Received 27 March 2025, accepted 23 August 2026, date of publication 30 September 2026
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.25.3.4841

Development of a Modified Siamese Neural Network for Object Recognition

IEVGEN FEDORCHENKO, ANDRII OLIINYK, TETIANA FEDORONCHAK, ALEXANDER STEPANENKO, YEVGENY GOFMAN, VLADYSLAV PRYMAKOV

Department of Software Tools, National University "Zaporizhzhia Polytechnic",
64 Zhukovskoho str., Zaporizhzhia, Ukraine, 69011

Corresponding author: Ievgen Fedorchenko (e-mail: evg.fedorchenko@gmail.com).

ABSTRACT The article explores methods for training Siamese neural network for classification and retrieval of similar objects using embeddings. The study focuses on the development and analysis of the effectiveness of various loss functions, including contrastive and triplet loss, as well as training-sample selection strategies such as Batch Hard. The models were implemented in the PyTorch framework, ensuring their flexibility and scalability. The MNIST dataset, a standard benchmark for image-recognition tasks, was used for testing. The scientific novelty of this study lies in applying of a genetic algorithm for the automatic optimization of the margin parameter in loss functions. Traditionally, this parameter is selected manually or determined empirically, which may negatively affect the model's training quality. The proposed approach enables adaptive tuning of the model to the characteristics of each data batch, improving the model's generalization ability and enhancing classification accuracy. Experimental results confirmed the effectiveness of the developed approaches. The best performance was demonstrated by the model with the triplet loss function and the Batch Hard strategy, indicating the feasibility of using more advanced sample selection strategies to improve neural network performance. Overall, the study confirms the efficiency of enhanced Siamese network training methods for object - recognition and matching tasks.

KEYWORDS Siamese neural networks; convolutional networks; contrastive loss function; triplet loss function; online pair and triplet selection strategies; DrLIM; PyTorch; MNIST.

I. INTRODUCTION

With the rapid advancement of computer science and machine learning, methods for data comparison and similarity search are becoming increasingly relevant. Traditionally, metrics such as Euclidean and cosine distance, as well as Pearson and Spearman correlation coefficients, are used for this purpose [1–3]. These methods work well when the compared data share the same type and scale. However, they are ineffective in cases where objects with different characteristics or complex data structures need to be compared.

One of the key challenges in this field is the comparison of complex objects, such as handwritten signature images for identity verification. To address such tasks, the Siamese neural network was introduced in 1994 as an architecture capable of learning to identify similarities between two input images [4]. Since then, Siamese networks have been widely applied in pattern recognition, biometric authentication, and similarity search.

The main objective of this study is to design, implement, and evaluate the effectiveness of Siamese neural networks for classification and similar image retrieval tasks. The research includes an analysis of the impact of different architectures, loss functions, and pair/triplet selection methods on training accuracy and speed.

II. ANALYSIS OF THE SUBJECT AREA

A. SIAMESE NEURAL NETWORK ARCHITECTURE

The Siamese neural network was first introduced in 1994 as a method for detecting forged signatures [4]. However, its concept had predecessors, including artificial neural networks used for fingerprint recognition. The core idea of a Siamese network is to compare two objects and determine their similarity, making it a popular choice for biometric identification and image classification tasks.

This architecture is based on traditional artificial neural networks, which consist of multiple layers of neurons that transmit information from the input layer to the output layer [5]. In the classical feedforward case, each neuron receives input data, weights it according to its parameters, and then passes the result forward until the final layer generates the final output. These networks are trained by comparing their predictions with the correct answers, calculating the error, and

adjusting the weights using the backpropagation algorithm.

A Siamese network is based on two identical neural networks that operate in parallel and process two different input objects [6, 7]. These networks have the same architecture and shared weights, allowing them to extract similar features from both inputs [8]. After processing by these parallel networks, the extracted representations are compared using a similarity measure, such as Euclidean or cosine distance. Based on this comparison, the model determines the similarity of the two objects.

The key advantage of this approach is its versatility: Siamese neural networks perform effectively even in cases where traditional similarity methods fail, such as when comparing images or other complex structured data. Due to their flexibility, they are widely used in biometric identification, facial recognition, handwriting analysis [6], and a wide range of other tasks related to similarity search.

B. DATA DIMENSIONALITY REDUCTION BY LEARNING AN INVARIANT MAPPING

Modern technologies handle large volumes of high-dimensional data, making processing and analysis more challenging. Therefore, dimensionality reduction is an essential step in simplifying data representation while preserving its fundamental structure and relationships between objects [9]. The key idea of this approach is to map similar objects in the input space to neighboring points in a new lower-dimensional space. However, many classical methods have limitations: they do not always produce generalizable mappings applicable to new data and often depend on the choice of an initial distance metric.

Another drawback of traditional methods is the tendency to densely cluster points in the low-dimensional space, sometimes leading to incorrect or unsuitable solutions for analysis. In such cases, achieving a more uniform sample distribution is desirable to better preserve data structure. To address these challenges, the Dimensionality Reduction by Learning an Invariant Mapping (DrLIM) method was proposed, enabling the discovery of a nonlinear mapping function that projects data into a reduced space while maintaining its semantic properties [9].

The main idea of DrLIM is to rely only on relationships between neighboring points in the dataset, which can be determined based on prior labeling or domain knowledge [9]. This method does not depend on the choice of a metric in the original space, making it more flexible for complex and nonlinear transformations such as lighting changes or geometric distortions. An essential aspect of the method is its generalization capability: the trained function can be used to map new points that were not present during training, ensuring the stability and coherence of the resulting space.

DrLIM employs an energy-based model that learns the mapping function based on predefined neighborhood relations. Formally, for a family of functions G_W, parameterized by weight vectors W, the goal is to find a representation satisfying condition (1) [9]:

$$D_w(X_1, X_2) = \|G_w(X_1) - G_w(X_2)\|_2 , \quad (1)$$

where the Euclidean distance in the reduced space should reflect the semantic similarity in the original space. No additional constraints are imposed on G_W except differentiability with respect to W.

The learning task is formulated as follows: given a set of input vectors (2) [9]:

$$I = \{X_1, \dots, X_n\},\ X_i \in R^D,\ \forall_i = 1, \dots, n , \quad (2)$$

where n – the number of elements in the dataset, D – the dimensionality of each element. The goal is to find a parametric function (3) [9]:

$$G_W: R^D \rightarrow R^d, d \ll D , \quad (3)$$

which meets the following criteria: relationships between neighboring points in the original space should be reflected in distances within the low-dimensional space, and the mapping should be robust to complex nonlinear transformations. Additionally, the function should work correctly for samples whose neighborhood information was not provided during training.

Thus, the DrLIM method enables the creation of generalizable representations of high-dimensional data in a compact form, preserving its structure and object relationships. This ensures effective handling of new data and makes the approach valuable for a wide range of machine learning tasks.

