers*: These are used to reduce the spatial dimensions of the feature vector or matrix to decrease the number of parameters. The max pooling layer transforms the data vector or matrix into a single value equal to the maximum value from that region.
4. *Fully Connected Layers*: The data is passed to the fully connected layers after several convolutional and pooling layers. Every neuron in a fully connected layer is connected to every neuron of the previous layer. The fully connected layers are used for classification or regression based on the features obtained. They take the features from the flattened layers and generate an output vector that can be presented as the model's output.
5. *Activation Functions*: After each convolutional layer, an activation function is applied. In most cases, these are nonlinear, allowing the network to detect complex dependencies in the data during the learning process. The most common activation functions are ReLU (Rectified Linear Unit), sigmoid, and hyperbolic tangent (tanh).
6. *Loss Function*: It determines the difference between the predicted and expected values. The derivatives of the loss function are used to update the weights and biases in the network during the backpropagation of the error. This allows the model to assess its accuracy and adjust its weights during training.

As mentioned above, the hyperparameters of CNN determine the network's architecture and training parameters. They are set during the initialization of the network and affect its learning and generalization capabilities. Some of the key hyperparameters of a CNN include:

- Number of Convolutional Layers: It defines the number of layers where convolutional filters detect features in the input data. Having more layers can help the model learn more complex dependencies, but it can also lead to greater complexity, longer training times and overfitting of the network. Overfitting can be determined by evaluating the convergence of accuracy values and the loss function calculated on the training and validation data during network training.
- Size of Convolutional Filters: It determines the size (width and height) of the filters that move over the input data to perform convolution. Larger filters can detect larger patterns but may also lead to increased computational load.
- Number of Filters in Convolutional Layer: It determines the number of filters applied to the input data in each convolutional layer. Each filter generates a feature map corresponding to a specific feature. Typically, the number of filters increases with each subsequent convolutional layer.
- Size of Pooling Window: This refers to the window size (width and height) that moves across the feature map to perform pooling operations.
- Activation Function: This is the function used to introduce non-linearity in the network after each layer. Common activation functions include ReLU (Rectified Linear Unit), Sigmoid, and Hyperbolic Tangent (tanh).
- Number of Fully Connected Layers: This determines the number of fully connected layers that should be added after the convolutional and pooling layers. These layers connect every neuron in one layer to every neuron in the next layer. They are typically used for classification or regression based on the features extracted by the preceding layers.

Within the framework of the current research, the optimal combination of CNN hyperparameters was determined using the ordered empirical grid search method by evaluating all possible combinations of hyperparameter values within predefined ranges.

The implementation of this procedure involves the following stages:

1. Definition of the range of hyperparameter values variation that are subject to optimization.
2. Determination of the metric for evaluating the efficiency of a particular combination of hyperparameter values during their sequential enumeration. Since the current research involved classifying objects based on gene expression data, metrics based on the assessment of type I and type II errors were applied [14]:

- Classification Accuracy – determines the proportion of the total number of samples that are correctly identified:

$$ACC = \frac{TP + TN}{TP + FP + TN + FN} \tag{2}$$

- F1-score is a measure to identify the correctness of the samples distribution into the relevant class and is calculated as the harmonic mean of precision (PR) and recall (RC):

$$F1 = \frac{2 \cdot PR \cdot RC}{PR + RC} \tag{3}$$

where: precision is defined as the probability of correctly identifying samples of the relevant class and is calculated as the ratio of correctly identified samples in the relevant class to the total number of samples in this class:

$$PR = \frac{TP}{TP + FP} \tag{4}$$

Recall is defined as the probability of correctly identifying true positive cases of the relevant class and is calculated as the ratio of correctly identified samples in the relevant class to the total number of samples that really belong to that class:

$$RC = \frac{TP}{TP + FN} \tag{5}$$

In the hereinbefore presented formulas, TP (True Positive) and TN (True Negative) represent the number of objects correctly classified to their respective classes, while FP (False Positive) and FN (False Negative) represent the number of objects misclassified. It's obvious that the maximum value of criteria (2) and (3) is equal to 1, corresponding to the perfect classification. It should be noted that when solving a multi-class problem, criterion (2) defines the overall accuracy of sample classification among classes, while criterion (3) defines the accuracy of sample classification in each class individually.

3. Creation of a grid of all possible combinations of hyperparameters within the range specified in item 1. Each cell of this grid structure represents a unique combination of the model's hyperparameters.
4. For each combination of hyperparameters:

4.1. Construct a neural network model, the architecture and parameters of which correspond to the current combination of hyperparameters.

4.2. Training, validation, and testing of the model.

4.3. Calculation of the quality criteria for sample identification according to formulas (2) and (3).

5. Analysis of the values of the obtained quality criteria for sample classification. Selection of the combination of hyperparameters that corresponds to the maximum values of the sample classification quality criteria.

It should be noted that a drawback of the empirical grid search method is the significant computational time it requires. However, it ensures systematic exploration of the hyperparameter space. It helps to choose the optimal combination for the neural network model, considering both the research objective and the type of experimental data.

3. Experiment, results, and discussion

The modeling process was carried out using gene expression data from patients who were studied for various types of cancer diseases. The data is freely available in The Cancer Genome Atlas (TCGA) [19]. Gene expression data obtained on the Illumina platform [20] was used by applying the method of RNA molecules genomic sequencing, and for each sample, the number of respective genes determining the state of the sample under study was identified. In the initial state, the experimental data contained 3269 samples and 19947 genes. The structure of the experimental data was the following:

- Adrenocortical carcinoma – ACC (79 samples).
- Glioblastoma multiforme – GB (169 samples).
- Sarcoma – SARC (263 samples).
- Lung squamous cell carcinoma – LUSC (502 samples).
- Lung adenocarcinoma – LUAD (541 samples).
- Stomach adenocarcinoma – STAD (415 samples).
- Kidney renal clear cell carcinoma – KIRC (542 samples).
- Brain Lower Grade Glioma – LGG (534 samples).
- Cancer is not identified – Normal (224 samples).

