Application of a Ten-Variate Prediction Ellipsoid for Normalized Data and Machine Learning Algorithms for Face Recognition

Sergiy Prykhodko[1,2] and Artem Trukhov[1]

[1] *Admiral Makarov National University of Shipbuilding, Heroes of Ukraine Ave., 9, Mykolaiv, 54007, Ukraine*
[2] *Odesa Polytechnic National University, Shevchenko Ave., 1, Odesa, 65044, Ukraine*

Abstract

Face recognition plays a pivotal role in enhancing security, improving efficiency, and enabling innovative applications across diverse industries, making it an indispensable tool in today's world. This study presents a comparative investigation between prediction ellipsoid and machine learning algorithms, such as a one-class support vector machine, isolation forest, and an autoencoder based on a neural network, in the context of face recognition. The construction of a prediction ellipsoid is based on the assumption of a multivariate normal distribution of the data, however, in real-world scenarios, the data often deviates from this assumption and may exhibit a non-Gaussian distribution. To mitigate this, the dataset underwent normalization via the multivariate Box-Cox transformation, facilitating the development of a decision rule based on a ten-variate prediction ellipsoid for the normalized data. This approach yielded notable enhancements in accuracy and method robustness. All the developed models have shown good efficiency in their respective performances. However, the application of the multivariate Box-Cox normalizing transformation significantly enhanced the performance of the prediction ellipsoid. As a result, the ten-variable prediction ellipsoid for normalized data has demonstrated superior efficiency compared to the other models. The study also highlights the critical role of feature dimensionality in face recognition, emphasizing the necessity of expanding the number of features and utilizing more complex models to achieve optimal performance.

Keywords

Face recognition, multivariate normal distribution, Box-Cox transformation, machine learning

1. Introduction

Face recognition, a quintessential task in computer vision, holds significant importance in various fields, including security, surveillance, human-computer interaction, and biometrics. It entails identifying or verifying individuals by analyzing and comparing facial features captured through images or videos.

The face recognition task contains several crucial steps, each essential for ensuring the system's accuracy, robustness, and reliability [1]. Firstly, image preprocessing enhances the quality and consistency of facial images for subsequent analysis. Next, face detection locates and recognizes faces within an image, laying the foundation for recognition procedures. This step employs methods like the Viola-Jones, MTCNN, or computer vision libraries such as DLib. Following detection, feature extraction characterizes each face uniquely. This involves capturing geometric characteristics like distances between facial landmarks, e.g., eyes, nose, and mouth, generating a distinctive feature vector. The final step involves the application of decision rules to determine whether the feature vector belongs to the representative of the target class.

Traditionally, face recognition systems have been approached through classification methods, where the goal is to classify an input face image into one of several predefined classes. However, these methods often face challenges in scenarios where the available data for training does not adequately represent the entire population, leading to poor performance on novel faces. To address this limitation, the one-class classification approach has emerged. This approach, closely related to anomaly detection [2], focuses on learning representations of a single class, typically

CMIS-2024: Seventh International Workshop on Computer Modeling and Intelligent Systems, May 3, 2024, Zaporizhzhia, Ukraine

sergiy.prykhodko@nuos.edu.ua (S. Prykhodko); artem.trukhov@gmail.com (A. Trukhov)

0000-0002-2325-018X (S. Prykhodko); 0000-0002-7160-8609 (A. Trukhov)

the target, without explicit knowledge of the negative classes, representatives of these classes are considered anomalies.

In the domain of face recognition, one-class classification stands as a pivotal methodology, enabling the identification of target classes amidst varying data distributions. Prediction ellipsoid and machine learning methods emerge as frontrunners among the popular approaches in this realm [3-5]. Given their significance in the field, this study aims to compare these diverse approaches in the context of face recognition tasks, examining their performance, robustness, and applicability.

2. Literature review

The field of face recognition has seen significant advancements in recent years, with various techniques developed such as linear discriminate analysis, support vector machine, and principal component analysis combined with face recognition by using K-Nearest-Neighbor and neural networks, including convolutional neural networks [6]. Traditional methods for face recognition typically rely on supervised learning approaches, where the system is trained on a dataset containing samples from each class or individual to be recognized. However, these methods require a predefined set of classes or identifiers, which makes them less suitable for scenarios where comprehensive training data is not available or when the study population is highly diverse. Another disadvantage is that such systems are poorly suited for the development of an identification model, where it is necessary to verify whether a specific individual belongs to a particular class. This makes one-class classification particularly well-suited for scenarios where only data from the target class is available or when the negative class is poorly defined. That is why for abnormal image detection, abnormal event detection, active authentication, and antispoofing one-class classification methods are extensively used [7].

Among the common approaches to one-class classification, prediction ellipsoid [3, 4] and machine learning methods [5] have emerged as popular choices. Notable examples include one-class support vector machine (OCSVM) [8, 9], isolation forest (IF) [10], and autoencoder (AE) [11, 12]. OCSVM is a support vector machine trained on a target class of data, learning a decision boundary that separates the data points from the origin in the feature space while maximizing the margin. IF is an ensemble learning method that isolates anomalies by randomly selecting features and partitioning data points until anomalies are isolated into small partitions, requiring fewer splits compared to target points. AE is a neural network designed to learn efficient representations of data by encoding the input into a lower-dimensional space–encoder, and then reconstructing the original input from this space-decoder, anomalies are discerned by evaluating the reconstruction error, with heightened errors indicating potential outliers.

The application of a prediction ellipsoid relies on the assumption of a multivariate normal distribution within the data, which may not always hold, this discrepancy necessitates the utilization of normalization transformations [13]. These transformations aim to align the data more closely with the assumptions of a multivariate normal distribution, thereby enhancing the accuracy and robustness of the analysis. Normalization techniques encompass various approaches, including univariate transformations such as logarithmic or Box-Cox transformations, which operate independently on each feature. Conversely, multivariate transformations like the multivariate Box-Cox transformation consider inter-feature relationships, resulting in more effective normalization of the data [14].

Our study primarily investigates the method based on prediction ellipsoid and machine learning algorithms, such as OCSVM, IF, and AE, which are commonly utilized and offer distinct approaches to one-class classification. Specifically, OCSVM delineates a discerning boundary around normal instances, IF isolates anomalies through iterative partitioning of the feature space, AE employs a neural network approach, and the prediction ellipsoid separates normal instances from potential outliers based on their position in the feature space.

In the realm of face recognition, where accuracy and efficiency are paramount, it's imperative to evaluate the efficacy of different approaches. While prediction ellipsoid offers interpretability and computational efficiency, they may encounter limitations with non-Gaussian data distributions. In contrast, machine learning methods such as OCSVM, IF, or AE present alternative methodologies, each with unique strengths and challenges.

