Expert Systems With Applications 217 (2023) 119562

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A novel combined approach based on deep Autoencoder and deep classifiers for credit card fraud detection

Hosein Fanai, Hossein Abbasimehr [*]

Faculty of Information Technology and Computer Engineering, Azarbaijan Shahid Madani University, Tabriz, Iran

ARTICLE INFO

Keywords:
Fraud detection
Representation learning
Autoencoder
ANN
RNN

ABSTRACT

Due to the growth of e-commerce and online payment methods, the number of fraudulent transactions has increased. Financial institutions with online payment systems must utilize automatic fraud detection systems to reduce losses incurred due to fraudulent activities. The problem of fraud detection is often formulated as a binary classification model that can distinguish fraudulent transactions. Embedding the input data of the fraud dataset into a lower-dimensional representation is crucial to building robust and accurate fraud detection systems. This study proposes a two-stage framework to detect fraudulent transactions that incorporates a deep Autoencoder as a representation learning method, and supervised deep learning techniques. The experimental evaluations revealed that the proposed approach improves the performance of the employed deep learning-based classifiers. Specifically, the utilized deep learning classifiers trained on the transformed data set obtained by the deep Autoencoder significantly outperform their baseline classifiers trained on the original data in terms of all performance measures. Besides, models created using deep Autoencoder outperform those created using the principal component analysis (PCA)-obtained dataset as well as the existing models.

1. Introduction

In recent years, due to the growth of e-commerce and the use of online payments, the number of fraudulent transactions has increased. According to credible Reports,[1] fraud losses related to credit and debit cards have increased exponentially between 2000 and 2015. It has also been shown that purchases made without authorization and counterfeit credit cards make up 10–15 % of total fraud cases, but account for 75–80 % of financial value (Saia & Carta, 2019). In response to these challenges, private and public entities have increased investments in research and development, aiming to design more and more effective fraud detection systems.

The use of automated fraud detection systems is vital for financial institutions that issue credit cards or manage online transactions. This reduces losses and also increases customer trust. With the rise of big data and artificial intelligence, new opportunities have arisen in using advanced machine learning models to detect fraud (Bao, Hilary, & Ke, 2022). In fact, Current fraud detection systems based on advanced data mining and machine/deep learning methods are very effective. By having a data set including labeled transactions (normal and fraudulent), a binary classification model with the ability to distinguish fraudulent transactions from normal transactions is built. The built model is used to decide whether incoming transactions are normal or fraudulent. The problem of detecting fraudulent transactions using classification techniques involves various challenges (Correa Bahnsen, Aouada, Stojanovic, & Ottersten, 2016): (1) class imbalance (the ratio of fraudulent transactions to normal transactions is very small), (2) cost sensitivity, i.e., the costs of misclassifying the fraudulent and normal transactions are not equal, (3) temporal dependence between transactions, (4) the presence of concept drift, i.e., Class conditional distributions change over time, requiring the classifier to be updated, (5) dimensionality of search space feature preprocessing.

According to the comprehensive literature review conducted by Ngai, Hu, Wong, Chen, and Sun (2011), decision trees, artificial neural networks, logistic regression, and support vector machines have been used most frequently among supervised learning techniques.

Inspired by the successful application of the deep learning methods in many areas such as compute vision (Dev, Khowaja, Bist, Saini, & Bhatia, 2021; Sultana, Usha Rani, & Farquad, 2020; Y. Xue & Qin, 2022), translation (Luong et al., 2015), speech recognition (Chorowski,

[*] Corresponding author.
E-mail address: abbasimehr@azaruniv.ac.ir (H. Abbasimehr).
[1] Nilson Report: https://www.nilsonreport.com/.

https://doi.org/10.1016/j.eswa.2023.119562
Received 26 September 2022; Received in revised form 24 December 2022; Accepted 13 January 2023
Available online 18 January 2023

Bahdanau, Serdyuk, Cho, & Bengio, 2015), forecasting complex time series data (Abbasimehr & Paki, 2021; Abbasimehr, Shabani, & Yousefi, 2020), recently some studies have exploited different variants of recurrent neural networks (e.g., LSTM (Hochreiter & Schmidhuber, 1997) and GRU (Chung, Gulcehre, Cho, & Bengio, 2014)) to build credit card fraud detection systems (Forough & Momtazi, 2021, 2022; Jurgovsky et al., 2018). LSTM and GRU are kind of recurrent neural networks (RNNs) to resolve the gradient vanishing/exploding problem of RNNs. They designed to learn temporal information from sequential data (Tran, Iosifidis, Kanniainen, & Gabbouj, 2018). Deep learning models are attractive because they can learn representations from raw inputs. In many research fields, neural networks and deep learning algorithms perform far better than traditional algorithms, but these models are not yet extensively used in credit card fraud detection.

To build robust and accurate fraud detection systems, it is crucial to embed the input data of a fraud dataset in a lower dimensional representation. Through representation learning, low-dimensional embeddings can be derived from features in the original input to provide a more expressive data representation (Y. Wang, Yao, Zhao, & Zheng, 2015). Autoencoders, which are utilized in this study, have recently gained popularity among representation learning techniques for their ability to identify underlying patterns in input data before classification (Bengio, Courville, & Vincent, 2012). Specifically, an Autoencoder consists of an encoder and a decoder module. The encoder module compresses inputs and the decoder reconstructs inputs from the compressed versions (Fournier & Aloise, 2019).

In this study we propose a framework that combines unsupervised neural network, a deep Autoencoder as the dimensionality reduction technique with supervised deep learning methods to enhance the accuracy of fraud detection. The idea is to get a compact representation of data to accurately detect fraudulent transactions. Firstly, deep Autoencoder model is built using the original data. Then the obtained encoder model is used as the data preparation tool and is applied to the input data to make the new dataset with reduced number of features (i.e., new representation of data). The transformed data are applied to three deep neural networks including Deep ANN (DNN), Deep RNN, and a combined method of RNN and Convolutional neural network (Y. Xue, Wang, Liang, & Slowik, 2021) (RNN_CONV), as well as the SVM method. Bayesian optimization algorithm (Turner, Eriksson, McCourt, Kiili, Laaksonen, Xu, & Guyon, 2021) is used to find the best hyperparameters of all utilized models.