According to the methodology presented in [17,21], in the first stage, the absolute values of the number of genes were transformed into a range more convenient for further processing (Count Per Million – CPM) according to the formula:

$$CPM_{ps} = \frac{count_{ps}}{\sum_{s=1}^{m} count_{ps}} \cdot 10^6 \tag{6}$$

where: $count_{ps}$ is the count of the s-th type of gene corresponding for the p-th sample; m is the total number of different types of genes studied during the experiment performing.

The implementation of this step significantly reduced the range of absolute values variation determining the expression (activity level) of respective genes. In the second stage, data normalization was carried out by applying the $log_2(CPM)$ function to all values. In the third stage, non-expressed genes were removed according to the condition $log_2(CPM) \leq 0$ for all samples under study. The number of genes at this stage was reduced by 682, and the matrix of experimental gene expression data took the form: $E = (3269 \times 19265)$. In the final stage, negative gene expression values were replaced with zeros, representing non-expressed genes for some samples. For proper initialization of CNN filters, the number of gene expression profiles was increased to 19,300 by supplementing with profiles with zero expression.

Figure 3 depicts the flowchart of a 1-D single-layer CNN with relevant hyperparameters at different stages of the neural network's operation. To determine the optimal combination of hyperparameters within the grid search concept, a heuristic search algorithm was proposed. The heuristic functions used were data classification accuracy across all classes (a coarse estimate) and the precision of sample distribution across individual classes by calculating the F1-score for each class (detailed analysis). Considering that when dealing with a large number of classes, analyzing the corresponding F1-score values to choose the optimal alternative from the hyperparameter list can be problematic, an integrated F1-score value was calculated based on the previously obtained values using Harrington's desirability method, which is one of the effective methods for solving multi-criteria problems and is currently successfully used in various fields of scientific research [22].

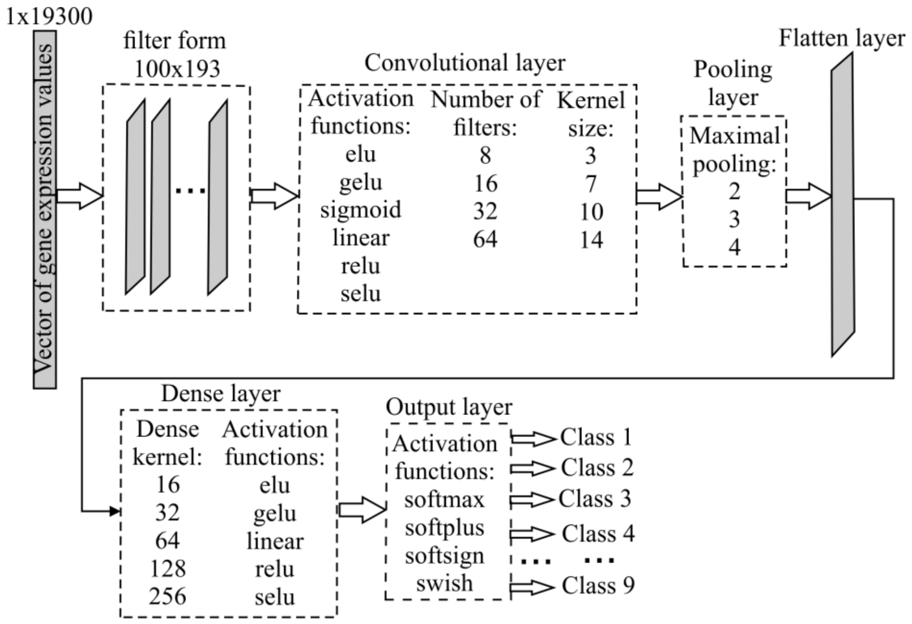
1x19300
Vector of gene expression values
filter form 100x193
Convolutional layer
Activation functions: elu gelu sigmoid linear relu selu
Number of filters: 8 16 32 64
Kernel size: 3 7 10 14
Pooling layer
Maximal pooling: 2 3 4
Flatten layer
Dense layer
Dense kernel: 16 32 64 128 256
Activation functions: elu gelu linear relu selu
Output layer
Activation functions: softmax softplus softsign swish
Class 1
Class 2
Class 3
Class 4
Class 9

Figure 3: Flowchart of a 1-D single-layer CNN for determining the optimal hyperparameter vector of the neural network

The algorithm for implementing this procedure involves the following steps:

Step 1: Initialization.

1.1. Representation of F1-score values in the form of a matrix where rows are classes and columns are hyperparameter values, the combination of which is being studied at this stage.

Step 2. Calculation of private desirabilities.

2.1. Determination of the minimum and maximum values of the F1-score at the corresponding stage of the CNN operation (when applying the corresponding combination of hyperparameters).