3. Face detection and feature extraction

In the domain of computer vision, a variety of techniques are employed for detecting faces in images, each characterized by distinct strengths and functionalities. Notable among these methods are the Viola-Jones, Multi-Task Cascaded Convolutional Neural Network (MTCNN), and DLib toolkit [15].

The Viola-Jones represents a classic yet effective approach to face detection, renowned for its simplicity and efficiency. Operating by scanning images at multiple scales, it utilizes a predefined set of features to pinpoint regions of interest potentially containing faces. It is implemented in the Python OpenCV library.

MTCNN, on the other hand, stands out as a deep learning-based face detection tool prized for its remarkable accuracy and resilience. Comprising three stages—face detection, bounding box regression, and facial landmark localization—MTCNN leverages a cascade of convolutional neural networks to precisely identify faces in images while concurrently estimating facial landmarks. This method is particularly advantageous for applications necessitating precise localization of facial features.

DLib emerges as a comprehensive C++ toolkit with formidable capabilities in facial detection and recognition. Leveraging a Histogram of Oriented Gradients and Linear Support Vector Machine face detector, DLib excels in detecting faces and estimating 68 facial points. These points facilitate accurate detection and localization of facial features, enabling seamless implementation of face alignment, merging, and other transformations.

While all face detection methods exhibit commendable performance in high-quality images, they display variations in performance under different conditions. MTCNN and DLib demonstrate comparable accuracy, with MTCNN exhibiting superior performance in recognizing faces in small images and low-light conditions. In contrast, the Viola-Jones method may encounter challenges with distortions. Performance evaluations conducted using a standardized test dataset reveal that Viola-Jones operates at the highest speed, processing 11.25 frames per second (fps), followed closely by DLib at 9.41 fps, and MTCNN at 4.92 fps.

Given its good speed and proficiency in extracting a substantial number of key points, DLib emerges as the preferred choice for face detection and feature extraction tasks, particularly in scenarios where real-time processing or high throughput is essential.

A program was created using the DLib computer vision library in Python to generate feature vectors. Once a face is detected in the input image, the program begins several image-processing tasks. These tasks include cropping and aligning the face to ensure that the eyes are aligned at the same level. This preprocessing step helps reduce distortions caused by the face's orientation in the input image. In the last phase, the program extracts a series of attributes from the aligned image. Each attribute denotes the pixel distance between facial landmarks identified by the DLib library. It used 17 key landmarks across the face (fig. 1).

Utilizing the pixel distances between these landmarks, a feature vector composed of 10 distinct attributes was meticulously crafted. Symmetrical distances were thoughtfully averaged to derive the following features:

1. The eyes-to-nose midpoint distance indicates the facial symmetry and proportionality between the eyes and the nose.
2. The eyes-to-mouth center distance reflects the vertical alignment and proportionality between the eyes and the mouth.
3. The eyes-to-eyebrows center distance offers insights into the spatial relationship between the eyes and the eyebrows.
4. The eyebrows-to-nose bridge distance delineates the vertical dimension and proportionality between the eyebrows and the nose bridge.
5. The eye's corners-to-nose bridge distance provides insights into the lateral dimension and proportionality between the eye's corners and the nose bridge.
6. The eyebrows distance.
7. The nose-to-mouth midpoint distance offers insights into the vertical alignment between the nose and the mouth.
8. The corners of the mouth distance, reflect the width of the mouth.
9. The edges of the nose distance, delineate the width of the nose.

10. The mouth-to-chin distance provides insights into the vertical dimension between the mouth and the chin.

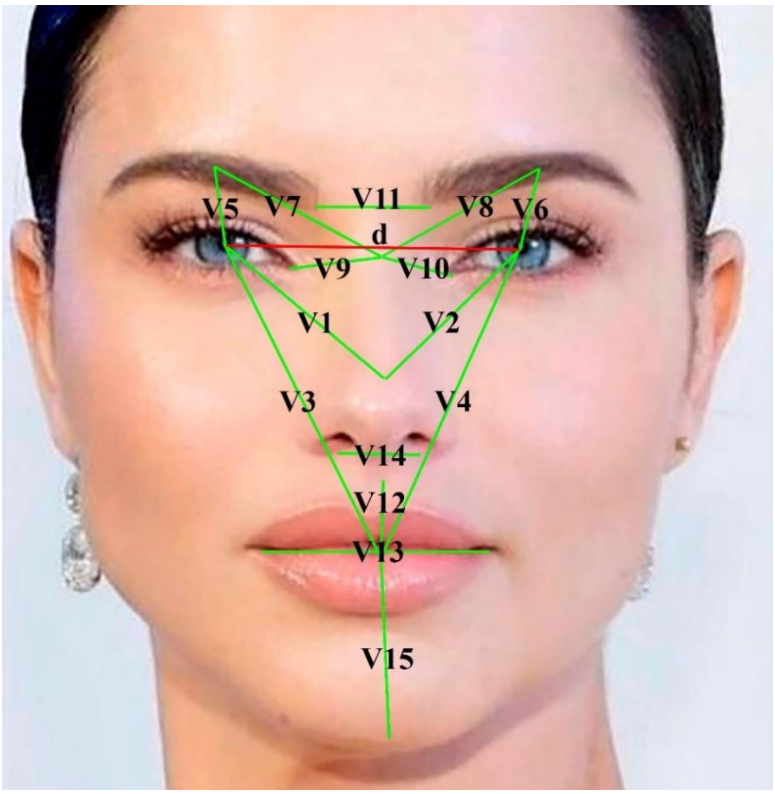

Figure 1: Distances between facial landmarks used to generate the feature vector

To adapt to variations in facial positioning within the image and varying distances from the camera, a normalization procedure is implemented. This involves dividing each feature by the distance between the eyes. By scaling each feature relative to the interocular distance, the normalization process ensures that facial characteristics remain consistent across different images and distances, enhancing the robustness and accuracy of subsequent analysis and recognition tasks.

For the research, the well-known dataset Pins Face Recognition was chosen, which is used in building decision rules for facial recognition. To ensure the reliability and accuracy of the study, the dataset underwent filtering, followed by additional downloading of photos from open sources and expansion of the dataset. In total, 400 high-quality photos were obtained, 200 for each of the two faces. Consequently, 400 feature vectors were acquired, each corresponding to a single photo, comprising 10 elements. Out of these, 100 vectors of a single person were used to construct a prediction ellipsoid and train the machine learning models for identifying the individual, while the remaining 300 photos were allocated as the test set.