C. LOSS FUNCTIONS

The contrastive loss function serves as the foundation for training models that distinguish between similar and dissimilar objects. Unlike traditional methods, where losses are computed separately for each instance, the contrastive function operates on pairs of data points. Let Y be a binary label assigned to a given pair: $Y = 0$, if X_1 and X_2 are considered similar, and $Y = =1$, if they are considered different. The parameterized distance function D_w between X_1 and X_2 is defined as Euclidean (1). For brevity, $D_w(X_1, X_2)$ is denoted as D_w. The general form of the loss function $L(W)$ is given by formulas (4–5) [9]:

$$L(W) = \sum_{i=1}^{P} L(W, (Y, X_1, X_2)^i), \quad (4)$$

$$L(W, (Y, X_1, X_2)^i) = (1 - Y)L_s\left(D_W^i\right) + YL_D(D_W^i), \quad (5)$$

where $(Y, X_1, X_2)^i$ – the i-th labeled pair of samples, L_s – the partial loss function for similar pairs, L_D – the partial loss function for dissimilar pairs, P – the number of training pairs.

L_s and L_D should be designed so that minimizing L with respect to W results in low D_w values for similar pairs and high D_w values for dissimilar pairs [9].

The exact form of the loss function $L(W, Y, X_1, X_2)$ is given by equation (6) [9]:

$$L(W, Y, X_1, X_2) = (1 - Y)\frac{1}{2}(D_W)^2 + Y\frac{1}{2}\{\max(0, m - D_w)\}^2, m > 0, \quad (6)$$

where m – a margin that defines a radius around $G_w(X)$. This means that dissimilar pairs contribute to the loss only when their distance falls within this radius [9].

In addition to the contrastive loss function, an effective approach is the triplet loss function, which is widely used in object recognition and person re-identification (ReID) tasks [10]. The main idea is that each sample consists of three points: an anchor (initial instance), a positive (another instance from

the same class), and a negative (an instance from a different class). The optimization of this function is aimed at ensuring that the distance between the anchor and the positive point is smaller than the distance between the anchor and the negative point by at least a given margin m.

The triplet loss function builds upon the improved Large Margin Nearest Neighbor Loss (LMNN), which consists of two components (7–9) [11]:

$$L_{LMNN}(W) = (1-\mu)L_{pull}(W) + \mu L_{push}(W), \quad (7)$$

$$L_{pull}(W) = \sum_{i,j\in T(i)} D_{i,j}, \quad (8)$$

$$L_{push}(W) = \sum_{\substack{a,n \\ y_a \neq y_n}} \left[m + D_{a,T(a)} - D_{a,n}\right]_+. \quad (9)$$

The first component, L_{pull}, ensures that objects from the same class are drawn closer together, where $T(i)$ represents the target neighbors of a selected point i. The second component, L_{push}, ensures the separation of points from different classes.

FaceNet introduced a modification in the form of the triplet loss function (10) [12]:

$$L_{tri}(W) = \sum_{\substack{a,p,n \\ y_a = y_p \neq y_n}} \left[m + D_{a,p} - D_{a,n}\right]_+. \quad (10)$$

This loss function ensures that the point X_p, belonging to the same class as the anchor X_a, is positioned closer to it than the point X_n, which belongs to a different class. If this condition is violated, the distance between the anchor and the negative sample increases until the required separation is achieved. As a result, samples from the same class form compact clusters that do not collapse into a single point but remain closer to each other than to objects from other classes [11].

Thus, using contrastive and triplet loss functions enables models to effectively distinguish between similar and dissimilar objects while preserving the data structure and ensuring generalization to new instances.

D. ONLINE GENERATION OF PAIRS/TRIPLETS

Online generation of pairs/triplets is a crucial component of model training. The main challenge of the triplet loss approach is that the total number of possible triplets in a dataset grows cubically, making training increasingly complex [11]. Most triplets turn out to be uninformative, as the model quickly learns to correctly classify obvious cases. For instance, if a training set contains images of people in vastly different clothing, the model will easily recognize that they belong to different classes, and further optimization using such examples will be ineffective. Instead, hard positive and hard negative examples, are particularly valuable: the former depict the same person under significantly altered conditions (e.g., different poses or lighting), while the latter involve visually similar but distinct individuals.

In traditional approaches, training triplets are formed at the beginning and then used to compute embeddings and the corresponding loss function [11]. However, this method is not always efficient because random triplet selection may not include a sufficient number of challenging examples. A more effective approach involves forming data batches that include randomly selected classes and their instances. For each selected anchor point, the hardest positive and negative examples within the batch are identified, improving the training process. This method, known as Batch Hard (BH), is formalized by the following loss function (11) [11]:

$$L_{BH}(W;X) = \overbrace{\sum_{i=1}^{P}\sum_{a=1}^{K}}^{all\ anchor} \left[m + \overbrace{\max_{p=1\ldots K} D(G_W(X_a^i), G_W(X_p^i))}^{hard \quad positive} - \overbrace{\min_{\substack{j=1\ldots P \\ n=1\ldots K \\ j\neq i}} D(G_W(X_a^i), G_W(X_n^j))}^{hardest\ negative} \right]_+, \quad (11)$$

where X – a batch of instances, x_i^j – corresponds to the j-th instance of the i-th class.

An alternative approach, known as Batch All (BA), considers all possible triplet combinations within a single data batch. This means that each anchor example is compared with all other positive and negative examples, leading to the following loss function (12) [11]:

$$L_{BA}(W;X) = \overbrace{\sum_{i=1}^{P}\sum_{a=1}^{K}}^{all\ anchors} \overbrace{\sum_{\substack{p=1 \\ p\neq a}}^{K}}^{all\ pos.} \overbrace{\sum_{\substack{j=1 \\ j\neq i}}^{P}\sum_{n=1}^{K}}^{all\ negativies} \left[m + D\left(G_W(X_a^i), G_W\left(X_p^i\right)\right) - D(G_W(X_a^i), G_W(X_n^j))\right]_+. \quad (12)$$

Although this approach considers all possible triplets, potentially increasing computational costs, it provides a broader coverage of interactions between data points, which is beneficial for model training.

Another loss formulation is Lifted Embedding (LE), which extends the idea of triplet loss. Unlike the standard approach, where a negative example is selected only once per triplet, this method considers all possible negative examples within a batch, as well as all positive examples. This results in the following loss function (13) [11]:

$$L_{LE}(W;X) = \overbrace{\sum_{i=1}^{P}\sum_{a=1}^{K}}^{all\ anch} \left[\overbrace{\log \sum_{\substack{p=1 \\ p\neq a}}^{K} e^{D(G_W(X_a^i), G_W(X_p^i))}}^{all\ positives} - \overbrace{\log \sum_{\substack{j=1 \\ j\neq i}}^{P}\sum_{n=1}^{K} e^{m - D(G_W(X_a^i), G_W(X_n^j))}}^{all\ negatives} \right]_+. \quad (13)$$

This formula uses a logarithmic function that allows for more even processing of positive and negative examples, ensuring stable learning.