The results of the experimental evaluations revealed that the utilized deep learning classifiers trained on the transformed data set by the deep Autoencoder largely outperform their baseline classifiers trained on the original data in terms of all performance measures including Recall, Precision, and F1, resulting in an effective credit card fraud detection solution. Specifically, the results indicate that the proposed approach improves the performance of the DNN model in terms of all performance measures. Additionally, for RNN, the use of the data obtained through deep Autoencoder improves its performance regarding the utilized performance metrics. This similarly holds for the CNN_RNN model whose performance is enhanced compared to the case when the original data are used. Also, SVM trained with data of the deep Autoencoder achieves better results than the baseline SVM model with respect to all performance measures. Overall, all of the proposed Autoencoder-based models outperform their baseline models across all performance metrics demonstrating the effectiveness of the proposed approach. Additionally, we compared the models resulting from the proposed approach with models trained on the dataset derived via PCA. Compared to the (PCA)-obtained dataset, the models created by deep Autoencoder performed better.

In Summary, it should be mentioned the contributions of this study are as follows:

1) Proposing to use a deep Autoencoder as a preprocessing step to gain better representations of input data,
2) Exploiting novel deep learning classification models for credit card fraud detection,
3) Employing the Bayesian optimization algorithm to optimize the hyperparameters of the utilized deep learning models,
4) Conducting extensive experiments to evaluate the effectiveness of the proposed approach.

The rest of this paper is organized as follows: In Section 2, the background and related work of the state-of-the-art credit card fraud detection methods are provided. Section 3, initially describes Autoencoders and then presents the two-stage framework for fraud detection. In Section 4, we give the experimental results on a real-world fraud dataset. Section 5 concludes the paper.

2. Background and related work

2.1. Background

2.1.1. Fraud detection task

As mentioned in the previous section, due to the widespread use of e-commerce platforms and, consequently using electronic payment systems, employing an effective fraud detection system plays a crucial role in preventing losses. The huge volume of credit-card transactions produced in the context of electronic payment systems allows the creation of suitable datasets for building a data-driven fraud detection system (Bao et al., 2022). Datasets of credit card transactions contain a variety of features that can be incorporated into machine learning models, such as transaction characteristics, cardholders, or transaction histories. The fraud detection datasets have the following characteristics:

Unavailability of public dataset

Despite the abundance of credit card transaction data, the number of public datasets for conducting experiments by researchers is limited, resulting in data scarcity. Restrictions on information disclosure in this area prohibit operators from disclosing information about their business activities. Many financial institutions will not accept even an anonymous release of data (Saia & Carta, 2019).

Cost sensitivity

By definition, detecting credit card fraud is a cost-sensitive issue, since false positives result in higher costs than false negatives. In the event that a transaction is deemed fraudulent, but is not, the financial institution incurs an administrative cost. Conversely, if a fraud is not detected, the value of the transaction is lost (Correa Bahnsen et al., 2016).

Data imbalance

In fraud detection, there is typically a high number of normal (legitimate) cases and a small number of fraud cases used to train a model, resulting in an imbalanced data classification problem, resulting in a reduction in the effectiveness of classification approaches. Some machine learning approaches require to use both genuine and fraudulent instances to train their model; this challenge is sometimes solved by preprocessing the dataset to create an artificial balance of data (Saia & Carta, 2019). Handling the data imbalance problem can be carried out using either the over-sampling methods or the under-sampling methods. In over-sampling, the balance in data is created by duplicating fraudulent instances, while under-sampling consists of balancing data by omitting the normal instances. Some studies have indicated that employing such strategies enhances the performance of modeling techniques. Also, it has been demonstrated that oversampling strategies outperform under-sampling procedures.

Table 1
Summary of past research on credit card fraud detection using machine/deep learning.

Reference	Contribution	Modeling techniques	Dimension reduction (Yes or No): with what method
(Forough & Momtazi, 2021)	Proposing a novel ensemble method of deep sequential neural network to predict credit card frauds.	LSTM, GRU, Ensemble of RNNs	No
(Jurgovsky et al., 2018)	Proposing two approaches.1.RNNs for sequence classification.2.Using traditional feature engineering (Adding time delta as an additional feature + Feature aggregation).	LSTM and (Random Forest for comparison)	No
(Fiore et al., 2019)	Proposing a framework for generating synthetic fraud data using GAN and comparing the results with SMOTE in the same way.	ANN classifier	No
(Sohony, Pratap, & Nambiar, 2018)	Proposing an algorithm to partition dataset and train different models on each partition and finally ensemble all models.	ANN, Random Forest, Logistic Regression	No
(Forough & Momtazi, 2022)	Proposing an algorithm to under-sample data by a sliding window.	ANN, GRU, LSTM, LSTM-CRF	No
(Benchaji et al., 2021)	Proposing a method containing steps are: Feature selection then using dimension reduction algorithms and applying data to two proposed deep models.	LSTM, Attention Mechanism	Yes, manually selecting features then using PCA, *t*-SNE, UMAP for dimension reduction.
(Tingfei, Guangquan, & Kuihua, 2020)	Using GAN and VAE to overcome imbalance problem then comparing the results with SMOTE.	ANN	No
(Correa Bahnsen et al., 2016)	Proposed to capture customers spending pattern and also use time of the transactions for better classification.	Decision Tree, Logistic Regression, Random Forest	No
(Yuksel et al., 2020)	Proposed to use dimension reduction algorithms for preparing data for KNN classifier.	KNN	Yes, using PCA and NCA
(Saia & Carta, 2019)	Proposing new proactive approaches, one based on the Fourier transform, and one based on the Wavelet transform for credit card fraud detection. These approaches only non-fraudulent data. The exploited proactive approaches show similar performance to Random Forest method.	Two proactive methods based on DWT and DFT	No
(Laurens, Jusak, & Zou, 2017)	Proposing a new proactive fraud detection model using invariant diversity to uncover patterns among characteristics of the devices that are used in executing the transactions. The model fits a regression function on diversity index of various attributes, which is employed to detect fraudulent transactions.	Regression	No
(Singh & Jain, 2019)	Evaluating the effectiveness of the filter and wrapper feature selection methods in enhancing the performance of some classifiers	Decision tree, AdaBoost, Random Forest, Naive Bayes, and Projective Adaptive Resonance Theory (PART)	Feature selection based on Information gain, and a wrapper feature selection method.
(Vaughan, 2020)	proposing an approach to model selection with big data that incorporates a subsampling approach into a stochastic stagewise framework.	The stochastic stagewise approach	No