4. Materials and methods

4.1. Prediction ellipsoid for normalized data

In the construction of the prediction ellipsoid, each observation's squared Mahalanobis distance is calculated and compared to critical values derived from the Chi-square distribution [16].

$$(X - \bar{X})^T S_X^{-1} (X - \bar{X}) \leq \chi^2_{k,\alpha} \ , \tag{1}$$

where α is the confidence level, k is the number of dimensions in the data,

$$S_X = \frac{1}{N} \sum_{i=1}^{N} (X_i - \bar{X})(X_i - \bar{X})^T. \tag{2}$$

Figure 2 illustrates the example of a 3-variate prediction ellipsoid. In our study, we use a 10-variate prediction ellipsoid. Objects within its boundaries are classified as the target class, while those outside are identified as anomalies.

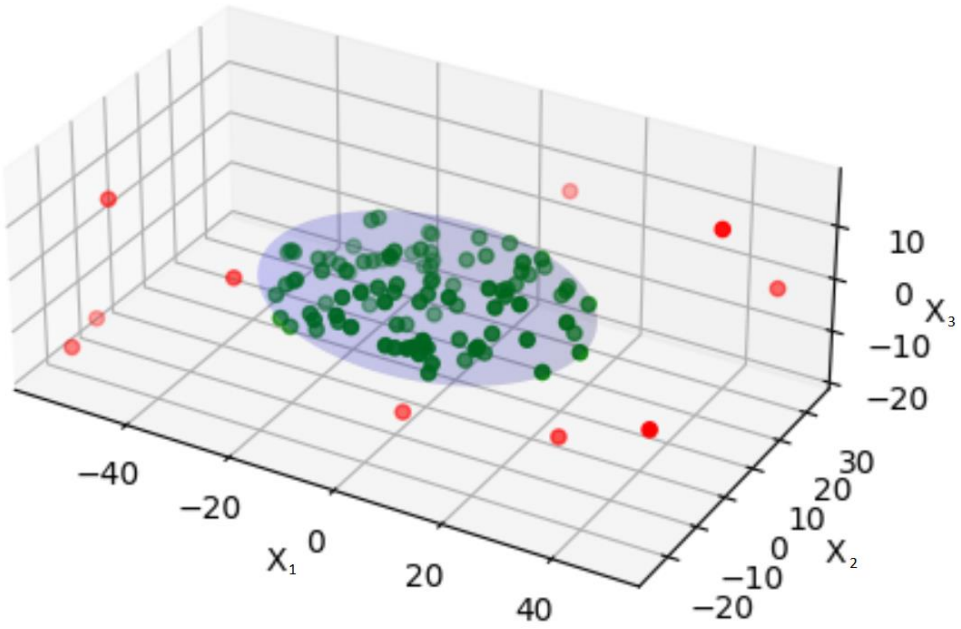

Figure 2: Example of a 3-variate prediction ellipsoid

Since the prediction ellipsoid relies on the assumption of data normality, the Mardia test was employed to evaluate the departure of the multivariate data distribution from normality.

$$\beta_1 = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\{(X_i - \bar{X})^T S_X^{-1}(X_j - \bar{X})\}^3; \quad (3)$$

$$\beta_2 = \frac{1}{N}\sum_{j=1}^{N}\{(X_j - \bar{X})^T S_X^{-1}(X_j - \bar{X})\}^2. \quad (4)$$

It is a statistical test used to assess whether a dataset follows a multivariate normal distribution. It measures the multivariate skewness β_1 (3) and multivariate kurtosis β_2 (4) of the data to determine how closely it resembles a normal distribution in multiple dimensions. This test is particularly useful in various fields such as statistics, econometrics, and multivariate analysis, where the assumption of multivariate normality is important for accurate interpretation and analysis of data.

Based on the results of the Mardia test, it is evident that the multivariate distribution of the training sample deviates from Gaussian. This is indicated by the test statistic for multivariate skewness $N\beta_1/6$, which measures 289.20, surpassing the quantile of the Chi-Square distribution set at 277.77 for 220 degrees of freedom and a significance level of 0.005. Conversely, the test statistic for multivariate kurtosis β_2 with a value of 122.35 does not exceed the Gaussian distribution quantile, which stands at 127.97 for a mean of 120, a variance of 9.6, and a significance level of 0.005. Consequently, it is imperative to apply a normalizing transformation like the Box-Cox transformation on a non-Gaussian random vector $X = \{X_1, X_2, \dots, X_{10}\}^T$, to convert it into a Gaussian random vector $Z = \{Z_1, Z_2, \dots, Z_{10}\}^T$.

The Box-Cox transformation is a statistical technique used to stabilize variance and make data conform more closely to the assumptions of normality. It is typically applied to univariate data, but there is also a multivariate extension known as the multivariate Box-Cox transformation.

$$Z_j = x(\lambda_j) = \begin{cases} (X_j^{\lambda_j} - 1)/\lambda_j, & \lambda_j \neq 0; \\ \ln(X_j), & \lambda_j = 0. \end{cases} \quad (5)$$

In the multivariate context, the BCT aims to normalize data across multiple variables simultaneously. It is designed to address situations where there are multiple correlated variables

and normalizing each variable individually may not be sufficient to achieve multivariate normality. The multivariate BCT involves estimating transformation parameters for each variable in the sample, similar to the univariate case. However, it also considers the interrelationships between variables to ensure that the transformation maintains the structure of the multivariate distribution. One of the methods to determine parameters for each variable in the dataset involves maximizing the log-likelihood of the transformed data:

$$l(X,\theta) = \sum_{j=1}^{k}(\lambda_j - 1)\sum_{i=1}^{N}\ln(x_{ji}) - \frac{N}{2}\ln[\det(S_Z)], \quad (6)$$

where S_Z is calculated as (2) using the normalized sample Z instead of X.

Following the completion of the task utilizing the maximum likelihood method of the logarithmic function (6), the ensuing parameter estimates were acquired: $\widehat{\lambda_1}$ = -0.5799, $\widehat{\lambda_2}$ = -0.9732, $\widehat{\lambda_3}$ = 0.4871, $\widehat{\lambda_4}$ = 2.888, $\widehat{\lambda_5}$ = 4.0714, $\widehat{\lambda_6}$ = -0.231, $\widehat{\lambda_7}$ = 0.087, $\widehat{\lambda_8}$ = -1.2973, $\widehat{\lambda_9}$ = -1.1866, $\widehat{\lambda_{10}}$ = 1.3321.

As a result of the BCT application with components (5), a sample was obtained, featuring the resulting vector of means: $\bar{Z}$ = {0.52627; 0.13911; -0.91083; 0.25654; -0.24379; 1.13224; -1.14383; -0.26329; 2.03304; -0.34494}.