2.2. Transformation of F1-score value scales into a linear scale of the dimensionless indicator Y, considering the boundary values of the F1-score determined in the previous step (the value of the parameter Y according to the desirability method varies in the range from -2 to 5). In this case, during the first step, the coefficients of the linear equation are calculated as follows:

$$Y_{min} = a + b \cdot F1_{min} \quad (7)$$
$$Y_{max} = a + b \cdot F1_{max}$$

Then, a direct transformation of F1-score values into Y values is carried out:

$$Y = a + b \cdot F1 \quad (8)$$

2.3. Calculation of private desirabilities d for each F1-score value:

$$d = \exp\left(-\exp\left(-Y\right)\right) \quad (9)$$

Step 3. Calculation of the integrated F1-score value.

3.1. For each column of the matrix obtained in step 2, calculate the integrated F1-score value as the geometric average of all private desirabilities:

$$F1_{int}^{j} = \sqrt[9]{\prod_{i=1}^{9} d_{ij}} \tag{10}$$

where j denotes the corresponding column of the matrix of private desirabilities.

Step 4. Analysis of the obtained results.

4.1. Creation of a diagram showing the dependency of the integrated F1-score value on the corresponding hyperparameter values. Selection of the optimal hyperparameter value corresponding to the maximum of the integrated F1-score.

During the simulation process, the following activation functions were applied to the output layer of neurons: *softmax, softplus, softsign*, and *swish*. For the fully connected inner layer of neurons, the *elu, gelu, linear, relu,* and *selu* activation functions were sequentially applied. Other activation functions applied to this layer showed unsatisfactory results. The following activation functions were applied to the inner convolutional layer: *elu, gelu, sigmoid, linear, relu,* and *selu*. In the initial data preprocessing stage, the data was split into two subsets in a 0.7/0.3 ratio (2288/981 samples). The first subset (2288 samples) was further divided into two subsets in a 0.8/0.2 ratio (1830/458). 1830 samples were used for training the network, 458 for validating the model during its training, and 981 samples were used for testing the model. The model's quality assessment was based on the analysis of the loss function value calculated during the model's validation, the classification accuracy, and the F1-score value calculated when applying the test data. The simulation results for determining the optimal activation function of the neurons' output layer are shown in Figure 4.

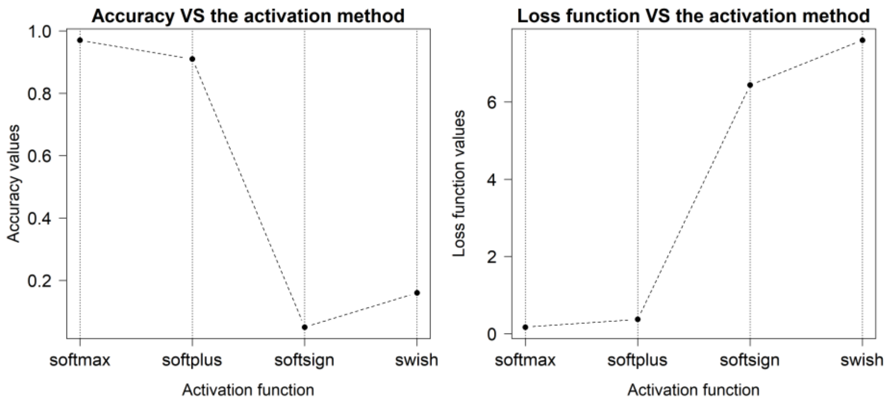

Figure 4: Distribution diagrams of classification quality criteria when determining the optimal activation function for the output layer of neurons in the neural network model (CNN)

As seen from Figure 4, the use of the *softmax* function allows obtaining the best classification results for samples in terms of accuracy, which was calculated on the test data subset, and in terms of the loss function, which was calculated on the validation data. When using the *softplus* function, the classification results are slightly worse. When using other functions, the classification results are unsatisfactory. These conclusions are confirmed by the analysis of F1-score values calculated for each of the nine classes. Due to the clear results obtained, the diagrams showing the dependence of the F1-score values on the type of activation function used are not shown.

In Figure 5, the simulation results are presented concerning determining the optimal activation function for the CNN's neurons for the dense layer. The analysis of the simulation results allows concluding that in terms of sample classification accuracy (Figure 5a) and loss function value (Figure 5b), the optimal activation functions are *elu* and *selu*, which to some extent does not match the results based on the analysis of F1-score values (Figure 5c, d). The analysis of the integrated F1-score criterion values (Figure 5d) allows concluding that the highest values of this criterion correspond to *relu* and *gelu* methods. Slightly lower values are achieved when using the *selu* method. The analysis of the distribution character of the F1-score values for individual clusters (Figure 5c) confirms this conclusion. Thus, based on the analysis of the values of all the criteria, the *selu* method was determined as optimal at this stage of research.

In Figure 6, the simulation results for the selection of the optimal activation function for the neurons of the convolutional layer are shown.