Dimensionality of feature space

Credit card transaction databases include a range of features, such as transaction-related attributes, cardholder information, or transaction histories. Therefore, to enhance the performance of the learning techniques, it is essential to conduct dimensionality reduction techniques. Some previous studies have been used methods such as PCA, and neighborhood components analysis (NCA) for dimensionality reduction (e.g., (Benchaji, Douzi, El Ouahidi, & Jaafari, 2021; Yuksel, Bahtiyar, & Yilmazer, 2020)). However, to obtain an expressive data representation from the input data, it is crucial to use deep representation techniques (Bengio, Courville, & Vincent, 2012).

2.2. *Related work*

A data-driven fraud detection system identifies transactions with a high probability of being fraudulent based on historical fraud patterns. Recent years have seen a growing interest in using machine learning and deep learning in fraud detection. (Bao et al., 2022) provided a complete review of the challenges of employing machine learning models in fraud detection and then investigated the academic literature that aimed to tackle those challenges. A comprehensive Table 1 provides some studies in the context of credit card fraud detection. It also describes their contributions, the utilized modeling techniques, and whether dimensionality reduction techniques were applied as data preprocessing or not. According to Table 1 previous studies have employed a variety of machine learning and deep learning methods, including ANN (Pradhan, Rao, Deepika, Harish, Kumar, & Kumar, 2021), logistic regression (T.

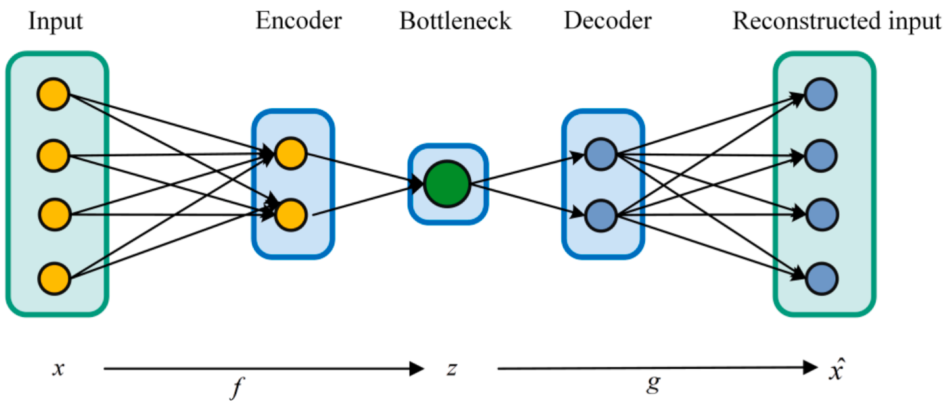

Fig. 1. The structure of a deep Autoencoder.

Wang & Zhao, 2022), *k*-nearest neighbors (*k*NN) (Yuksel et al., 2020), support vector machines (Bhattacharyya, Jha, Tharakunnel, & Westland, 2011), artificial Immune Systems (Brabazon, Cahill, Keenan, & Walsh, 2010; Soltani Halvaiee & Akbari, 2014), random forest (Dal Pozzolo, Caelen, Le Borgne, Waterschoot, & Bontempi, 2014). LSTM (Benchaji et al., 2021; Forough & Momtazi, 2021, 2022; Jurgovsky et al., 2018) is a widely-used deep learning method in this context. It has been stressed in the literature that a lower dimensional representation of the input data of a fraud dataset is crucial to constructing robust and efficient fraud detection systems. Some previous studies have used dimensionality reduction techniques, as shown in Table 1. Principle component analysis (PCA) is the most commonly used linear dimensionality reduction technique (e.g., (Benchaji et al., 2021; Yuksel et al., 2020)). Autoencoder as a representation learning technique has been demonstrated to be effective in a wide range of fields (Bengio et al., 2012; Y. Wang, Yao, Zhao, & Zheng, 2015; Xu, Jang-Jaccard, Singh, Wei, & Sabrina, 2021). In this, we try to investigate the dimensionality reduction ability of Autoencoders in credit card fraud detection. So, we develop a deep Autoencoder model, which learns representations from its input, which are then used as inputs to deep learning classifiers.

3. Proposed model

In this section, we firstly describe the structure of the Autoencoder model and then we describe our proposed approach for credit card fraud detection.

3.1. Autoencoder

An Autoencoder is an unsupervised neural network that contains two modules including encoder and decoder. Given an input $\boldsymbol{x}$ belongs to the m-dimensional space. The encoder part maps $\boldsymbol{x}$ into the new projection (latent representation) $z$in k-dimensional space. Then a decoder module, decodes the latent representation to an output with the same dimensions as input$\boldsymbol{x}$. The classic Autoencoder composed of three layers including input, hidden and output. Fig. 1 illustrates a deep Autoencoder network in which the encoder function denoted by $f(x) = z$and the decoder function defined by$g(z) = \hat{x}$. The training objective is to minimize the reconstruction error, i.e., the distance between input$x$and the decoded$\hat{x}$. The cost function of an Autoencoder is the reconstruction error, the difference between input (x_i) and its reconstruction $(\hat{x}_i)$ (as shown in Eq. (1)) and this cost function is called Mean Squared Error. There are other cost functions that are usable for autoencoders; We used three different