The covariance matrix of the normalized sample S_Z is:

$$\begin{bmatrix}
0.0^2 44 & 0.0^2 13 & -0.0^2 2 & -0.0^4 6 & -0.0^5 6 & -0.0^3 8 & -0.0^2 3 & 0.0^4 70 & 0.0^2 62 & -0.0^2 2 \\
0.0^2 13 & 0.0^2 11 & -0.0^4 2 & 0.0^4 77 & -0.0^5 3 & 0.0^3 17 & 0.0^2 11 & -0.0^4 4 & 0.0^2 28 & 0.0^3 13 \\
-0.0^2 2 & -0.0^4 2 & 0.0^2 37 & 0.0^3 36 & 0.0^6 51 & 0.0^2 12 & 0.0^2 38 & -0.0^3 4 & -0.0^2 2 & 0.0^2 19 \\
-0.0^4 6 & 0.0^4 77 & 0.0^3 36 & 0.0^4 66 & -0.0^6 3 & 0.0^3 23 & 0.0^3 38 & -0.0^3 2 & -0.0^3 3 & 0.0^3 12 \\
-0.0^5 6 & 0.0^5 28 & 0.0^6 51 & 0.0^6 27 & 0.0^7 48 & 0.0^5 16 & 0.0^5 21 & -0.0^6 4 & -0.0^4 2 & 0.0^5 17 \\
-0.0^3 8 & 0.0^3 17 & 0.0^2 12 & 0.0^3 23 & 0.0^5 17 & 0.0^2 85 & 0.0^2 27 & 0.0^3 12 & -0.0^2 1 & 0.0^4 89 \\
-0.0^2 3 & 0.0^2 11 & 0.0^2 38 & 0.0^3 38 & 0.0^5 21 & 0.0^2 27 & 0.0105 & -0.0^3 9 & -0.0^2 3 & 0.0^2 38 \\
0.0^4 70 & -0.0^4 4 & -0.0^3 3 & -0.0^3 2 & -0.0^6 4 & 0.0^3 12 & -0.0^3 9 & 0.0^2 71 & 0.0139 & 0.0^2 17 \\
0.0^2 62 & 0.0^2 28 & -0.0^2 2 & -0.0^3 3 & -0.0^4 2 & -0.0^2 1 & -0.0^2 3 & 0.0139 & 0.0857 & 0.0^2 12 \\
-0.0^2 2 & 0.0^3 13 & 0.0^2 19 & 0.0^3 12 & 0.0^5 17 & 0.0^4 89 & 0.0^2 38 & 0.0^2 17 & 0.0^2 12 & 0.0^2 31
\end{bmatrix}$$

The normalized sample shows no departure from the multivariate normal distribution. This is supported by the test statistics: the multivariate skewness test statistic $N\beta_1/6$, measuring at 265.59 does not exceed the critical value of 277.77, and the multivariate kurtosis test statistic β_2, recorded at 121.52, is lower than the critical value of 127.97.

After applying the normalization transformations, a ten-variate prediction ellipsoid is constructed based on equation (1):

$$(Z-\bar{Z})^T S_Z^{-1}(Z-\bar{Z}) \leq \chi^2_{10,\ 0.005}. \quad (7)$$

The quantile value of the Chi-square distribution is 25.19 for 10 degrees of freedom and a significance level of 0.005.

4.2. Machine learning algorithms

Several popular techniques in machine learning serve as effective tools for outlier detection and can be used for face recognition, including a one-class support vector machine, isolation forest, and autoencoder. These methods are particularly notable for their application in unsupervised learning and they can be trained only on data from the target class, studying the features of the provided data.

4.2.1. One-class support vector machine

OCSVM constructs a decision function or boundary that effectively separates the target data from the rest of the feature space. This boundary is established by finding a hyperplane with a maximum margin and optimizing the distance between the hyperplane and the origin in the high-dimensional feature space.

In an OCSVM, an implicit transformation function $\varphi(\cdot)$ is employed, it is a non-linear projection that is evaluated through a kernel function, which serves as a mapping from the original feature space to a potentially higher-dimensional feature space: $k(x,y) = \varphi(x)\cdot\varphi(y)$ [17]. Commonly used kernel functions include: the linear kernel, which computes dot products in the original

feature space and is suitable for linearly separable data; the polynomial kernel captures non-linear relationships between data points by raising dot products to specified powers, allowing for the modeling of complex decision boundaries; the radial basis function kernel, employing a Gaussian function, is highly effective in capturing intricate relationships between data points, particularly in scenarios where data is not linearly separable; lastly, the sigmoid kernel, based on the hyperbolic tangent function, adeptly captures non-linear patterns in the data, making it useful for handling complex relationships between features and classes.

Subsequently, the algorithm learns a decision boundary aimed at segregating the majority of the data from the origin, it is defined as [18]:

$$g(x) = \omega^T \varphi(x) - \rho,$$

where ω the normal vector of the hyperplane, ρ is the bias.

Formulating OCSVM as a quadratic optimization problem, the main aim is to minimize the weight vector while simultaneously maximizing the margin, subject to specific constraints:

$$min_{\omega,\xi,\rho} \frac{||\omega||^2}{2} - \rho + \frac{1}{\nu N}\sum_{i=1}^{N} \xi_i ,$$

$$\text{subject to: } \omega^T \varphi(x_i) \geq \rho - \xi_i, \xi_i \geq 0,$$

where ξ_i is slack variables, it is utilized to model separation errors; $\nu \in (0,1]$ is the regularization parameter, it characterizes the solution by setting an upper bound on the fraction of outliers, such as the training examples regarded as out-of-class, and simultaneously establishes a lower bound on the number of training examples utilized as support vectors [19].

The optimization problem is typically solved in its dual form, resulting in a decision function that can classify new data points as either belonging to the target class or representing anomalies. This function yields a positive value for the target and a negative value otherwise:

$$f(x) = sgn\big(g(x)\big).$$

In Python, implementing OCSVM typically involves creating a OneClassSVM object from the SVM module in sci-kit-learn. This allows users to adjust essential parameters such as the choice of kernel function, regularization parameter ν, and kernel-specific parameters. The model is configured with a ν parameter of 0.15, which determines the fraction of training errors and the upper bound on the fraction of training set outliers. For the kernel function, the radial basis function is utilized. This kernel is popular for SVMs due to its flexibility in capturing non-linear relationships between data points. Additionally, the gamma parameter is set to auto, which means its value is automatically calculated based on the inverse of the number of features. Gamma defines the influence range of a single training example, with low values meaning far and high values meaning close.

4.2.2. Isolation forest

Departing from traditional approaches reliant on profiling normal data points, isolation forest takes a unique route by directly focusing on isolating anomalies. This method hinges on constructing isolation trees, binary trees where internal nodes represent features and split values, while leaf nodes represent individual data points [20].