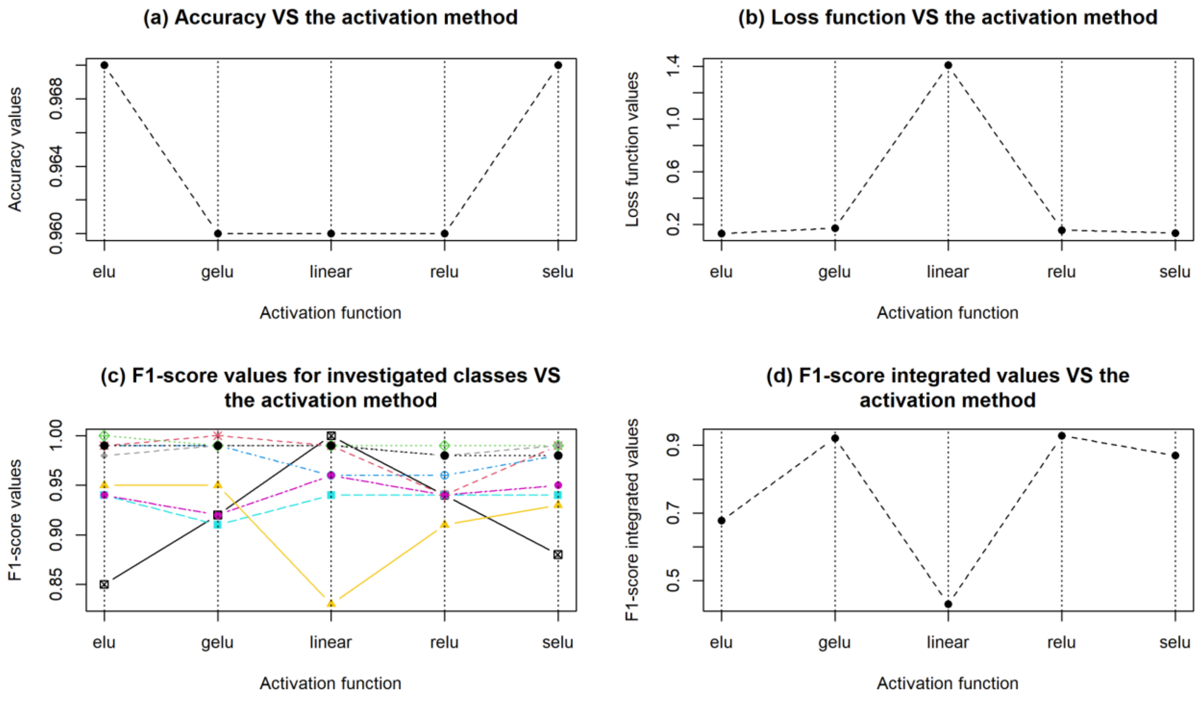

Figure 5: Simulation results regarding determining the optimal activation function for the CNN's dense layer neurons

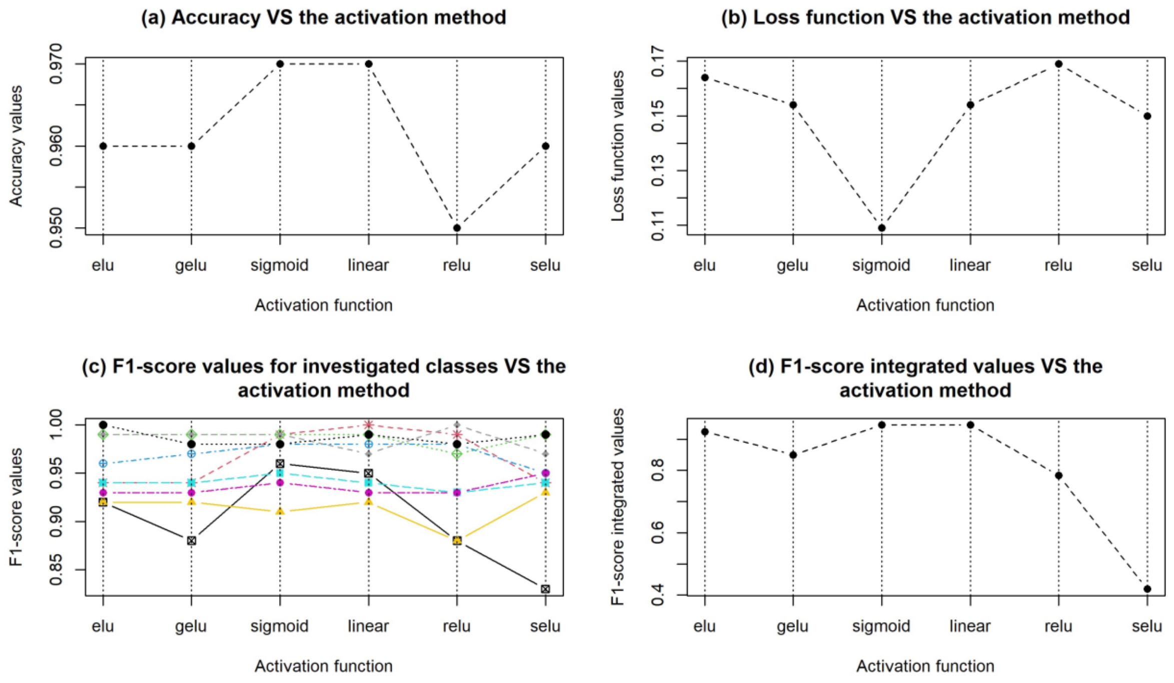

Figure 6: Simulation results for determining the optimal activation function for the neurons of the convolutional layer of the CNN model

As can be seen, in terms of sample classification accuracy and the integrated value of the F1-score, the sigmoidal (sigmoid) and linear (linear) activation functions are optimal. However, in terms of the loss function value, the sigmoidal function is more preferable.

In Figures 7-10, similar results are depicted for determining other types of CNN's optimal hyperparameters. The analysis of the obtained results suggests that in terms of the classification accuracy of the samples (Figure 7a) and the integrated value of the F1-measure (Figure 7d), the optimal value of the hyperparameter maximal pooling could be 2 or 3. However, in terms of the loss function value, 2 corresponds to better results.