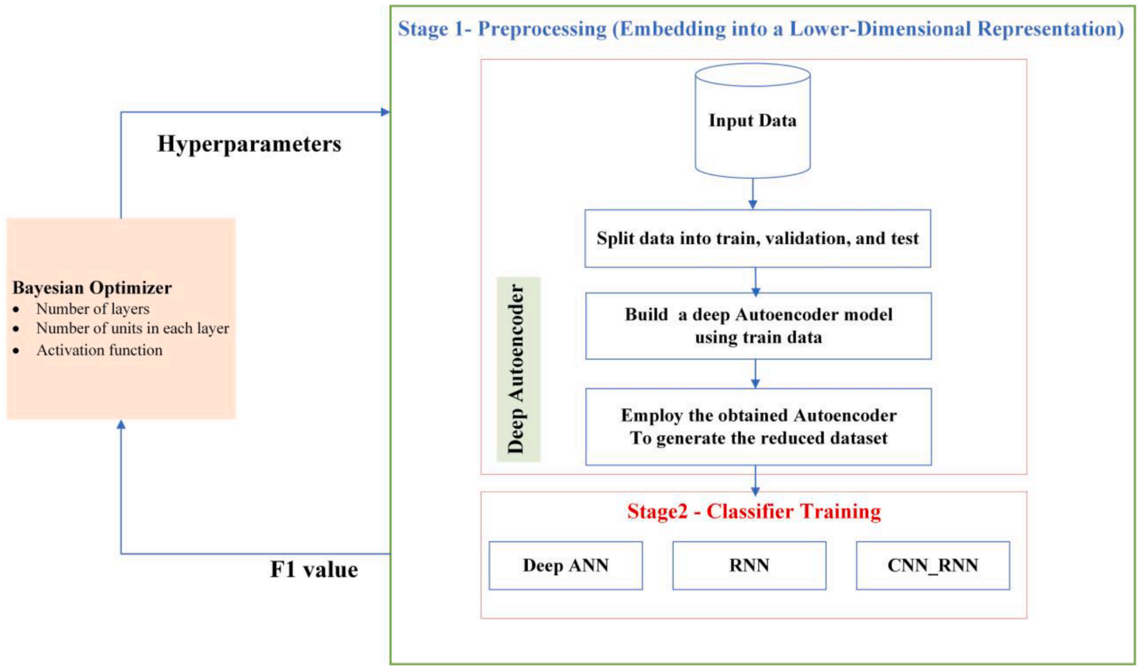
Stage 1- Preprocessing (Embedding into a Lower-Dimensional Representation)
Hyperparameters
Input Data
Split data into train, validation, and test
Build a deep Autoencoder model using train data
Employ the obtained Autoencoder To generate the reduced dataset
Deep Autoencoder
Bayesian Optimizer
Number of layers
Number of units in each layer
Activation function
Stage2 - Classifier Training
Deep ANN
RNN
CNN_RNN
F1 value

Fig. 2. The proposed methodology for credit card fraud detection.

Table 2
The confusion matrix.

		Predicted label	
		Positive	Negative
Actual label	**Positive**	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

candidate cost functions in hyperparameter tuning phase. Similar to ANN, Autoencoders are trained using the backpropagation algorithm. Since the dataset is normalized to range of 0 to 1 and the objective of autoencoder is to reconstruct input data, we should use Sigmoid activation function for output layer of autoencoder (decoder's output).

$$L(x,\widehat{x}) = \frac{1}{2}\sum_{i}(x_i - \widehat{x}_i)^2 \tag{1}$$

3.2. *The proposed framework*

As mentioned earlier, constructing a better and more expressive representation of the input data before performing classification is among the approaches that are implemented to increase the classification performance. In recent years, autoencoders have become popular among representation learning techniques for identifying underlying patterns in input data (Bengio et al., 2012). Our proposed model firstly utilizes an Autoencoder model to perform data reduction. Then, the transformed data is employed for fitting fraud detection models. Actually, the proposed model is a three-stage training algorithm including the following steps:

1) (Autoencoder training) build an Autoencoder model on the original dataset with *m* features.
2) (Producing new transformed dataset) apply the fitted Autoencoder model on the input data and generate the new dataset with *k* features ($k < m$) by feeding entire dataset to Autoencoder's input (Encoder's input) and extracting features from output of Bottleneck layer (encoder's output).
3) (Classifier training) classifier training using the new transformed dataset obtained from the previous step.

In the following, we give more detail on each phase. The overall procedure of the proposed method for credit card fraud detection is shown in Fig. 2. As illustrated in Fig. 2 the hyperparameters of autoencoder and classifier model are being tuned together with respect to validation F1 score that classifier was evaluated on.

4. Empirical study

In this study, we perform our empirical study using two datasets including 1) European Cardholders Dataset and 2) German Credit Dataset. In the following subsections, we give detailed descriptions of experiments and results for each dataset. In this study, we implement the proposed approach in Python with Keras library and TensorFlow (Chollet, 2015).

4.1. *Evaluation measures*

In order to evaluate the performance of the obtained models, Recall, Precision, and F1-Measure are used (Han, Kamber, & Pei, 2011). Considering the confusion matrix illustrated in Table 2, these measures are calculated using

Table 3
Description of the utilized dataset (European dataset).

	Total instances	Normal instances	Fraudulent instances	Imbalance Rate (%)
Entire dataset	284,807	284,315	492	0.1727
Train dataset	189,871	189,501	370	0.1948
Test dataset	94,936	94,814	122	0.1285

$$\begin{aligned} Recall &= \frac{TP}{TP+FN} \\ Precision &= \frac{TP}{TP+FP} \\ F1-Measure &= 2\times\frac{Precision\times Recall}{Precision+Recall} \end{aligned} \tag{2}$$

In Eq. (2), TP is the number of fraud instances predicted as fraud. FP is the number of normal transactions predicted as fraud. FN is the number of fraud instances that are incorrectly classified as normal. TN represents the number of normal transactions classified as normal.

Considering the aforementioned description, Recall denotes the ratio of the correctly recognized fraud instances. Precision represents the ratio of predicted frauds that are correctly recognized. F1-measure is the weighted harmonic average of Recall and Precision.