The process of constructing isolation trees initiates with the random selection of a feature and split value within its range. This randomness recurs until each data point is secluded in its own leaf node, or until a predetermined maximum tree depth is attained. The method's ability to efficiently isolate anomalies sans prior knowledge of the dataset's distribution stems from this stochastic feature and split value selection. Anomalies, expected to be inherently easier to isolate than normal data points, typically reside in sparser regions of the feature space. Consequently, they necessitate fewer splits along paths from the root to leaf nodes for isolation. Hence, the average path length from the root to leaf nodes for each data point is computed across all trees within the forest [21].

Subsequently, an anomaly score is computed for each data point based on its average path length [22]:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}},$$

where $E(h(x)) = \frac{\sum_{i=1}^{t} h_i(x)}{t}$ is the average length of x over t isolation trees; $c(n)$ is the average length of unsuccessful search in a binary tree:

$$c(n) = 2H(n-1) - (2(n-1)/n),$$

where $H(i) = \ln(i) + \gamma$, γ is Euler's constant.

Data points with shorter path lengths, indicative of proximity to the root, are deemed more likely to be anomalies, while those with longer paths are considered more typical. A threshold is established to classify data points as anomalies or normal points based on their anomaly scores, with points above the threshold classified as anomalies and those below as normal.

Isolation forest harnesses an ensemble of these isolation trees, where each tree operates independently, contributing to the final anomaly score for a given data point. This ensemble approach enhances the algorithm's robustness and accuracy, rendering it more adept at handling noise and data variability.

The IsolationForest algorithm is implemented in the widely-used Python machine learning library sci-kit-learn and offers various adjustable parameters for optimizing its performance. Among these is the contamination parameter, essential for determining the threshold in the decision function, which distinguishes new data points as either target or anomalous [23]. After conducting experiments, a value of 0.08 was selected for the contamination parameter as it strikes the optimal balance between detecting true anomalies and minimizing false alarms.

Other significant parameters of the IsolationForest model include: n_estimators, specifying the number of decision trees within the forest, set to 100; max_samples, indicating the maximum number of samples used for training each tree, set to 256; and max_features, determining the maximum number of features used for splitting each node, set to 1.0, indicating that all features are used. These parameter settings were chosen based on their widespread usage and their ability to strike an optimal balance between model complexity and computational efficiency.

To classify a sample as either target or anomalous, the anomaly score is compared to a threshold value. These scores can range from positive to negative, with negative values indicating samples more likely to be the target and positive values indicating samples more likely to be anomalous. The selection of the threshold value depends on the specific application and the characteristics of the data. In our case, a threshold value of 0 yielded the best results.

4.2.3. Autoencoder

An autoencoder is a form of artificial neural network employed for learning efficient data representations, dimensionality reduction, and detecting anomalies. It is an unsupervised learning method composed of two primary components: an encoder and a decoder. The main objective of an autoencoder is to learn a condensed and meaningful representation of the input data, referred to as the latent space or bottleneck, by reconstructing the input data with minimal loss [24].

The encoder's role is to map the input data to the latent space, compressing the information into a lower-dimensional representation. It is generally a neural network with multiple layers, where each layer applies a non-linear transformation to the input data. The encoder's output, the latent space, captures the most significant features and patterns in the input data.

In contrast, the decoder is responsible for reconstructing the input data from the latent space representation. It is also a neural network with multiple layers, but its architecture is typically the inverse of the encoder's architecture. The decoder takes the latent space representation as input and tries to recreate the original input data by applying a series of non-linear transformations through its layers.

During training, the autoencoder learns to minimize the reconstruction error, which is the discrepancy between the original input data and the reconstructed data. This is usually accomplished by optimizing a loss function, such as mean squared error or binary cross-entropy, using gradient-based optimization techniques like backpropagation.

In the context of recognition, the autoencoder is trained using target class instances, enabling it to learn the patterns and structure of normal data. When presented with new data, the autoencoder will reconstruct the input with a low reconstruction error if the input is the target. However, if the input is an anomaly, the reconstruction error will be higher, as the autoencoder

has not been trained to reconstruct such instances effectively [25]. By setting an appropriate threshold on the reconstruction error, it is possible to detect and distinguish anomalies from target instances.

When considering the development of autoencoder models, various machine learning frameworks present distinct advantages. TensorFlow and PyTorch emerge as prominent contenders in this domain. TensorFlow, originating from Google, offers a rich ecosystem of tools and libraries, rendering it a favored option among both researchers and practitioners. Its utilization of a static computation graph facilitates efficient optimization and distributed computing, making it particularly suited for handling large-scale neural network models. Conversely, PyTorch, hailing from Facebook's AI Research lab, emphasizes flexibility and user-friendliness. With its dynamic computation graph and intuitive API, PyTorch becomes the preferred choice for swift prototyping and experimentation [26].

In the creation of autoencoder models, TensorFlow and Keras were selected due to their collective strengths in flexibility, scalability, and usability. Keras, functioning as a high-level neural networks API layered atop TensorFlow, streamlines the process of constructing and training autoencoder models. Meanwhile, TensorFlow furnishes the foundational infrastructure for proficient computation, ensuring optimal performance throughout the training and inference phases.