All the mentioned methods – BH, BA, and LE – are generalized variations of the triplet loss function that optimize training by selecting the most relevant samples within a data batch.

E. USING A GENETIC ALGORITHM TO OPTIMIZE THE MARGIN IN LOSS FUNCTIONS

One of the key hyperparameters in contrastive and triplet loss functions is the margin, which defines the minimum difference between the distances of positive and negative pairs. The value of this parameter significantly affects the quality of training: a margin that is too small may lead to insufficient distinction

between objects, while a margin that is too large may make optimization unstable and hinder convergence. Determining the optimal margin value is a challenging task, as it depends on both the structure of the data and the specifics of the neural network architecture.

To automatically select the margin value, a genetic algorithm (GA) [13] was proposed, which has proven effective in solving optimization problems in complex multidimensional spaces. The idea behind this approach is to evolve a margin value that minimizes the loss function while ensuring the best separation between classes.

The optimization process begins with the initialization of an initial population containing a set of random margin values. Each value in this population is evaluated by computing the loss function on the training data. Then, the core mechanisms of the genetic algorithm are applied: selection identifies the most promising margin values, crossover generates new potential margin values by combining parent values, and mutation introduces random changes that help explore the space of possible solutions and prevent the algorithm from getting stuck in local minima [13].

This process is repeated for a certain number of generations or until a stopping criterion is met, such as the absence of significant improvement over several iterations [13]. After optimization is complete, the best-found margin value is assigned to each training batch, allowing the parameters to be adaptively adjusted to the local characteristics of each data subset. At the testing stage, the average margin value across the entire dataset is calculated and used to assess the overall effectiveness of the model.

This approach ensures more consistent training by accounting for the specific characteristics of individual data subsets without sacrificing the model's generalization ability. Optimizing the margin with a genetic algorithm is an effective way to enhance the performance of neural networks in tasks involving recognition, classification, and object comparison.

III. DEVELOPMENT OF NEURAL NETWORKS

A. SOFTWARE TOOLS AND SAMPLING

The deep learning framework PyTorch is used for implementation and experimentation, providing flexible tools for developing, training, and optimizing neural networks [14]. Its capability for dynamic computation graph definition allows easy modification of model architecture and experimentation with different training approaches. Additionally, PyTorch includes optimized modules for GPU acceleration, significantly speeding up the training process, especially when working with large datasets.

The MNIST dataset is used for training the model, serving as a standard benchmark in handwritten digit recognition. This dataset contains 60,000 images for training and 10,000 for testing, each being a grayscale 28×28 pixel image representing a single digit from 0 to 9 [15]. MNIST is widely used for testing computer vision algorithms and machine learning methods, as it is well-balanced and provides sufficient variability to evaluate model performance [15-26].

B. NEURAL NETWORK STRUCTURES

To compare the performance of Siamese networks, a convolutional network (CNN) was designed, with its layers presented in Table 1.

The use of convolutional layers ensures the automatic extraction of essential image features, such as edges, textures, and shapes. Fully connected layers are used for classification. The final layer, with the Softmax activation function, converts the model output into probabilities for each class. The total number of parameters in this network is 384842.

Table 1. Convolutional network layers for classification

Layer	Depth	Filter Height	Filter Width	Stride	Output Shape	Weights
Input	1				1×28×28	
Conv2d	32	5	5	1	32×28×28	832
MaxPool2d	32			2	32×14×14	
Conv2d	64	5	5	1	64×10×10	53248
MaxPool2d	64			2	64×5×5	
Flatten	1024				1024	
Linear	256				256	262400
Linear	256				256	65792
Linear	10				10	2570

Since the dataset is not very complex, a Siamese network for binary classification was also included for comparison. The layers of this network are presented in Table 2.

Table 2. Siamese network layers (sigmoid activation func.)

Layer	Depth	Filter Height	Filter Width	Stride	Output Shape	Weights
Input	1				1×28×28	
Conv2d	32	5	5	1	32×28×28	832
MaxPool2d	32			2	32×14×14	
Conv2d	64	5	5	1	64×10×10	53248
MaxPool2d	64			2	64×5×5	
Flatten	1024				1024	
Cat	2048				2048	
Linear	256				256	524544
Linear	256				128	65792
Linear	128				1	256

At the input of this network, two images are fed, which pass through convolutional layers to extract embeddings. These embeddings are then combined into a single vector (Cat layer) and processed through several fully connected layers, with the final activation performed using a sigmoid function for binary classification. It is important to note that the convolutional layers are not trained separately for the first and second images but share the same weights during training. The total number of parameters in this network is 644672.

For the embedding search task, a CNN was chosen, with its layers presented in Table 3. This network consists of two main components: a CNN and fully connected layers.

Table 3. Layers of the CNN for embeddings

Layer	Depth	Filter Height	Filter Width	Stride	Output Shape	Weights
Input	1				1×28×28	
Conv2d	32	5	5	1	32×28×28	832
MaxPool2d	32			2	32×14×14	
Conv2d	64	5	5	1	64×10×10	53248
MaxPool2d	64			2	64×5×5	
Flatten	1024				1024	
Linear	256				256	262400
Linear	256				128	65792
Linear	128				128	16394

The model consists of two convolutional layers and three

linear (fully connected) layers, allowing for effective dimensionality reduction and the extraction of compact embeddings. Pooling layers help reduce the image size while preserving essential information for further analysis. The chosen model enables the implementation of both contrastive and triplet loss functions. The total number of parameters is 398666.

IV. TRAINING AND TESTING OF NETWORKS

The training parameters of the convolutional neural network for classifying the MNIST dataset are presented in Table 4. The network's performance results are shown in Figures 1-2.

Table 4. Parameters of the convolutional neural network for classification

Parameter	Value
Number of epochs	20
Model	ConvolutionalNet(1, 10)
Number of weights	384842
Dataset	MNISTClassification
Number of training images	60000
Number of test images	20000
Dataset balancing	No
Training batch size	64
Testing batch size	64
Optimizer	Adam(lr=1e-3)
Scheduler	StepLR(step_size=2, gamma=0.5)
Loss function	CrossEntropyLoss()

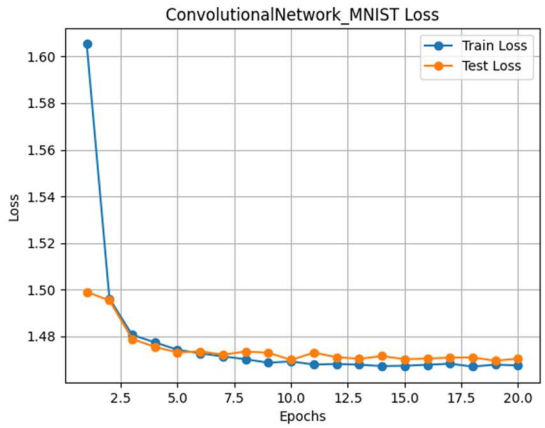

Figure 1. Loss during training of the convolutional neural network.