Furthermore, we need to consider two different measures that are useful for imbalance problems: AUC-ROC and AUC-PR. AUC stands for area under the curve and it is for calculating certain curves. ROC curve is a chart based on the tradeoff between true positive rate (TPR) and false positive rate (FPR) (Han et al., 2011). Similar to AUC-ROC in order to define AUC-PR we need to define Precision-Recall curve. For imbalanced datasets, the Precision-Recall curve provides an accurate picture of a classifier's performance. On the PR curve, each point indicates the classifier's Precision at a specific Recall level (Jurgovsky et al., 2018).

4.2. *Empirical study on European Cardholders dataset*

4.2.1. *Description of European Cardholders dataset*

In this section, we use the credit card transactions dataset of Dal Pozzolo et al. (2014). The dataset consists of 284,807 transactions made over two days in September 2013. This dataset contains 492 fraudulent transactions. There are only 0.172 percent positive class samples (fraud transactions) in the data set, which indicates a major imbalance in the data set. This data set contains numerical variables obtained through PCA dimension reduction. Due to data privacy concerns, the original features are not provided. Features V1, V2, …, V28 are the principal components obtained with the PCA method, the only features to which PCA has not been applied are "Time" and "Amount". The "Time" feature indicates how many seconds passed between each transaction and the first. Additionally, the "Amount" feature indicates the amount for each transaction. Dataset is divided into two sets; The first two third of dataset is training set and the remaining part is test set which we will mention this splitting method in our experiment results as Holdout

Table 4
The range of hyperparameters for Autoencoder model.

Autoencoder hyperparameter	Search Space
Number of layers in Encoder and Decoder	{2, 3, 4, 5}
Number of units in each layer	[1–512]
Bottleneck Layer's Units number	{10, 12, 14, 16, 18, 20, 22, 24}
Bottleneck Layer's Activation Function	{Sigmoid, Linear}
Loss Function	{Mean Squared Error, Mean Absolute Error, Binary Cross-Entropy}

Table 5
The range of hyperparameters of all utilized deep learning models.

Common hyperparameters for all classifiers	Optimizer	{Adam, Nadam, Adamax}
	Learning Rate	{0.001, 0.01, 0.1}
	Dropout Rate	Uniform (0, 0.5)
	L1 layer weight Regularizer coefficient	Loguniform(log(1e-6), log(1e-2))
DNN	Number of hidden Dense layers	[1–5]
	Number of units for each Dense layer	[1–512]
RNN	RNN Type	{LSTM, GRU}
	Number of hidden RNN layers	{1, 2}
	Number of hidden Dense layers	{0, 1, 2}
	Number of units for each RNN layer	[1–128]
	Number of units for each Dense layer	[1–64]
CNN_RNN	RNN Type	{LSTM, GRU}
	Number of hidden CNN layers	{1, 2, 3, 4, 5, 6}
	Number of hidden RNN layers	{0, 1, 2}
	Number of hidden Dense layers	{0, 1, 2}
	Number of units for each CNN layer	[1–256]
	Number of units for each RNN layer	[1–128]
	Number of units for each Dense layer	[1–64]
	Filter Size	{2, 3, 4, 5}
	Pool Size	{2, 3, 4}
	Batch Normalization (after CNN layer)	{Yes, No}

method (Han et al., 2011). Each dataset's characteristics are shown in Table 3.

4.2.2. Experimental setup and model building stage

For this case study, we employ-four classification models, including three deep learning models based on DNN (or MLP), RNN, and CNN as well as SVM to test the usefulness of the proposed approach. We are using SVM model with RBF Kernel as a shallow model for comparison purposes. As indicated in Fig. 2, we firstly train a deep auto-encoder model. Table 4 summarizes the hyperparameters for the Autoencoder model which is trained in preprocessing stage and the range of values considered for each of them. Also, the utilized deep learning-based classifiers contain some crucial hyperparameters that their proper tuning significantly impacts on the performance of models. The models are applied both to the original dataset as well as to the new dataset obtained with the Autoencoder model. Furthermore, these techniques are applied to the data resulting from the PCA method as a baseline algorithm for dimension reduction to gain a better insight about our AE's representation power. Table 5 gives the range of values for hyperparameters of the utilized deep learning classifiers. The hyperparameters of all utilized models are fine-tuned with the Bayesian optimization algorithm. Two common properties that all deep classifiers contain are: They use Binary Cross-Entropy as their loss function and their output layer is a Dense layer with 1 unit and Sigmoid activation. This is because the Credit Card Fraud Detection problem is formulated as a binary classification.

A training epoch is a complete run through the entire training data. If the epoch size of the model is too small, the model will be unable to

Table 6
The architecture of the obtained models using the original data. In all models, Output is a Dense layer with one unit and Sigmoid activation.

Model	Topology
DNN	• Input (29 dimensions) • Dense (units: 64 and activation: ReLU) • Dense (units: 32 and activation: ReLU) • Output o Optimizer: Adam o Learning Rate: 1e-2
RNN	• Input (29 dimensions) • LSTM (units: 128) • LSTM (units: 64) • Dense (units: 64 and activation: ReLU) • Output o Optimizer: Adamax o Learning Rate: 1e-3
CNN_RNN	• Input (29 dimensions) • 1D Convolutional (filters: 256 and filter size: 3 and activation: ReLU) • 1D Max Pooling (pool size: 3) • LSTM (units: 128) • Output o Optimizer: Adamax o Learning Rate: 1e-3

Table 7
The topology of the best AE model (European dataset).

Autoencoder	• Input (29 dimensions) • Dense (units: 256 and activation: ReLU) • Dense (units: 128 and activation: ReLU) • Dense (units: 32 and activation: ReLU) • **(Bottleneck)** Dense (units: 22 and activation: Sigmoid) • Dense (units: 32 and activation: ReLU) • Dense (units: 128 and activation: ReLU) • Dense (units: 256 and activation: ReLU) • Output (Dense layer with 29 units and Sigmoid activation)

Table 8
The architecture of the obtained models using the new data. In all models, output is a dense layer with one unit and Sigmoid activation.