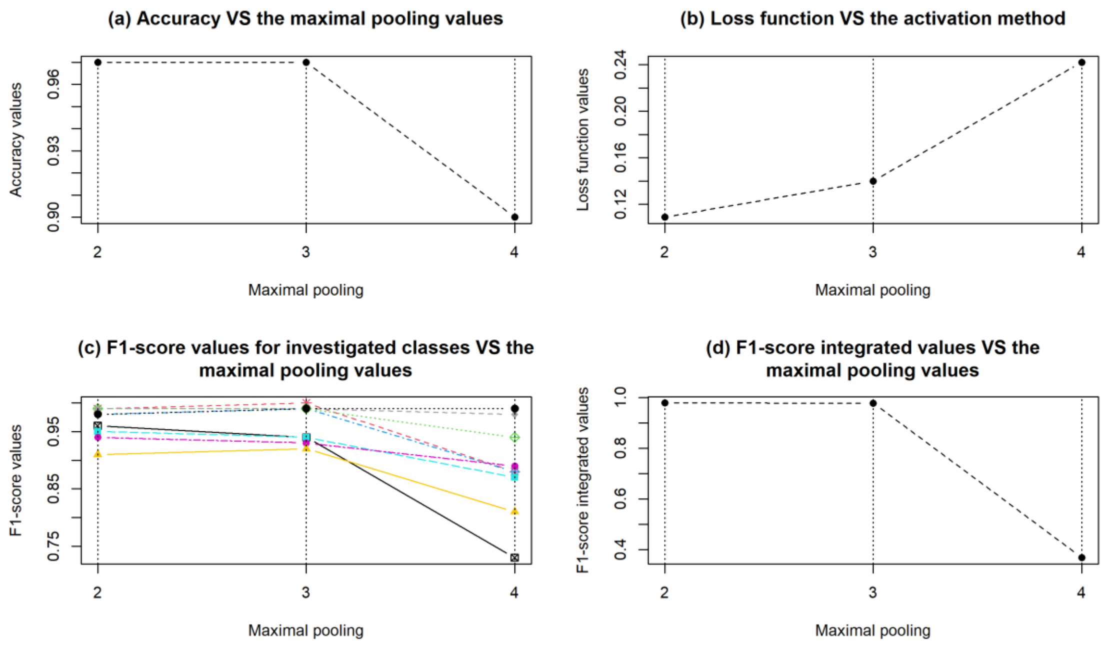

Figure 7: Results of simulation to determine the optimal value of maximal pooling for neurons of the convolutional layer

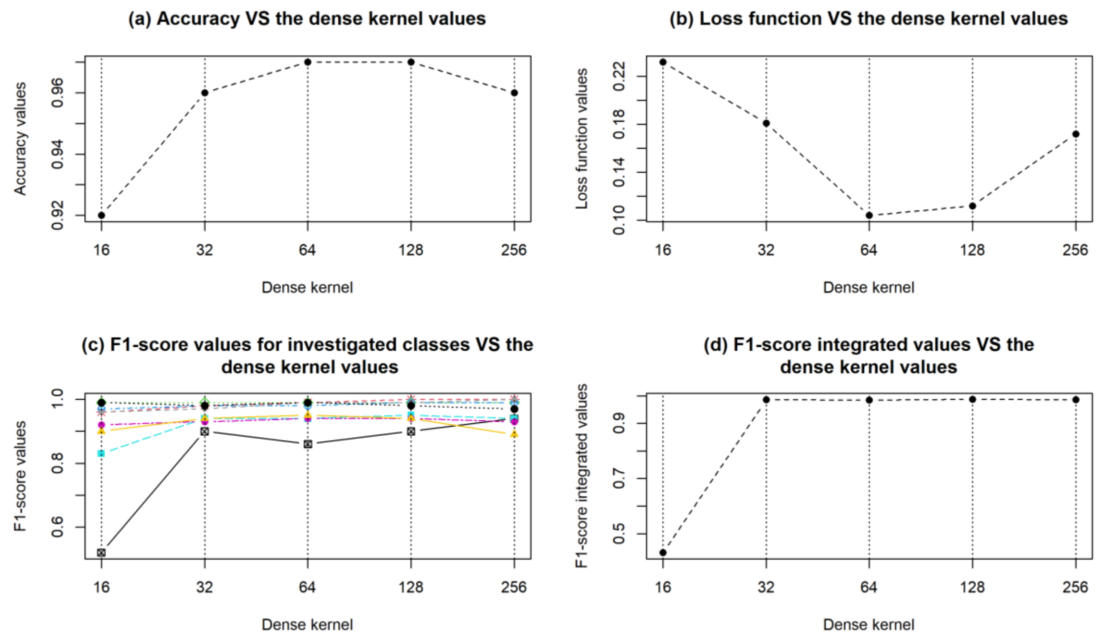

Figure 8: Results of simulation to determine the optimal kernel size for the dense layer neurons (dense kernel)

The analysis of the diagrams, which depict the dependence of classification quality criteria on the size of the kernel function for neurons of the dense layer (dense kernel), shown in Figure 8, indicates that choosing the optimal kernel size based on the F1-score is problematic since the results are almost indistinguishable for values 32, 64, 128, and 256 (Figure 8d). In terms of classification accuracy, optimal values are 64 and 128 (Figure 8a). In terms of the loss function value, 64, in this case, appears to be more attractive (Figure 8b).