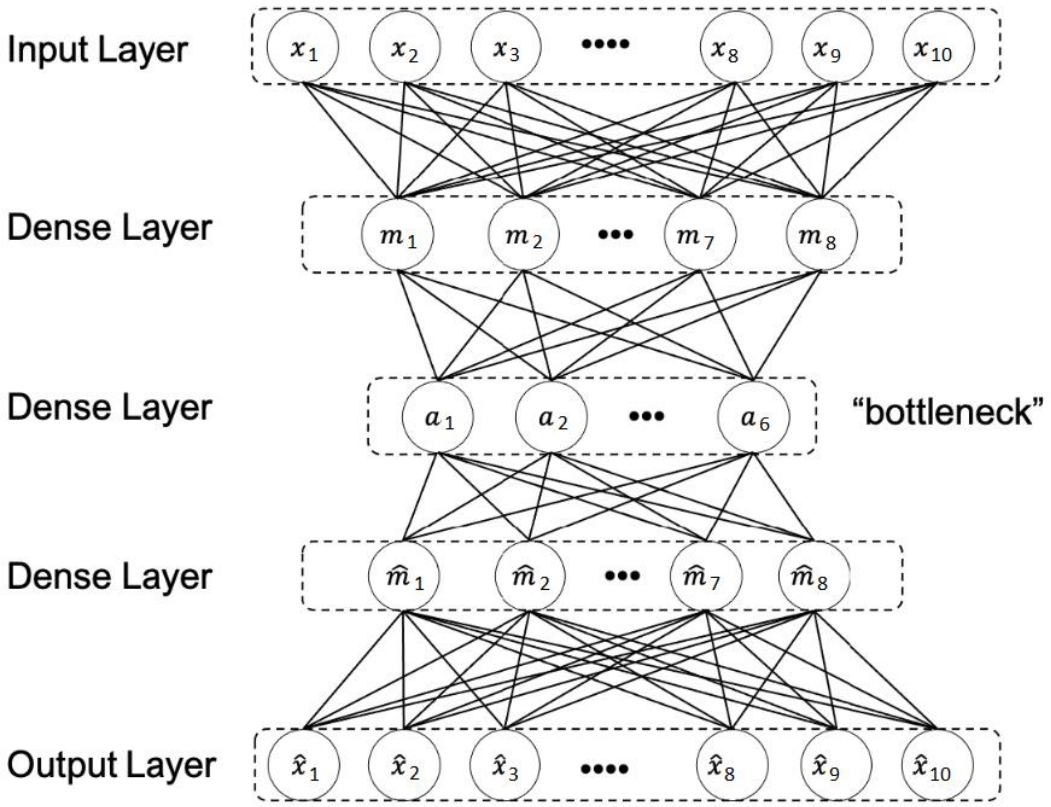

Figure 3: Autoencoder structure

Before feeding the data into the neural network, min-max normalization is applied to each feature individually. This normalization method scales each feature to a range [0, 1] [27]. By applying min-max normalization, all features are brought to the same scale and are constrained within the desired range, promoting stable and efficient learning processes. The autoencoder model is designed with an input layer that receives data in the shape of a 10-variate representation. The model consists of a series of fully connected layers for both encoding and decoding operations. The encoding phase compresses the input data into a lower-dimensional representation, gradually reducing the dimensionality from 10 to 8 and then further down to 6, forming a bottleneck in the network architecture [28]. This bottleneck layer restricts the model's capacity to capture the most salient features of the input data, forcing it to learn a compact representation that captures essential information. Each encoding layer utilizes rectified linear

unit (ReLU) activation functions, which introduce non-linearity to the model and facilitate the extraction of complex features from the input data. Following the encoding layers, the decoding phase reverses the process, reconstructing the original input dimensions. The decoded layers expand the dimensionality back to 8 and finally, to the original 10 dimensions. Similar to the encoding layers, ReLU activation functions are employed in the decoding layers, preserving the non-linear relationships learned during the encoding phase. The final layer of the model uses a sigmoid activation function, ensuring that the output values are within the range [0, 1]. This activation function is commonly used for binary classification tasks and reconstruction problems, providing smooth and interpretable output. The autoencoder structure (fig. 3) is similar to [29].

To train the model, the Adam optimizer is utilized with binary cross-entropy loss, a common choice for reconstruction tasks aiming to minimize the difference between the input and reconstructed data. Adam optimizer is an adaptive learning rate optimization approach that combines the advantages of both AdaGrad and RMSProp [30]. It dynamically adjusts the learning rate during training, allowing for faster convergence and improved performance. Binary cross-entropy loss, on the other hand, measures the dissimilarity between the input and the reconstructed data, particularly suitable for binary classification problems where each instance belongs to one of two classes. This configuration ensures efficient training of the autoencoder model, optimizing its ability to capture the underlying patterns in the data while minimizing reconstruction error.

The training process spans 100 epochs, with a batch size of 8 instances per batch. Shuffling the data at each epoch introduces randomness and prevents the model from memorizing the training data order, aiding in generalization.

4.3. Evaluation metrics

Various evaluation metrics are employed to assess the effectiveness of created models in recognition of the target instances or anomalies. Commonly used metrics include accuracy, precision, recall (sensitivity), specificity, and the $F1$ score [31, 32]. While accuracy provides a broad perspective on correctness, precision, recall, specificity, and the $F1$ score offers nuanced insights into the model's efficacy in detecting anomalies.

The evaluation metrics are derived from a confusion matrix, which offers four distinct measures. True Positive (TP) represents instances where the model correctly identifies an anomaly as such. False Positive (FP) instances occur when the model inaccurately labels a target instance as an anomaly. True Negative (TN) denotes instances where the model accurately identifies target instances. Lastly, False Negative (FN) instances arise when the model incorrectly labels target instances as normal.

Accuracy acts as a gauge for overall correctness, derived from the ratio of correctly classified instances to the total number of instances :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}.$$

Precision, on the other hand, quantifies the proportion of correctly identified anomalies among all instances classified as anomalies. This metric is calculated as the ratio of true positives to the sum of true positives and false positives:

$$Precision = \frac{TP}{TP + FP}.$$

Recall, also termed sensitivity, represents the proportion of actual anomalies correctly identified by the model. It serves as a pivotal indicator of the model's ability to detect all anomalies within the dataset:

$$Recall = \frac{TP}{TP + FN}.$$

Specificity provides complementary insights by measuring the proportion of true negatives and correctly predicted target instances, among all actual target instances. It highlights the model's capacity to accurately identify target instances while minimizing false alarms.

$$Specificity = \frac{TN}{TN + FP}.$$

The $F1$ score is the harmonic mean of precision and recall:

$$F1\ score = 2 * \frac{Precision * Recall}{Precision\ +\ Recall}.$$

5. Results

Table 1 presents a comparison of recognition performance of prediction ellipsoid for non-Gaussian data (PENGD) (1), prediction ellipsoid for normalized data (PEND) (7), one-class support vector machine (OCSVM), isolation forest (IF) and autoencoder (AE).

Table 1
Comparison of models

Model	Precision	Recall	Specificity	*F*1 score	Accuracy
PENGD	0.9598	0.9550	0.9200	0.9574	0.9433
PEND	0.9848	0.9750	0.9700	0.9793	0.9733
OCSVM	0.8878	0.9100	0.7700	0.8988	0.8633
IF	0.9393	0.9300	0.8800	0.9347	0.9133
AE	0.9366	0.9600	0.8700	0.9481	0.9300

In general, all developed models have good efficiency. The OCSVM exhibited the lowest performance among the models evaluated, with scores generally lower across precision, recall, specificity, *F*1 score, and accuracy. The IF demonstrated competitive performance across most metrics, falling between the scores of OCSVM and the other models. This indicates that IF may offer a decent compromise between simplicity and performance, making it a viable choice depending on the specific requirements of the application.

AE showed performance comparable to PENGD, with scores across various metrics similar to or slightly lower than PENGD. However, by employing normalization techniques, PEND achieved notable enhancements in performance compared to both AE and PENGD. This underscores the importance of normalizing transformation in enhancing prediction ellipsoid models for recognition, particularly in scenarios involving non-Gaussian data.