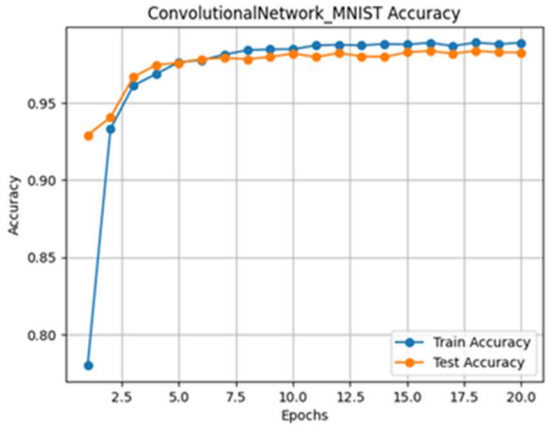

Figure 2. Classification accuracy during training of the convolutional neural network.

On the final epoch, the metric values were as follows: during training, the loss was 1.4675, and accuracy was 0.9889; during testing, the loss was 1.4704, and accuracy was 0.9826.

The model achieved a relatively high classification accuracy by the 10th epoch and then remained at the same level. This provides baseline metric values for comparison with Siamese networks.

The parameters for launching the Siamese neural network with a sigmoid activation function for classifying the MNIST dataset are presented in Table 5, and the network's performance results are shown in Figures 3-4.

Table 5. Parameters of the Siamese neural network with a sigmoid activation function

Parameter	Value
Number of epochs	20
Model	SiameseNetSigmoid(1)
Number of weights	644672
Dataset	MNISTPair
Number of training images	60000
Number of test images	20000
Dataset balancing	No
Training batch size	64
Testing batch size	64
Optimizer	Adam(lr=1e-3)
Scheduler	StepLR(step_size=2, gamma=0.5)
Loss function	BCELoss()

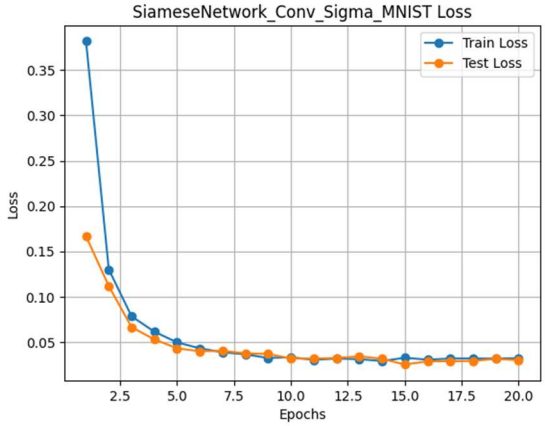

Figure 3. Loss during training of the Siamese neural network with a sigmoid activation function.

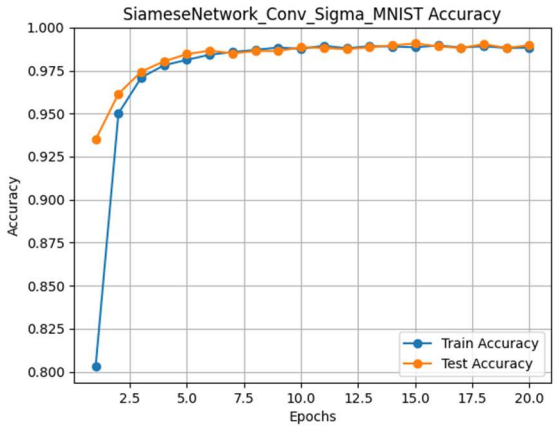

Figure 4. Classification accuracy during training of the Siamese neural network with a sigmoid activation function.

On the final epoch, the metric values were as follows: during training, the loss was 0.0326, and accuracy was 0.9883; during testing, the loss was 0.0305, and accuracy was 0.9897.

The model achieved a similar accuracy as the convolutional network; however, its convergence during training was significantly better.

The parameters for launching the Siamese neural network with a contrastive loss function for classifying the MNIST dataset are presented in Table 6, and the network's performance results are shown in Figures 5-6.

Table 6. Parameters of the Siamese neural network with a contrastive loss function

Parameter	Value
Number of epochs	20
Model	EmbeddingNet(1)
Number of weights	398666
Dataset	MNISTPair
Number of training images	60000
Number of testing images	20000
Dataset balancing	No
Training batch size	64
Testing batch size	64
Optimizer	Adam(lr=1e-3)
Scheduler	StepLR(step_size=2, gamma=0.5)
Loss function	ContrastiveLoss(reduction="sum")

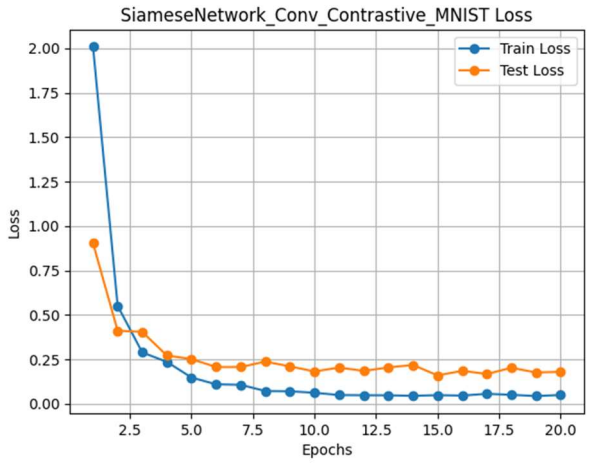

Figure 5. Loss during training of the Siamese neural network with a contrastive loss function.

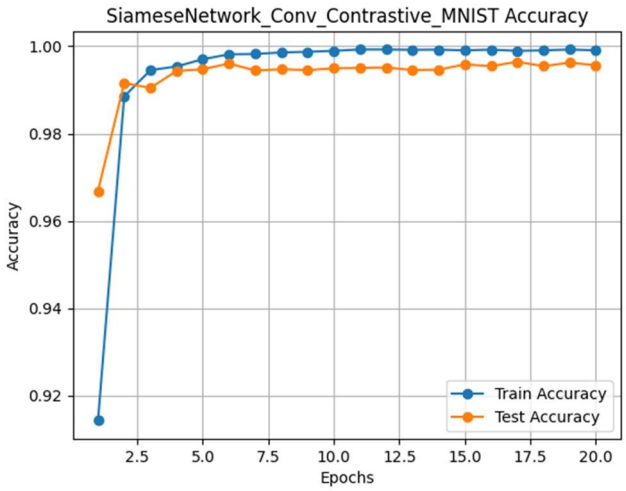

Figure 6. Classification accuracy during training of the Siamese neural network with a contrastive loss function.

On the final epoch, the metric values were as follows: during training, the loss was 0.0499, and accuracy was 0.9962; during testing, the loss was 0.1800, and accuracy was 0.9956.

The parameters for launching the Siamese neural network with a contrastive loss function and adaptive margin for classifying the MNIST dataset are presented in Table 7, and the model's performance results are shown in Figures 7-8.