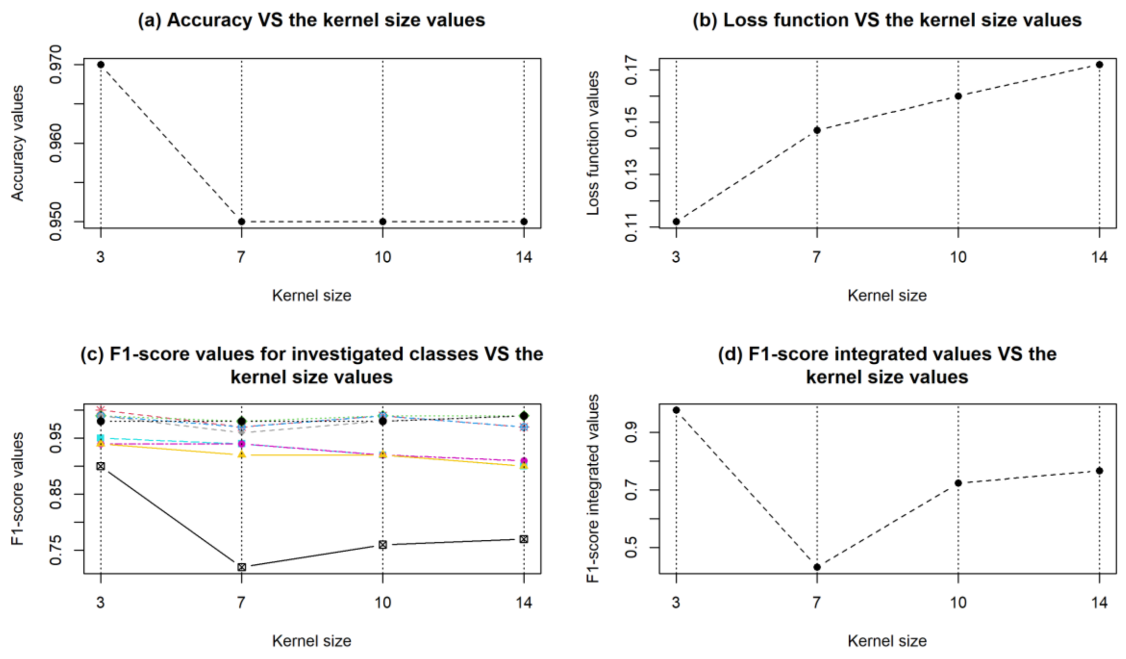

Figure 9: Results of simulation to determine the optimal kernel size for the convolutional layer neurons (kernel size)

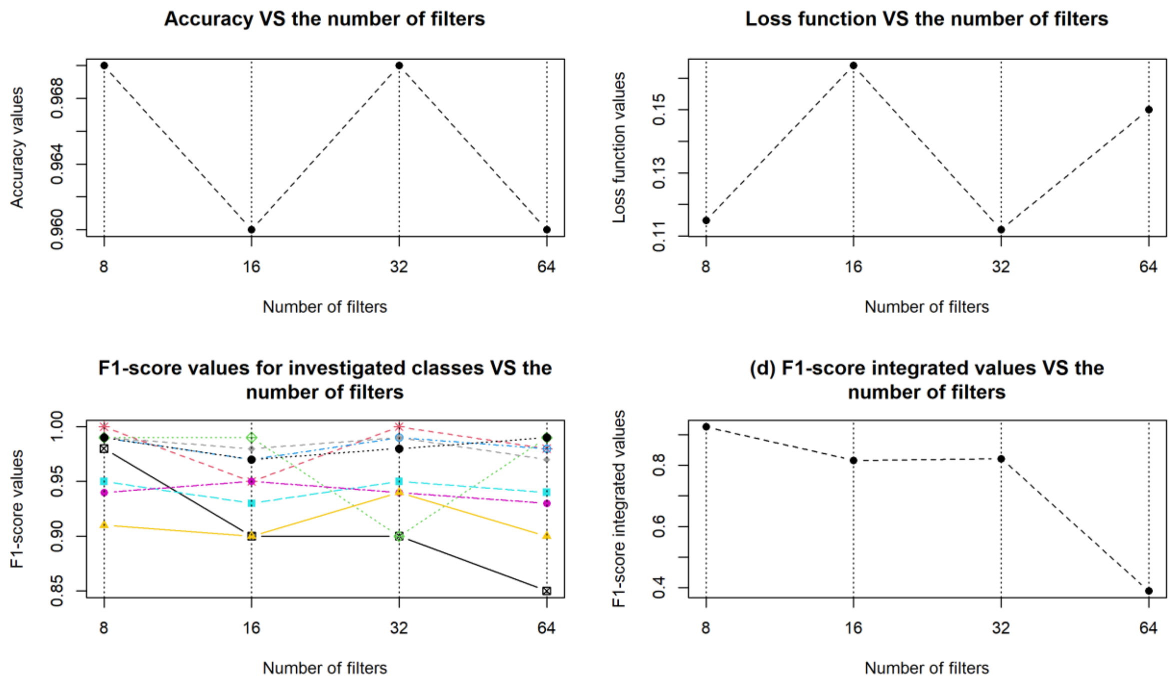

Figure 10: Results of simulation to determine the optimal number of filters for the convolutional layer neurons

The analysis of simulation results, as shown in Figure 9, suggests that, by all quality criteria of sample classification, the optimal kernel size for the functions of convolutional layer neurons is 3.

Analysis of the distribution diagrams of quality criteria for classification at different values of the number of convolutional layer filters (Figure 10) indicates that according to the accuracy criterion, the optimal options are 8 and 32 filters. The loss value when using 32 filters is slightly less. The integrated F1-score value suggests a slightly higher attractiveness of using eight filters. In this case, a compromise decision was made to use 32 filters, as reducing the number of filters can lead to a decrease in the sensitivity of the CNN, which is not acceptable within the current research framework.