6. Discussion

As evidenced by the results, the prediction ellipsoid for normalized data consistently outperformed machine learning models such as OCSVM, IF, and AE. Compared to the prediction ellipsoid for non-Gaussian data, the application of a ten-variate Box-Cox transformation significantly improved the model. This highlights the importance of applying a normalizing transformation to data with a non-Gaussian distribution. The choice of an appropriate normalization technique is far from trivial and exerts a profound influence on the efficacy and robustness of the created model. Univariate normalization methods often fall short of capturing the intricate relationships between features, leading to suboptimal results. In contrast, multivariate normalization techniques offer a more comprehensive approach by considering the correlations and interdependencies between multiple features. By applying the ten-variate Box-Cox transformation, the prediction ellipsoid can account for correlations between features. This approach enables the model to more accurately represent the underlying data distribution, leading to enhanced robustness and efficiency.

It should be noted that a crucial aspect in the construction of a prediction ellipsoid is the definition of the confidence level. In our study, a confidence level of 0.005 was selected, aligning with its widespread usage in outlier detection tasks [33].

The main advantage of using the prediction ellipsoid for normalized data lies in its ability to enhance prediction accuracy and reliability. Normalizing the data not only facilitates the development of more accurate prediction ellipsoid but also ensures they capture the underlying patterns and relationships within the data more effectively, leading to more reliable and precise predictions. Additionally, by transforming the data to data with the Gaussian distribution, the prediction ellipsoid gains increased robustness and resilience.

The application of prediction ellipsoid for normalized data has some disadvantages. Firstly, a substantial dataset size is often required to robustly study the model's performance, typically necessitating a minimum of 100 instances to yield meaningful insights. Additionally, the process of identifying an appropriate normalizing transformation can be intricate and resource-intensive, particularly in cases where the underlying data distribution is complex or poorly understood. The last one is the necessity of selecting a confidence level, which can have a significant impact on the interpretation and reliability of the prediction ellipsoid.

It's important to acknowledge the inherent limitations of the prediction ellipsoid model. One notable constraint is its inherent design to recognize and delineate prediction regions for individual instances rather than accommodating multiple instances simultaneously. While the model excels in providing localized predictions for individual data points, its applicability to scenarios involving collective analysis or batch processing may be limited, warranting careful consideration in practical deployment scenarios.

Future research could involve exploring alternative normalizing transformations, such as the multivariate Johnson transformation. In addition, it's important to consider the impact of model complexity and feature richness on performance. While the models evaluated in this study demonstrate commendable results with the available features, there's a clear indication that employing more complex models and incorporating a broader range of features could yield even better outcomes. This emphasizes the necessity of advancing both model architecture and feature selection techniques to effectively address the intricacies present in the data.

7. Conclusions

The study examined various one-class classification methods in the context of face recognition. Various models were constructed including prediction ellipsoid for normalized data and machine learning algorithms such as OCSVM, IF, and AE. These methodologies aim to unveil anomalies within a dataset by apprehending its underlying structure, often gleaned from instances representing the target class.

The application of the ten-variate Box-Cox transformation significantly enhanced the performance of the prediction ellipsoid, which led to a 3% increase in recognition accuracy. Consequently, the model surpassed the performance of the considered machine learning methods. This underscores the significance of applying normalization techniques to non-Gaussian data. The primary advantage of utilizing the prediction ellipsoid for normalized data is its capacity to enhance prediction accuracy and reliability. Normalizing the data not only facilitates the development of more precise prediction ellipsoids but also ensures they capture the underlying patterns.

The main disadvantage of the developed model lies in the complexity and resource-intensive nature of determining the appropriate normalizing transformation. This challenge is particularly pronounced in cases where the underlying data distribution is complex or poorly understood, especially when dealing with high-dimensional data.

An essential aspect of employing the method involves determining the confidence level, with a confidence level of 0.005 utilized in the study.

Future research could explore the impact of employing alternative normalizing methods, such as the multivariate Johnson transformation, to further enhance model performance. Additionally, deeper investigations into the reasons behind the observed performance differences among models and the potential benefits of incorporating more complex models with a broader array of features could contribute to the advancement of one-class classification methodologies in face recognition tasks.

References

[1] Y. Kortli, M. Jridi, A. Al Falou, M. Atri, Face Recognition Systems: A Survey. Sensors. 2020 20(2):342. doi:10.3390/s20020342.

[2] H. Marques, L. Swersky, J. Sander, R. Campello, A. Zimek, On the evaluation of outlier detection and one-class classification: a comparative study of algorithms, model selection,

and ensembles. Data Min Knowl Disc 37, 1473–1517, 2023. doi: 10.1007/s10618-023-00931-x.
[3] V. Bezerra V, V. da Costa, S. Barbon Junior, R. Miani, B. Zarpelão, IoTDS: A One-Class Classification Approach to Detect Botnets in Internet of Things Devices. Sensors. 2019; 19(14):3188. doi:10.3390/s19143188.
[4] S. Kim, D. Park, J. Jung, Evaluation of One-Class Classifiers for Fault Detection: Mahalanobis Classifiers and the Mahalanobis–Taguchi System. Processes. 2021; 9(8):1450. doi:10.3390/pr9081450.
[5] A. Nassif, M. Talib, Q. Nasir, F. Dakalbab, Machine learning for anomaly detection: A systematic review, in: IEEE Access, 9, 78658-78700. 2021. doi:10.1109/ACCESS.2021.3083060.
[6] L. Li, X. Mu, S. Li and H. Peng, A Review of Face Recognition Technology, in: IEEE Access, vol. 8, pp. 139110-139120, 2020. doi: 10.1109/ACCESS.2020.3011028.
[7] P. Perera, P. Oza, V. Patel, One-Class Classification: A Survey. ArXiv. 2021. doi: 10.48550/arXiv.2101.03064.
[8] N. Seliya, A. Abdollah Zadeh, M. Taghi, A literature review on one-class classification and its potential applications in big data. Journal of Big Data 8, 2021: 1-31. doi: 10.1186/s40537-021-00514-x.
[9] N. Damer, J. H. Grebe, S. Zienert, F. Kirchbuchner, A. Kuijper, On the Generalization of Detecting Face Morphing Attacks as Anomalies: Novelty vs. Outlier Detection, 2019 IEEE 10th International Conference on Biometrics Theory, Applications and Systems (BTAS), Tampa, FL, USA, 2019, pp. 1-5. doi: 10.1109/BTAS46853.2019.9185995.
[10] H. Xu, G. Pang, Y. Wang, Y. Wang, Deep Isolation Forest for Anomaly Detection, in: IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 12, pp. 12591-12604, 1 Dec. 2023, doi: 10.1109/TKDE.2023.3270293
[11] T. Vaiyapuri, A. Binbusayyis, Application of deep autoencoder as an one-class classifier for unsupervised network intrusion detection: a comparative evaluation. PeerJ Computer Science 6:e327, 2020. doi:10.7717/peerj-cs.327.
[12] P. Oza and V. M. Patel, Active Authentication using an Autoencoder regularized CNN-based One-Class Classifier,"2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019), Lille, France, 2019, pp. 1-8. doi: 10.1109/FG.2019.8756525.
[13] S. Prykhodko, A. Prykhodko, I. Shutko, Estimating the Size of Web Apps Created Using the CakePHP Framework by Nonlinear Regression Models with Three Predictors, in: 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT). Vol. 1. IEEE, 2021. doi: 10.1109/CSIT52700.2021.9648680
[14] S. Prykhodko, L. Makarova, K. Prykhodko, A. Pukhalevych, Application of Transformed Prediction Ellipsoids for Outlier Detection in Multivariate Non-Gaussian Data, in: 2020 IEEE 15th Inter-national Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), pp 359-362, 2020. doi: 10.1109/TCSET49122.2020.235454.
[15] J. Avanija, K. Madhavi, G. Sunitha, S. Sangapu, S. Raju, Facial Expression Recognition using Convolutional Neural Network, in: 2022 First International Conference on Artificial Intelligence Trends and Pattern Recognition (ICAITPR), pp. 1-7, 2022. doi: 10.1109/ICAITPR51569.2022.9844221.
[16] T. Etherington, Mahalanobis distances for ecological niche modelling and outlier detection: implications of sample size, error, and bias for selecting and parameterising a multivariate location and scatter method. PeerJ, 9. 2021. doi: 10.7717/peerj.11436.
[17] R. Ghiasi, M. A. Khan, D. Sorrentino, C. Diaine, A. Malekjafarian, An unsupervised anomaly detection framework for onboard monitoring of railway track geometrical defects using one-class support vector machine. Engineering Applications of Artificial Intelligence, 133, 108167, 2024. doi:10.1016/j.engappai.2024.108167.
[18] S. Todkar, V. Baltazart, A. Ihamouten, X. Dérobert, D. Guilbert, One-class SVM based outlier detection strategy to detect thin interlayer debondings within pavement structures using Ground Penetrating Radar data. Journal of Applied Geophysics, 192, 104392, 2021. doi: 10.1016/j.jappgeo.2021.104392.