Table 7. Parameters of the Siamese neural network with a contrastive loss function and adaptive margin

Parameter	Value
Number of epochs	20
Model	EmbeddingNet(1)
Number of weights	398666
Dataset	MNISTPairr
Number of training images	60000
Number of testing images	20000
Dataset balancing	No
Training batch size	64
Testing batch size	64
Optimizer	Adam(lr=1e-3)
Scheduler	StepLR(step_size=2, gamma=0.5)
Loss function	ContrastiveLoss(useGA=True, reduction="sum")

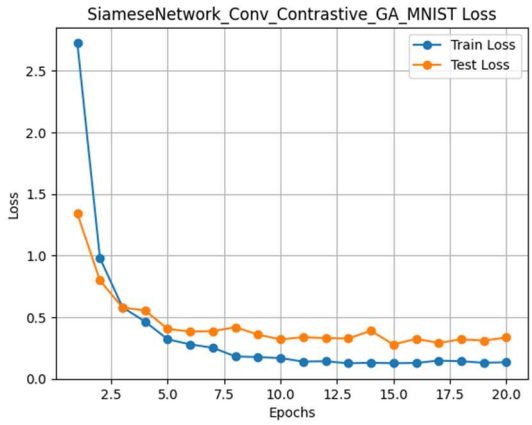

Figure 7. Loss during training of the Siamese neural network with a contrastive loss function and adaptive margin.

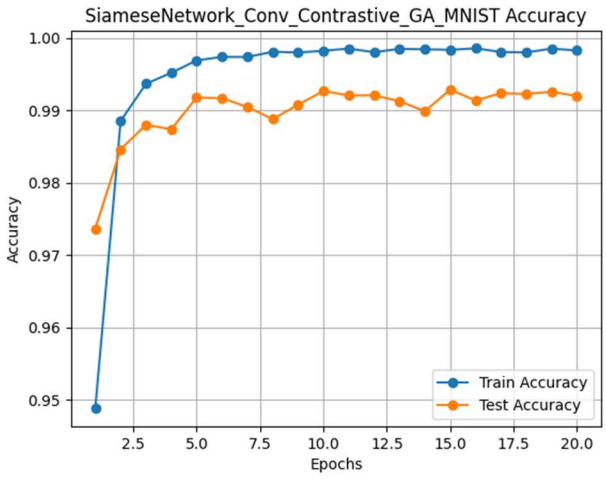

Figure 8. Classification accuracy during training of the Siamese neural network with a contrastive loss function and adaptive margin.

On the final epoch, the metric values were as follows: during training, the loss was 0.1349, and accuracy was 0.9982; during testing, the loss was 0.3356, and accuracy was 0.9919.

The parameters for launching the Siamese neural network with an online contrastive loss function and Batch Hard strategy for classifying the MNIST dataset are presented in

Table 8, and the network's performance results are shown in Figures 9-10. For this network, dataset balancing was used to ensure an even class distribution within batches.

Table 8. Parameters of the Siamese neural network with an online contrastive loss function and Batch Hard strategy

Parameter	Value
Number of epochs	20
Model	EmbeddingNet(1)
Number of weights	398666
Dataset	MNISTBase
Number of training images	60000
Training dataset balancing	Yes
Training batch size	64
Training loss function	OnlineContrastiveLoss(threshold=0.7, reduction="sum")
Optimizer	Adam(lr=1e-3)
Scheduler	StepLR(step_size=2, gamma=0.5)
Testing dataset	MNISTPair
Number of testing images	20000
Testing dataset balancing	No
Testing batch size	64
Testing loss function	ContrastiveLoss(reduction="sum")

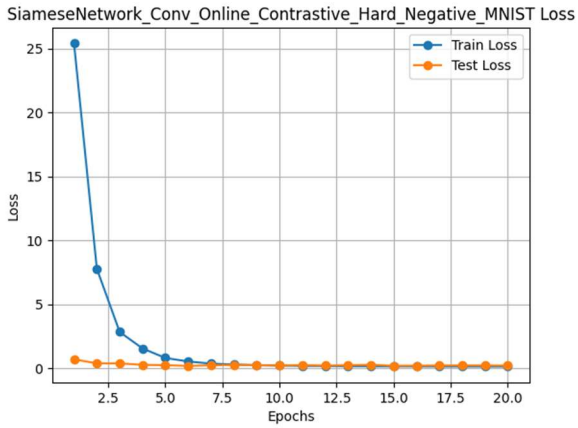

Figure 9. Loss during training of the Siamese neural network with an online contrastive loss function and Batch Hard strategy.

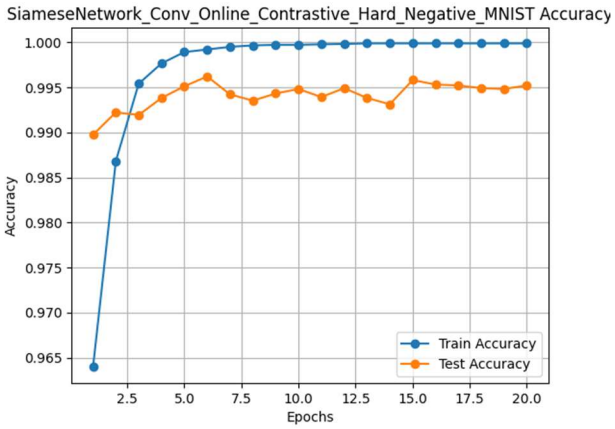

Figure 10. Classification accuracy during training of the Siamese neural network with an online contrastive loss function and Batch Hard strategy.

On the final epoch, the metric values were as follows: during training, the loss was 0.1240, and accuracy was 0.9999; during testing, the loss was 0.1941, and accuracy was 0.9952.

A comparison of the accuracy of the considered models with contrastive loss functions on test datasets is shown in Figure 11.

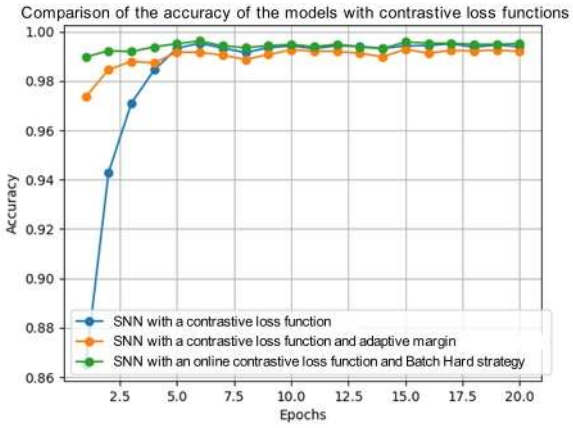

Figure 11. Classification accuracy during testing of Siamese neural networks with contrastive loss functions.

The model with a genetically optimized adaptive margin performed better during the early epochs than the model with a fixed margin. However, the online loss function achieved the best overall performance.

The parameters for launching the Siamese neural network with a triplet loss function for classifying the MNIST dataset are presented in Table 9, and the network's performance results are shown in Figures 12-13.