Model	Topology
AE-DNN	• Input (data with 22 dimensions) • Dense (units: 16 and activation: ReLU) • Dense (units: 8 and activation: ReLU) • Dense (units: 2 and activation: ReLU) • Output o Optimizer: Adamax o Learning Rate: 1e-3
AE-RNN	• Input (22 dimensions) • LSTM (units: 64) • LSTM (units: 64) • Dense (units: 32 and activation: ReLU) • Dense (units: 16 and activation: ReLU) • Output o Optimizer: Adamax o Learning Rate: 1e-3
AE-CNN_RNN	• Input (22 dimensions) • 1D Convolutional (filters: 32 and filter size: 3 and activation: ReLU) • 1D Max Pooling (pool size: 3) • LSTM (units: 64) • Dense (units: 16 and activation: ReLU) • Dense (units: 16 and activation: ReLU) • Output o Optimizer: Adamax o Learning Rate: 1e-2

Table 9
The results of the utilized models on original dataset and extracted dataset using holdout method (European dataset).

Data	Model	Recall	Precision	F1-Measure	AUC-ROC	AUC-PR
29-Dimension	DNN	0.7131	0.9456	0.8131	0.8810	0.7254
	RNN	0.6803	0.9764	0.8019	0.8686	0.7202
	CNN_RNN	0.7213	0.9361	0.8148	0.9095	0.7355
	SVM	0.7103	0.8650	0.7800	0.8428	0.6333
22-Dimension (extracted from AE)	**AE-DNN**	0.7213	0.9670	0.8263	0.9093	0.7746
	AE-RNN	0.7212	0.9777	0.8302	0.8973	0.7630
	AE-CNN_RNN	0.7377	0.9677	0.8372	0.9217	0.7511
	AE-SVM	0.7295	0.9780	0.8357	0.8647	0.7153
	Fiore et al. (2019)	0.7328	0.9320	0.8205	–	–

capture patterns from the training data, whereas if it is too large, the model will be overfitted to the data (Abbasimehr & Paki, 2021). Early stopping technique (Prechelt, 2012) is used to determine the best epoch size for each experiment. We use different variants of Adam optimizer (Yu Xue, Tong, & Neri, 2022), to train deep neural networks.

Before we discuss our AE's representational power, we need to build our baseline models only using the original dataset. As mentioned earlier this is possible via Bayesian Optimization to find suitable architectures for these baseline models. The topologies of the best obtained deep learning models are depicted in Table 6.

4.2.3. *The obtained Autoencoder model*

As illustrated in Fig. 2, the Autoencoder model learns a new representation of data via fitting on the original dataset. After we fitted the AE model, we extract the new transformed dataset through the Bottleneck layer. The topology of the best AE model (Autoencoder) is presented in Table 7. As shown in Table 6 the number of units in the hidden layer is 22, which means that the dimensionality of the transformed dataset by AE is 22.

To evaluate the dimension reduction ability of the obtained Autoencoder model, we apply-four distinct methods to the new dataset generated by the AE model. As previously noted, we used a Bayesian Optimization technique to fine-tune the hyperparameters of the deep learning methods. The architectures of the fine-tuned models are summarized in Table 8.

4.2.4. *Results*

The results of the utilized models on the original and AE-transformed datasets are shown in Table 9 which are evaluated using the Holdout method. On original data, the best model in terms of F1 is the CNN_RNN model with F1 = 0.8148. Also, the performance of the DNN model is very close to the CNN in terms of F1. SVM shows poor performance in terms of F1 which indicates that this model performs poorly when dealing with the class imbalance problem. Also, in terms of AUC-ROC and AUC-PR, the CNN_RNN model achieves the best result.

Besides, as seen in Table 9, comparing the results of baseline models with the models resulted from the proposed approach indicates that AE-DNN outperforms the DNN model in terms of all performance measures. Also, regarding F1, Recall, and Precision, AE-RNN outperforms the baseline RNN. Specifically, in terms of Recall and F1, the improvement of AE-RNN over RNN is apparent. Besides, the results demonstrate the superiority of AE-CNN_RNN over CNN_RNN in terms of all performance measures. Also, AE-SVM achieves better results than the baseline SVM model with respect to all performance measures. Especially, in terms of Precision and F1, the improvement of the AE-SVM over SVM is significant indicating the effectiveness of utilizing the Autoencoder model to transform data representation. Overall, all of the proposed Autoencoder-based models outperform their baseline models across all performance metrics demonstrating the effectiveness of the proposed approach.

Table 11
PCA-DNN and AE-DNN and Original DNN using holdout method (European dataset).

# Dimensions (D)	F1	Recall	Precision	AUC-ROC	AUC-PR
10 (PCA)	0.7707	0.6475	0.9518	0.8237	0.6528
12 (PCA)	0.7962	0.6885	0.9438	0.8442	0.6482
14 (PCA)	0.7854	0.7049	0.8866	0.8524	0.66
16 (PCA)	0.785	0.6885	0.913	0.8442	0.6577
18 (PCA)	0.8018	0.7295	0.89	0.8647	0.6712
20 (PCA)	0.7814	0.6885	0.9032	0.8442	0.6482
22 (PCA)	0.8018	0.7295	0.89	0.8647	0.6482
24 (PCA)	0.7964	0.7213	0.8889	0.8606	0.632
26 (PCA)	0.7963	0.7049	0.9149	0.8524	0.6528
28 (PCA)	0.7926	0.7049	0.9053	0.8524	0.6527
29 (original dataset)	0.8131	0.7213	0.9670	0.8810	0.7254
22 (extracted from AE)	0.8263	0.7131	0.9456	0.9093	0.7746

As further analysis of the results of the proposed method, we consider (Fiore, De Santis, Perla, Zanetti, & Palmieri, 2019) as another baseline method. As illustrated in Table 9, AE-CNN_RNN outperforms the model presented in (Fiore et al., 2019) in terms of all performance metrics. Besides, in terms of Precision and F1, all variants of the proposed methods including AE-DNN, AE-RNN, and AE-CNN_NN outperform the model presented in (Fiore et al., 2019). This indicates the strong capability of the AE model in learning the new representation of data.