[19] T. Duong, NS. Vo, L. Nguyen, QT. Vien, VD. Nguyen, Anomaly Detection Using One-Class SVM for Logs of Juniper Router Devices. Industrial Networks and Intelligent Systems. INISCOM 2019. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 293. Springer, Cham. 2019. doi:10.1007/978-3-030-30149-1_24.
[20] J. Lesouple, C. Baudoin, M. Spigai, JY. Tourneret, Generalized isolation forest for anomaly detection, Pattern Recognition Letters, Volume 149, 2021, Pages 109-119, ISSN 0167-8655. doi:10.1016/j.patrec.2021.05.022.
[21] Y. Chabchoub, M. U. Togbe, A. Boly, R. Chiky, An In-Depth Study and Improvement of Isolation Forest, in: IEEE Access, vol. 10, pp. 10219-10237, 2022. doi: 10.1109/ACCESS.2022.3144425.
[22] S. Imashev, Extended isolation forest–application to outlier detection in geomagnetic data, in: IOP Conference Series: Earth and Environmental Science. Vol. 929, No. 1, p. 012022. 2021, November. doi: 10.1088/1755-1315/929/1/012022.
[23] M. U. Togbe, M. Barry, A. Boly, Y. Chabchoub, R. Chiky, J. Montiel, T.V. Tran, Anomaly detection for data streams based on isolation forest using scikit-multiflow, in: Computational Science and Its Applications–ICCSA 2020: 20th International Conference, Cagliari, Italy, July 1–4, 2020, Proceedings, Part IV 20 (pp. 15-30). Springer International Publishing. doi: 10.1007/978-3-030-58811-3_2.
[24] W. Xu, J. Jang-Jaccard, A. Singh, Y. Wei, F. Sabrina, Improving Performance of Autoencoder-Based Network Anomaly Detection on NSL-KDD Dataset, in: IEEE Access, vol. 9, pp. 140136-140146, 2021, doi: 10.1109/ACCESS.2021.3116612.
[25] Z. Wang, Y-J. Cha, Unsupervised deep learning approach using a deep auto-encoder with a one-class support vector machine to detect damage. Structural Health Monitoring. 2021; 20(1):406-425. doi:10.1177/1475921720934051.
[26] O-C. Novac, MC. Chirodea, CM. Novac, N. Bizon, M. Oproescu, OP. Stan, CE. Gordan, Analysis of the Application Efficiency of TensorFlow and PyTorch in Convolutional Neural Network. Sensors. 2022; 22(22):8872. doi:10.3390/s22228872.
[27] B. Min, J. Yoo, S. Kim, D. Shin, D. Shin, Network Anomaly Detection Using Memory-Augmented Deep Autoencoder, in: IEEE Access, vol. 9, pp. 104695-104706, 2021. doi: 10.1109/ACCESS.2021.3100087
[28] M. Sewak, S. K. Sahay, H. Rathore, An overview of deep learning architecture of deep neural networks and autoencoders. Journal of Computational and Theoretical Nanoscience, 17(1), 182-188. 2020. doi: 10.1166/jctn.2020.8648.
[29] W. Xu, J. Jang-Jaccard, A. Singh, Y. Wei, F. Sabrina, Improving Performance of Autoencoder-Based Network Anomaly Detection on NSL-KDD Dataset, in: IEEE Access, vol. 9, pp. 140136-140146, 2021. doi: 10.1109/ACCESS.2021.3116612.
[30] I. H. Kartowisastro, J. Latupapua, A Comparison of Adaptive Moment Estimation (Adam) and RMSProp Optimisation Techniques for Wildlife Animal Classification Using Convolutional Neural Networks. Revue d'Intelligence Artificielle, Vol. 37, No. 4, pp. 1023-1030. 2023. doi: 10.18280/ria.370424.
[31] W. Hilal, S. A. Gadsden, J. Yawney, Financial Fraud: A Review of Anomaly Detection Techniques and Recent Advances. Expert Systems with Applications. Volume 193, 2022, 116429, ISSN 0957-4174. doi:10.1016/j.eswa.2021.116429.
[32] S. Nurmaini, A. Darmawahyuni, AN. Sakti Mukti, MN. Rachmatullah, F. Firdaus, B. Tutuko, Deep Learning-Based Stacked Denoising and Autoencoder for ECG Heartbeat Classification. Electronics. 2020; 9(1):135. doi:10.3390/electronics9010135.
[33] S. Prykhodko, N. Prykhodko, L. Makarova, K. Pugachenko, Detecting outliers in multivariate non-Gaussian data on the basis of normalizing transformations, in: Proceedings of the First Ukraine Conference on Electrical and Computer Engineering (UKRCON), IEEE, 2017, pp. 846–849. doi:10.1109/UKRCON.2017.8100366.