Table 9. Parameters of the Siamese neural network with a triplet loss function

Parameter	Value
Number of epochs	20
Model	EmbeddingNet(1)
Number of weights	398666
Dataset	MNISTTriplet
Number of training images	90000
Number of testing images	20000
Dataset balancing	No
Training batch size	64
Testing batch size	64
Optimizer	Adam(lr=1e-3)
Scheduler	StepLR(step_size=2, gamma=0.5)
Loss function	torch.nn.TripletMarginWithDistanceLoss(reduction="sum")

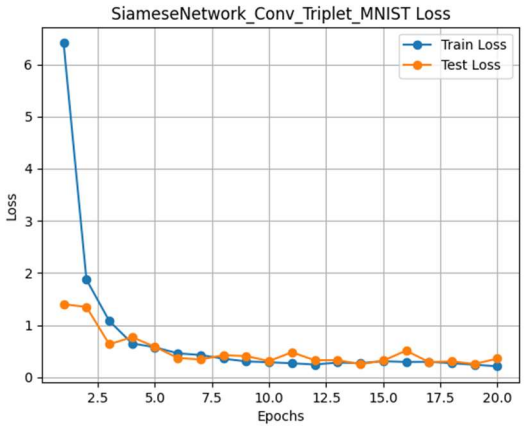

Figure 12. Loss during training of the Siamese neural network with a triplet loss function.

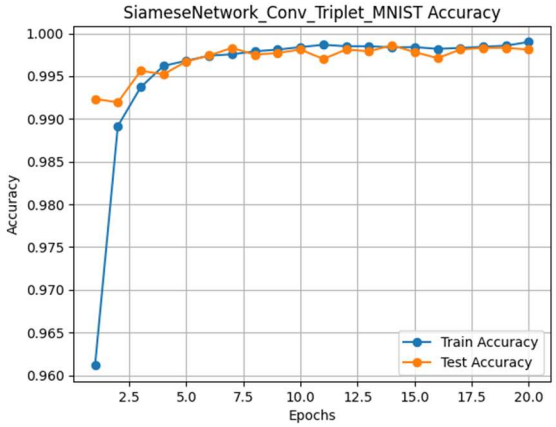

Figure 13. Classification accuracy during training of the Siamese neural network with a triplet loss function.

On the final epoch, the metric values were as follows: during training, the loss was 0.2082, and accuracy was 0.9990; during testing, the loss was 0.3559, and accuracy was 0.9981.

The model demonstrated the same accuracy and convergence as the online contrastive loss function with the Batch Hard strategy.

The parameters for launching the Siamese neural network with a triplet loss function and adaptive margin for classifying the MNIST dataset are presented in Table 10, and the results are shown in Figures 14-15.

Table 10. Parameters of the Siamese neural network with a triplet loss function and adaptive margin

Parameter	Value
Number of epochs	20
Model	EmbeddingNet(1)
Number of weights	398666
Dataset	MNISTTriplet
Number of training images	90000
Number of testing images	20000
Dataset balancing	No
Training batch size	64
Testing batch size	64
Optimizer	Adam(lr=1e-3)
Scheduler	StepLR(step_size=2, gamma=0.5)
Loss function	torch.nn.TripletMarginWithDistanceLoss (useGA=True, reduction="sum")

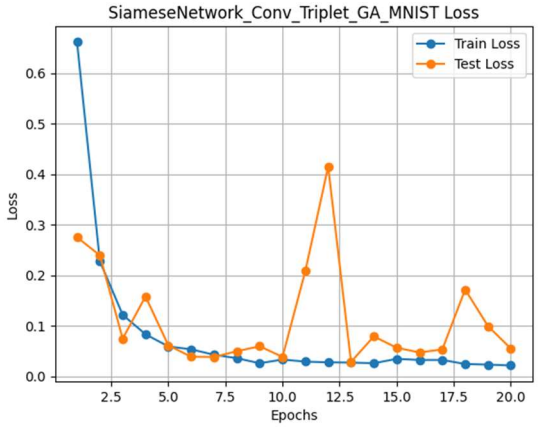

Figure 14. Loss during training of the Siamese neural network with a triplet loss function and adaptive margin.

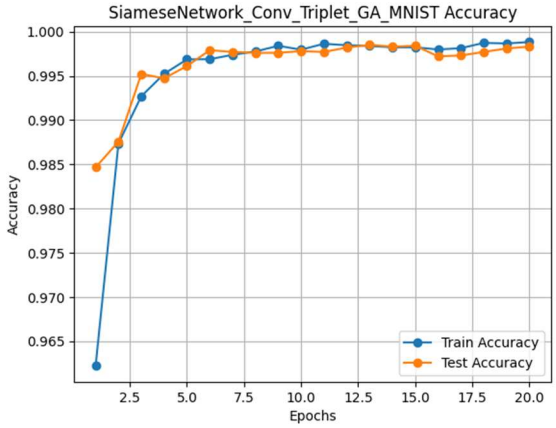

Figure 15. Classification accuracy during training of the Siamese neural network with a triplet loss function and adaptive margin.

On the final epoch, the metric values were as follows: during training, the loss was 0.0220, and accuracy was 0.9988; during testing, the loss was 0.0554, and accuracy was 0.9983.

The parameters for launching the Siamese neural network with an online triplet loss function and Batch Hard strategy for classifying the MNIST dataset are presented in Table 11, and the network's performance results are shown in Figures 16-17. For this network, dataset balancing was used to ensure an even class distribution within batches.

Table 11. Parameters of the Siamese neural network with an online triplet loss function and Batch Hard strategy

Parameter	Value
Number of epochs	20
Model	EmbeddingNet(1)
Number of weights	398666
Dataset	MNISTTriplet
Number of training images	90000
Training dataset balancing	Yes
Training batch size	64
Training loss function	OnlineTripletLoss(margin=1, reduction="sum")
Optimizer	Adam(lr=1e-3)
Scheduler	StepLR(step_size=2, gamma=0.5)
Testing dataset	MNISTTriplet
Number of testing images	20000
Testing dataset balancing	No
Testing batch size	64
Testing loss function	torch.nn.TripletMarginWith DistanceLoss(reduction="sum")

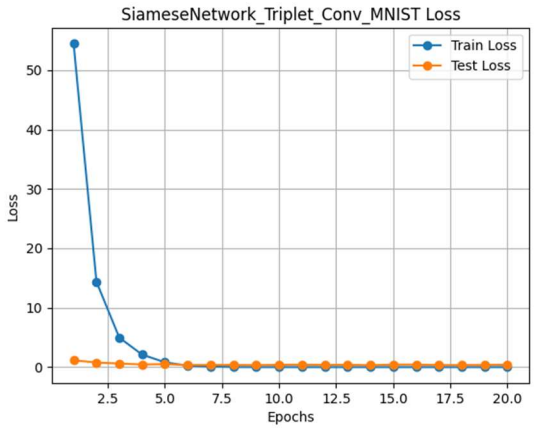

Figure 16. Loss during training of the Siamese neural network with an online triplet loss function and Batch Hard strategy.