The obtained simulation results allowed us to compile a list of optimal hyperparameters for a 1D single-layer CNN, the values of which are presented in Table 1. The next step in the simulation process is to compare the efficiency of a 1D single-layer, two-layer, and three-layer CNN when using the hyperparameters determined in the previous simulation stage. On the first convolutional layer, a filter (100×193) was applied to the gene expression value vector, on the second (50×386), and on the third (25×772).

Table 1
Optimal hyperparameters values for a 1D single-layer CNN

Number of filters	Kernel size	Dense kernel	Maximal pooling	Activation function of convolutional layer	Activation function of dense layer	Activation function of output layer
32	3	64	2	sigmoid	selu	softmax

The simulation results are presented in Figure 11. The training time for the model was almost the same in all cases, approximately 83 seconds.

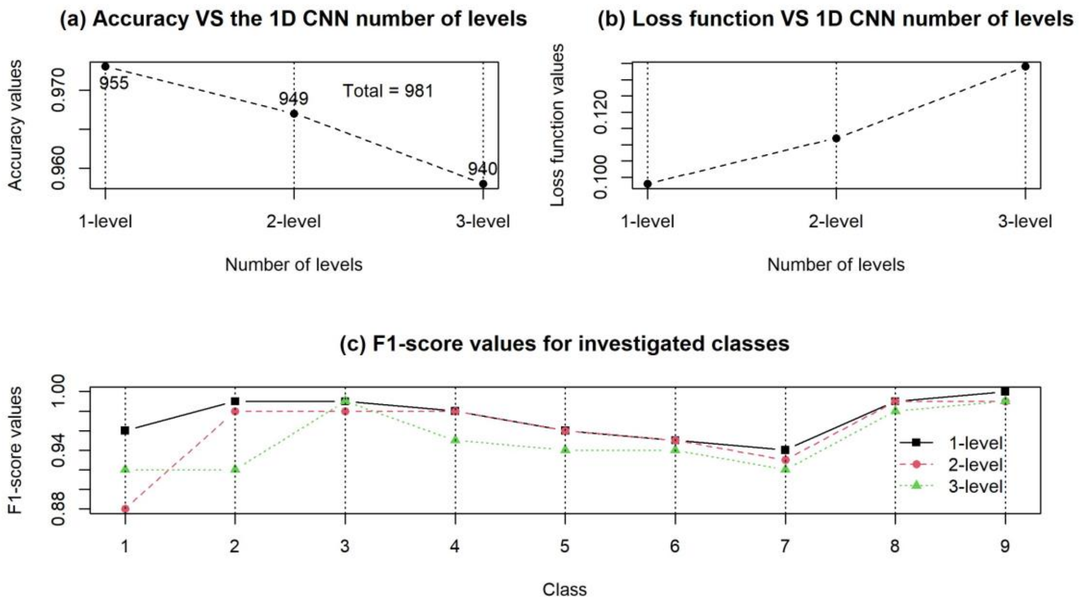

Figure 11: Simulation results for determining the number of convolutional layers in 1D CNN

Figure 11a also displays the total number of samples that made up the test data subset and the number of samples correctly identified in each case. An analysis of the results allows us to conclude that, by all criteria, the single-layer CNN is more efficient for this type of data. The number of correctly identified samples is 955 out of 981. The classification accuracy is 97.3%. The F1-score values, calculated for all classes using a single-layer network, are also higher than the two- and three-layer networks. The loss function value, computed using the validation data in this case, is also the lowest.

4. Conclusions

In the paper, we have presented the research results on applying a convolutional neural network (CNN) for the classification of samples based on gene expression data. Various architectures of one-dimensional CNNs are considered. As optimization hyperparameters, the following were studied: activation functions of output, convolutional and dense layers, the number of filters, the kernel size of neurons of convolutional and dense layers, and max pooling. As criteria for evaluating the quality of the corresponding model, the classification accuracy of samples (Accuracy), the loss function value calculated on the data subset for model validation, and the F1-score, which includes errors of the first and second kind (sensitivity and specificity) as components and is one of the effective criteria for the quality of sample distribution into separate classes. An integrated F1-score criterion has been proposed, the calculation of which involves applying Harrington's desirability function to partial F1-score values calculated for individual classes. The simulation results regarding applying different CNN architectures for the classification of gene expression data are presented. The experimental data used were gene expression data from patients who were studied for various types of cancer and contained eight classes of samples taken from patients with the corresponding type of cancer. The ninth group included samples from patients in whom no cancer was detected. According to the simulation results, a single-layer CNN demonstrated higher effectiveness across a set of quality criteria. When using the ordered grid search algorithm, the following hyperparameters were identified as optimal: *softmax* activation function for the output layer neurons, *selu* activation function for dense layer neurons, *sigmoid* activation function for convolutional layer neurons, max pooling of 2, kernel size for dense layer neurons of 64, kernel size for convolutional layer neurons of 3, and the number of filters in the convolutional layer of 32.

Future directions for the authors' research include exploring alternative algorithms for neural network hyperparameter optimization and incorporating other deep learning methods within the complex object diagnosis system.

5. References

[1] E.M. Nikolados, D.A. Oyarzún. Deep learning for optimization of protein expression. Current Opinion in Biotechnology, 2023, vol. 81, art. no. 102941. DOI: 10.1016/j.copbio.2023.102941

[2] E. Mustafa, E.K. Jadoon, S. Khaliq-uz-Zaman, M.A. Humayun, M. Maray. An Ensembled Framework for Human Breast Cancer Survivability Prediction Using Deep Learning. Diagnostics, vol. 13(10), art. no. 1688. DOI: 10.3390/diagnostics13101688.