For reassurance purposes we repeated the experiments with 5-Fold cross validation method mainly for the severe imbalance problem of this dataset. Cross validation results are shown in Table 10 and as the results on original dataset indicate that the SVM model is outperformed

Table 10
The results of the utilized models on original dataset and extracted dataset using 5-fold cross validation method (European dataset).

Data	Model	Recall	Precision	F1-Measure	AUC-ROC	AUC-PR
29-Dimension	DNN	0.7265	0.8259	0.7720	0.8928	0.7096
	RNN	0.7648	0.8403	0.7959	0.9198	0.7469
	CNN_RNN	0.7274	0.8521	0.7834	0.9044	0.7308
	SVM	0.4288	0.8755	0.5757	0.7143	0.3790
22-Dimension (extracted from AE)	**AE-DNN**	0.7376	0.9211	0.8131	0.9277	0.7705
	AE-RNN	0.7617	0.8685	0.8044	0.9254	0.7532
	AE-CNN_RNN	0.7525	0.8979	0.8128	0.9278	0.7553
	AE-SVM	0.7134	0.8540	0.7774	0.8566	0.6117

Table 12
PCA-RNN and AE-RNN and Original RNN using holdout method (European dataset).

# Dimensions (D)	F1	Recall	Precision	AUC-ROC	AUC-PR
10 (PCA)	0.7963	0.7049	0.9149	0.8524	0.67
12 (PCA)	0.8057	0.6967	0.9551	0.8483	0.6688
14 (PCA)	0.7476	0.6311	0.9167	0.8155	0.6133
16 (PCA)	0.7593	0.6721	0.8723	0.836	0.5815
18 (PCA)	0.8108	0.7377	0.9	0.8688	0.6269
20 (PCA)	0.8	0.7049	0.9247	0.8524	0.6024
22 (PCA)	0.8053	0.7459	0.875	0.8729	0.6173
24 (PCA)	0.7679	0.7049	0.8431	0.8524	0.6738
26 (PCA)	0.7563	0.7377	0.7759	0.8687	0.5942
28 (PCA)	0.8018	0.7295	0.89	0.8647	0.5491
29 (original dataset)	0.8019	0.6803	0.9764	0.8686	0.7202
22 (extracted from AE)	0.8302	0.7212	0.9777	0.8973	0.7630

Table 13
PCA- CNN_RNN and AE- CNN_RNN and Original CNN_RNN using holdout method (European dataset).

# Dimensions (D)	F1	Recall	Precision	AUC-ROC	AUC-PR
10 (PCA)	0.7476	0.6311	0.9167	0.8155	0.6447
12 (PCA)	0.7944	0.6967	0.9239	0.8483	0.6447
14 (PCA)	0.7826	0.7377	0.8333	0.8688	0.5997
16 (PCA)	0.8125	0.7459	0.8922	0.8729	0.6592
18 (PCA)	0.767	0.6475	0.9405	0.8237	0.5424
20 (PCA)	0.7479	0.7295	0.7672	0.8646	0.6295
22 (PCA)	0.7768	0.7131	0.8529	0.8565	0.6447
24 (PCA)	0.7565	0.7131	0.8056	0.8564	0.5384
26 (PCA)	0.7407	0.7377	0.7438	0.8687	0.6007
28 (PCA)	0.8019	0.6967	0.9444	0.8483	0.635
29 (original dataset)	0.8148	0.7213	0.9361	0.9095	0.7355
22 (extracted from AE)	0.8372	0.7377	0.9677	0.9217	0.7511

Table 14
PCA- SVM and AE- SVM and Original SVM using holdout method (European dataset).

# Dimensions (D)	F1	Recall	Precision	AUC-ROC	AUC-PR
10 (PCA)	0.7475	0.6066	0.9737	0.8033	0.6011
12 (PCA)	0.75	0.6148	0.9615	0.8074	0.6011
14 (PCA)	0.7254	0.5738	0.9859	0.7869	0.5682
16 (PCA)	0.712	0.5574	0.9855	0.7787	0.5518
18 (PCA)	0.7053	0.5492	0.9853	0.7746	0.5437
20 (PCA)	0.6845	0.5246	0.9846	0.7623	0.5191
22 (PCA)	0.7016	0.5492	0.971	0.7746	0.5358
24 (PCA)	0.6809	0.5246	0.9697	0.7623	0.5113
26 (PCA)	0.6738	0.5164	0.9692	0.7582	0.5031
28 (PCA)	0.6772	0.5246	0.9552	0.7623	0.5037
29 (original dataset)	0.7800	0.7103	0.8650	0.8428	0.6333
22 (extracted from AE)	0.8357	0.7295	0.9780	0.8647	0.7153

by other deep models. Besides, each model yielded from the proposed approach outperforms its baseline in terms of all performance measures.

4.2.5. Comparison with PCA

PCA is a prominent technique to deal with dimension reduction problems. PCA aims to project *n*-dimensional data into a subspace with d dimensions ($d < n$). To gain more insights into our proposed approach, in this section, we apply PCA on the dataset and train DNN, RNN, CNN_RNN, and SVM on the transformed dataset. We iterate the experiments with d = [10, 12, 14, 16, 18, 20, 22, 24, 26, 28] dimensions. The results of the four applied models are in Tables 11-14. As Table 11 indicates the best model in terms of F1 is achieved using PCA-DNN with 18 dimensions. In fact, using PCA as the dimension reduction technique resulted in a model with lower F1 when compared to the baseline DNN as shown in Table 9 (DNN = 0.8131, PCA-DNN = 0.8018). For RNN the best F1 value is also obtained using 18 dimensions. This result is better than the baseline RNN model. CNN_RNN model achieves the best results when applied on the data with 16 dimensions. There is no improvement in using PCA with CNN_RNN in comparison to the baseline CNN_RNN. Finally, the best SVM model with respect to F1 is obtained by applying it on the dataset with 12 dimensions. Using PCA with SVM reduced the performance in terms of F1 as seen in Table 9 and Table 14.