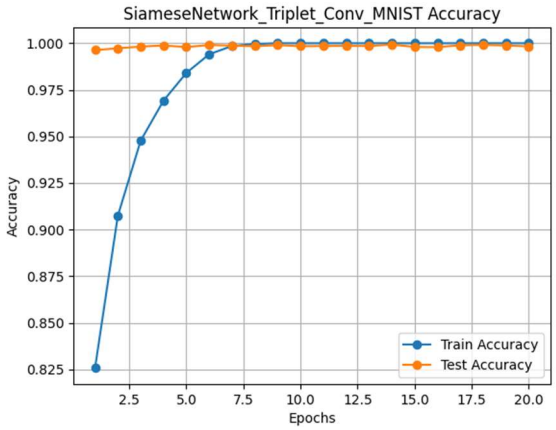

Figure 17. Classification accuracy during training of the Siamese neural network with an online triplet loss function and Batch Hard strategy.

On the final epoch, the metric values were as follows: during training, the loss was 0.0000, and accuracy was 1.0000; during testing, the loss was 0.4109, and accuracy was 0.9983.

A comparison of accuracies on test datasets is shown in Figure 18. It can be observed that the online loss function performs better than the standard one. The adaptive margin loss function works slightly worse than the simple triplet loss function during the initial epochs.

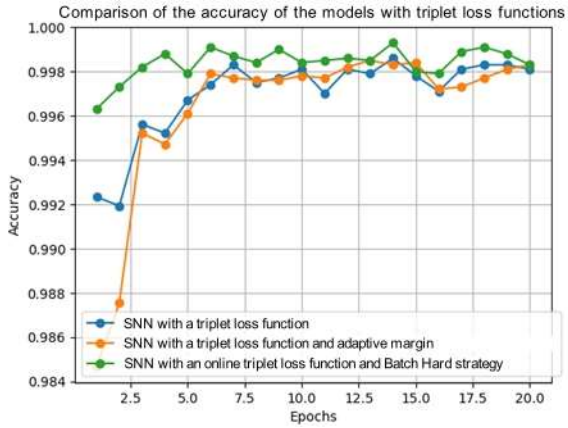

Figure 18. Classification accuracy during testing of Siamese neural networks with triplet loss functions.

Figure 19 presents a comparison of all discussed networks based on the classification accuracy of the training dataset.

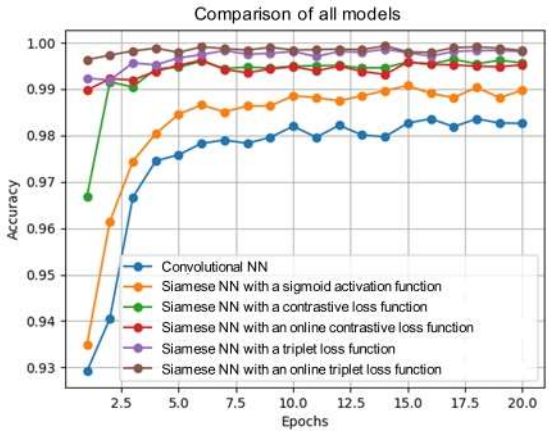

Figure 19. Classification accuracy of all networks.

As previously described, the more complex the model and sample-selection strategy, the better the model's classification performance. The Siamese neural network with an online triplet loss function and the Batch Hard strategy demonstrated the best classification accuracy. Additionally, models using online image selection strategies showed better results than those using random image selection.

V. CONCLUSIONS

During the study, methods for object recognition in images based on Siamese neural networks were examined and implemented programmatically. The primary focus was on loss functions, which determine the efficiency of training such models. Specifically, contrastive and triplet loss functions were implemented, allowing models to be trained for classification and similarity search between objects.

The scientific novelty of this work lies in the application of a genetic algorithm for the automatic optimization of the margin parameter in loss functions. The traditional approach involves manual selection of this parameter or its empirical determination, which may lead to suboptimal results. The proposed method dynamically adjusts the margin during training, taking into account the specifics of each data batch. This enhances the adaptability of the model and improves its generalization ability, especially in complex tasks with high data variability.

Additionally, the study explored pair and triplet selection strategies, including Batch Hard, Batch All, and Lifted Embedding. These strategies enable more efficient formation of training samples, positively impacting training speed and accuracy. Experimental comparisons showed that the triplet loss function combined with the Batch Hard strategy yielded the best results.

The models were implemented using PyTorch, which provides flexibility in creating, configuring, and training neural networks. The MNIST dataset was used as the training and testing set, as it is a standard benchmark for evaluating image recognition models.

The developed models were evaluated using several metrics, allowing for the determination of the optimal architecture and training parameters. The obtained results confirmed the effectiveness of using the genetic algorithm for margin optimization and the importance of well-designed strategies for selecting training samples. Overall, the study demonstrates that enhanced training approaches for Siamese neural networks can improve classification accuracy and similarity-search performance on the MNIST dataset.

References

[1] S. C. Mana, T. Sasipraba, "Research on cosine similarity and Pearson correlation based recommendation models," *Journal of Physics: Conference Series*, vol. 1770, issue 1, p. 012014, 2021, https://doi.org/10.1088/1742-6596/1770/1/012014.

[2] S. Mukherjee, R. Sonal, "A reconciliation between cosine similarity and Euclidean distance in individual decision-making problems," *Indian Economic Review*, vol. 58, issue 2, pp. 427–431, 2023, https://doi.org/10.1007/s41775-023-00206-8.

[3] P. K. Singh, S. Setta, I. S. Rajput, "A modified Spearman's rank correlation coefficient for an efficient method of similarity calculation in collaborative filtering-based recommendation," *SSRN Electron. J.*, 2019, https://doi.org/10.2139/ssrn.3350274.

[4] D. Chicco, "Siamese neural networks: An overview," *Methods in Molecular Biology*, pp. 73–94, 2020, https://doi.org/10.1007/978-1-0716-0826-5_3.