[3] G. Mao, Z. Pang, K. Zuo, J. Liu. Gene Regulatory Network Inference Using Convolutional Neural Networks from scRNA-seq Data. Journal of Computational Biology, 2023, vol. 30(5), pp. 619-631. DOI: 10.1089/cmb.2022.0355.

[4] A. Kaur, A.P.S. Chauhan, A.K. Aggarwal. Prediction of Enhancers in DNA Sequence Data using a Hybrid CNN-DLSTM Model. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2023, vol. 20(2), pp. 1327-1336. DOI: 10.1109/TCBB.2022.3167090.

[5] N.P. Kumar, S. Vijayabaskar, L. Murali, K. Ramaswamy. Design of optimal Elman Recurrent Neural Network based prediction approach for biofuel production. Scientific Reports, 2023, vol. 13(1), art. no. 8565. DOI: 10.1038/s41598-023-34764-x.

[6] R. Jain, A. Jain, E. Mauro, K. LeShane, D. Densmore. ICOR: improving codon optimization with recurrent neural networks. BMC Bioinformatics, 2023, vol. 24(1), art. no. 132. DOI: 10.1186/s12859-023-05246-8.

[7] H. Xu, J. Lin, D. Zhang, F. Mo. Retention time prediction for chromatographic enantioseparation by quantile geometry-enhanced graph neural network. Nature Communications, 2023, vol. 14(1), art. no. 3095. DOI: 10.1038/s41467-023-38853-3.

[8] Z. Wu, J. Wang, H. Du, et al. Chemistry-intuitive explanation of graph neural networks for molecular property prediction with substructure masking. Nature Communications, 2023, vol. 14(1), art. no. 2585. DOI: 10.1038/s41467-023-38192-3.

[9] L. Comanducci, D. Gioiosa, M. Zanoni, F. Antonacci, A. Sarti. Variational Autoencoders for chord sequence generation conditioned on Western harmonic music complexity. Eurasip Journal on

Audio, Speech, and Music Processing, 2023, vol. 2023(1), art. no. 24. DOI: 10.1186/s13636-023-00288-5.

[10] G. Nikolentzos, M. Vazirgiannis, C. Xypolopoulos, M. Lingman, E.G. Brandt. Synthetic electronic health records generated with variational graph autoencoders. Digital Medicine, 2023, vol. 6(1), art. no. 83. DOI: 10.1038/s41746-023-00822-x.

[11] X. Lu, P. Li. Research on gearbox temperature field image fault diagnosis method based on transfer learning and deep belief network. Scientific Reports, 2023, vol. 13(1), art. no. 6664. DOI: 10.1038/s41598-023-33858-w

[12] D. Ma, P. Jiang, L. Shu, Y. Qiu, Y. Zhang, S. Geng. DBN-based online identification of porosity regions during laser welding of aluminum alloys using coherent optical diagnosis. Optics and Laser Technology, 2023, vol. 165, art. no. 109597. DOI: 10.1016/j.optlastec.2023.109597.

[13] T. Zan, H. Wang, M. Wang, et al. Application of Multi-Dimension Input Convolutional Neural Network in Fault Diagnosis of Rolling Bearings. Applied Sciences, 2019, vol. 9, art no. 2690. DOI: 10.3390/app9132690.

[14] P. Metipatil, P. Bhuvaneshwari, S.M. Basha, S.S. Patil. An Efficient Framework for Predicting Cancer Type Based on Microarray Gene Expressions Using CNN-BiLSTM Technique. SN Computer Science, 2023, vol. 4(4), art. no. 381. DOI: 10.1007/s42979-023-01774-5.

[15] S. Babichev, J. Krejci, J. Bicanek, V. Lytvynenko. Gene expression sequences clustering based on the internal and external clustering quality criteria. Proceedings of the 12th International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT 2017, 2017, vol. 1, art. no. 8098744, pp. 91-94. DOI: 10.1109/STC-CSIT.2017.8098744.

[16] S. Babichev, V. Osypenko, V. Lytvynenko, M. Voronenko, M. Korobchynskyi. Comparison Analysis of Biclustering Algorithms with the use of Artificial Data and Gene Expression Profiles. 2018 IEEE 38th International Conference on Electronics and Nanotechnology, ELNANO 2018 - Proceedings, 2018, art. no. 8477439, pp. 298-304. DOI: 10.1109/ELNANO.2018.8477439

[17] S. Babichev, L. Yasinska-Damri, I. Liakh, J. Škvor. Hybrid Inductive Model of Differentially and Co-Expressed Gene Expression Profile Extraction Based on the Joint Use of Clustering Technique and Convolutional Neural Network. Applied Sciences (Switzerland), 2022, vol. 12(22), art. no. 11795, DOI: 10.3390/app122211795.

[18] L. Yasinska-Damri, S. Babichev, B. Durnyak, T. Goncharenko. Application of Convolutional Neural Network for Gene Expression Data Classification. Lecture Notes on Data Engineering and Communications Technologies, 2023, vol. 149, pp. 3-24. DOI: 10.1007/978-3-031-16203-9_1.

[19] The Cancer Genome Atlas Program (TCGA). National Cancer Institute. Center for Cancer Genomics. URL: https://www.cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcga

[20] Illumina. URL: https://www.illumina.com/

[21] T. Girke. R & Bioconductor Manual. Institute for Integrative Genome Biology. URL: http://manuals.bioinformatics.ucr.edu/home/R_BioCondManual

[22] V. Koilo. Financial performance under stress: the case of the Norwegian maritime cluster. Public and Municipal Finance, 2019, vol. 8(1), pp. 54-72. DOI: 10.21511/pmf.08(1).2019.05