[5] R. Mohanasundaram, A. S. Malhotra, R. Arun, P. S. Periasamy, "Deep learning and semi-supervised and transfer learning algorithms for medical imaging," *Deep Learning and Parallel Computing Environment*

for Bioengineering Systems, pp. 139–151, 2019, https://doi.org/10.1016/B978-0-12-816718-2.00015-4.
[6] H. de Rosa Gustavo, J. P. Papa, "Learning to weight similarity measures with Siamese networks: a case study on optimum-path forest," *Elsevier eBooks*, pp. 155–173, 2022, https://doi.org/10.1016/B978-0-12-822688-9.00015-3.
[7] T. Lesort, N. Díaz-Rodríguez, J.-F. Goudou, D. Filliat, "State representation learning for control: An overview," *Neural Networks*, vol. 108, pp. 379–392, 2018, https://doi.org/10.1016/j.neunet.2018.07.006.
[8] A. Dhiman, K. Gupta, D. K. Sharma, *An Introduction to Deep Learning Applications in Biometric Recognition*, in: Trends in Deep Learning Methodologies, Elsevier, 2021, pp. 1–36, https://doi.org/10.1016/B978-0-12-822226-3.00001-5.
[9] R. Hadsell, S. Chopra, Y. LeCun, "Dimensionality reduction by learning an invariant mapping," *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, New York, NY, USA, 2006, pp. 1735-1742, https://doi.org/10.1109/CVPR.2006.100.
[10] S. Zhang, Q. Zhang, X. Wei, Y. Zhang, Y. Xia, "Person re-identification with triplet focal loss," *IEEE Access*, vol. 6, pp. 78092–78099, 2018, https://doi.org/10.1109/ACCESS.2018.2884743.
[11] A. Hermans, L. Beyer, B. Leibe, "In defense of the triplet loss for Person re-identification," *arXiv: Computer Vision and Pattern Recognition*, 2017, https://doi.org/10.48550/arXiv.1703.07737.
[12] F. Schroff, D. Kalenichenko, J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 815-823, https://doi.org/10.1109/CVPR.2015.7298682.
[13] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications*, vol. 80, issue 5, pp. 8091–8126, 2021, https://doi.org/10.1007/s11042-020-10139-6.
[14] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," *arXiv (Cornell University)*, 2019, https://doi.org/10.48550/arxiv.1912.01703.
[15] H. Shao, E. Ma, M. Zhu, X. Deng, S. Zhai, "MNIST handwritten digit classification based on convolutional neural network with hyperparameter optimization," *Intelligent Automation & Soft Computing*, vol. 36, issue 3, pp. 3595–3606, 2023, https://doi.org/10.32604/iasc.2023.036323.
[16] J. Alsayaydeh, J. Abedalrahim, A. Aziz, M. I. Abbasi, A. W. Y. Khang, "Development of programmable home security using GSM system for early prevention," *ARPN J. Eng. Appl. Sci*., vol. 16, issue 1, pp. 88–97, 2021.
[17] W. A. Indra, N. S. Zamzam, A. Saptari, J. A. Alsayaydeh, N. B. Hassim, "Development of security system using motion sensor powered by RF energy harvesting," *Proceedings of the 2020 IEEE Student Conference on Research and Development (SCOReD)*, Batu Pahat, Malaysia, 2020, pp. 254-258. https://doi.org/10.1109/SCOReD50371.2020.9250984.
[18] A. Kharchenko, I. Fedorchenko, T. Fedoronchak, A. Oliinyk, A. Stepanenko, M. Saman, "The development of a genetic method for predicting the incidence of diabetes," Proceedings of the *2021 11th International Conference on Advanced Computer Information Technologies (ACIT)*, Deggendorf, Germany, 2021, pp. 708-713. https://doi.org/10.1109/ACIT52158.2021.9548405.
[19] Z. Zhang, E. Sejdić, "Radiological images and machine learning: Trends, perspectives, and prospects," *Computers in Biology and Medicine*, vol. 108, pp. 354–370, 2019, https://doi.org/10.1016/j.compbiomed.2019.02.017.
[20] I. Fedorchenko, A. Oliinyk, O. Stepanenko, T. Zaiko, A. Svyrydenko, D. Goncharenko, "Genetic method of image processing for motor vehicle recognition," Proceedings of the 2nd International Workshop on Computer Modeling and Intelligent Systems, CMIS 2019, Zaporizhzhia; Ukraine; 15-19 April 2019, vol. 2353, pp. 211-226. https://doi.org/10.32782/cmis/2353-17.
[21] G. Jain, T. Mahara, and K. N. Tripathi, "A survey of similarity measures for collaborative filtering-based recommender system," *Soft Computing: Theories and Applications*, vol. 1053, pp. 343–352, 2020. https://doi.org/10.1007/978-981-15-0751-9_32.
[22] I. Izonin, R. Tkachenko, N. Shakhovska, N. Lotoshynska, "The additive input-doubling method based on the SVR with nonlinear kernels: Small data approach," *Symmetry*, vol. 13, issue 4, 612, 2021. https://doi.org/10.3390/sym13040612.
[23] F. Fkih, "Similarity measures for collaborative filtering-based recommender systems: Review and experimental comparison," *Journal of King Saud University – Computer and Information Sciences*, vol. 34, no. 9, pp. 7645–7669, 2022. https://doi.org/10.1016/j.jksuci.2021.09.014.
[24] A. Gazdar and L. Hidri, "A new similarity measure for collaborative filtering based recommender systems," *Knowledge-Based Systems*, vol. 188, Art. 105058, 2020. https://doi.org/10.1016/j.knosys.2019.105058.
[25] C. R. Lee and K. Kim, "An improved similarity measure for collaborative filtering-based recommendation system," *Knowledge-Based Systems*, vol. 26, no. 2, pp. 137-147, 2022. https://doi.org/10.3233/KES220013.
[26] N. Nudrat et al., "Users' rating predictions using collaborating filtering based on users and items similarity measures," *Computational Intelligence and Neuroscience*, vol. 2022, 2347641, pp. 1-13, 2022. https://doi.org/10.1155/2022/2347641.

IEVGEN FEDORCHENKO, *Senior Lecturer of Software Tools Department of National University "Zaporizhzhia Polytechnic". Field of scientific interests: Artificial Intelligence, Modeling, Diagnostics.*
https://orcid.org/0000-0003-1605-8066
Email: evg.fedorchenko@gmail.com

PROF. ANDRII OLIINYK *Dr. Sc., Professor of Software Tools Department of National University "Zaporizhzhia Polytechnic". Field of scientific interests: Artificial Intelligence, Modeling, Diagnostics.*
https://orcid.org/0000-0002-6740-6078
Email: olejnikaa@gmail.com

TETIANA FEDORONCHAK *Assoc. Professor of Software Tools Department of National University "Zaporizhzhia Polytechnic". Field of scientific interests: Artificial Intelligence, Modeling, Diagnostics.*
https://orcid.org/0000-0001-6238-1177
Email: t.fedoronchak@gmail.com

ALEXANDER STEPANENKO *Assoc. Professor of Software Tools Department of National University "Zaporizhzhia Polytechnic". Field of scientific interests: Artificial Intelligence, Modeling, Diagnostics.*
https://orcid.org/0000-0002-0267-7440
Email: alex@zntu.edu.ua

YEVGENY GOFMAN *Assoc. Professor of Software Tools Department of National University "Zaporizhzhia Polytechnic". Field of scientific interests: Artificial Intelligence, Diagnostics, Databases.*
https://orcid.org/0009-0001-2885-4185
Email: gofmanjenek@gmail.com

VLADYSLAV PRYMAKOV *Graduate student of the National University "Zaporizhzhia Polytechnic". Field of scientific interests: Machine Learning, Neural Networks.*
Email: primakov2002@gmail.com