Comparing the results of the proposed method with the PCA indicates that AE-DNN outperforms PCA-DNN in terms of F1 and Precision measures. AE-RNN performs better than PCA-RNN with respect to F1 and Precision measures. AE-CNN_RNN outperforms PCA- CNN_RNN in terms of F1 and Precision. Also, AE-SVM performs significantly better than PCA-SVM which show the ability of the AE model in learning new representations.

Table 15
Description of the German Credit Data.

	Total instances	Good instances	Bad instances	Imbalance Rate (%)
Entire dataset	1000	700	300	30
Train dataset	670	469	201	30
Test dataset	330	231	99	30

Table 16
The range of hyperparameters for Autoencoder model (German dataset).

Autoencoder hyperparameter	Search Space
Number of layers in Encoder and Decoder	{2, 3, 4, 5}
Number of Units in each Layer	[1–512]
Bottleneck Layer's Units number	{14, 15, 16, 17, 18, 19, 20, 21}
Bottleneck Layer's Activation Function	{Sigmoid, Linear}
Other Hidden Layers' Activation Function	{Sigmoid, ReLU, SELU, ELU, Tanh}
Loss Function	{Mean Squared Error, Mean Absolute Error, Binary Cross-Entropy}

4.3. Empirical study on German credit data

4.3.1. Description of German credit dataset

To further demonstrate the effectiveness of the proposed approach, we used another imbalanced dataset. In this dataset (German Credit Data[2]), each instance represents a customer with regards to their credit status, which can be good or bad, representing someone who takes out a loan from a bank. The original form of this dataset contains categorical/ symbolic attributes and in total it has 20 attributes. On the other hand, Strathclyde University produced a numerical format of the original dataset which contains 24 numerical attributes. The second format of this dataset is suitable for algorithms that cannot cope with categorical variables and this format is our chosen format for further experiments. Instances of this dataset are grouped into two classes, Good and Bad. The are 1000 instances in the dataset but only 30 percent of them are labeled as positive (Bad). Dataset is divided into two sets; The first two third of the dataset is the training set and the remaining part is the test set. Each dataset's characteristics are shown in Table 15.

4.3.2. Experimental setup and model building stage -German credit data)

The steps of our framework are repeated for this dataset as well. Table 16 summarizes the hyperparameters for the Autoencoder model which is trained in preprocessing stage on the German dataset. Table 17 gives the range of values for hyperparameters of the utilized deep

[2] https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data).

Table 17
The range of hyperparameters of utilized deep learning model (German dataset).

Common hyperparameters for all classifiers	Optimizer	{Adam, Nadam, Adamax}
	Learning Rate	{0.0003, 0.001, 0.003, 0.03}
	Dropout Rate	Uniform (0, 0.5)
	L1 layer weight Regularizer coefficient	Loguniform(log(1e-6), log(1e-2))
DNN	Number of hidden Dense layers	[1–5]
	Number of units for each Dense layer	[1–512]

Table 18
The architecture of the obtained model using the original data. In the model, Output is a Dense layer with one unit and Sigmoid activation (German dataset).

Model	Topology
DNN	• Input (24 dimensions) • Dense (units: 1024 and activation: ReLU) • Output o Optimizer: Nadam o Learning Rate: 1e-2

Table 19
The topology of the best AE model (German dataset).

Autoencoder	• Input (24 dimensions) • Dense (units: 512 and activation: ELU) • Dense (units: 256 and activation: ELU) • **(Bottleneck)** Dense (units: 18 and activation: Linear) • Dense (units: 256 and activation: ELU) • Dense (units: 512 and activation: ELU) • Output (Dense layer with 24 units and Linear activation) o Loss Function: Mean Squared Error o Optimizer: Adam o Learning Rate: 3e-3

Table 20
The architecture of the obtained models using the new data. In all models, output is a dense layer with one unit and Sigmoid activation (German dataset).

Model	Topology
AE-DNN	• Input (data with 22 dimensions) • Dense (units: 256 and activation: ReLU) • Dense (units: 128 and activation: ReLU) • Output o Optimizer: Adam o Learning Rate: 3e-4

learning classifier.

As we proceeded with our first case study, we build a baseline model using the original dataset that was fin-tuned by the Bayesian Optimization technique. This baseline model's architecture is shown in Table 18.

4.3.3. The obtained Autoencoder model

The topology of the second AE model is presented in Table 19. As shown in Table 19 the number of units in the Bottleneck hidden layer is 18, which means that the dimensionality of the transformed dataset by AE is 18.Table 20.

4.3.4. Results

In this case study, we employ only one deep learning model which is the DNN. The reason that none of the other previous deep learning models are used for this case study is that the numerical German dataset is better fitted with simple models due to the small size of the dataset. The model is applied both to the original dataset as well as to the new dataset obtained with the Autoencoder model. As we did in the previous case study, in the last step we compare our AE with a baseline algorithm such as PCA. As the aim is to construct a binary classification model, our DNN's last layer is a one-unit layer with the Sigmoid activation function.

The results of baseline and proposed models using both Holdout and Cross Validation methods are shown in Table 21 and Table 22 respectively. As seen in Table 21, AE-DNN outperforms the DNN model in terms of all performance measures. Also, by analyzing Table 22 we can get to the same conclusion that employing the AE model improves the performance of DNN.

4.3.5. Comparison with PCA

Here, we compare the PCA algorithm with our AE model. To do this, as mentioned in the first case study, we conduct the experiments with d = [14, 15, 16, 17, 18, 19, 20, 21] dimensions by training the DNN model with the PCA-transformed data. The results are shown in Table 23. Our AE model is superior to the PCA algorithm when we compare PCA-DNN with AE-DNN as revealed in the first case study.

5. Conclusion and future research

Our study proposes a two-stage framework comprised of a deep AE model as the dimensionality reduction technique, and three deep learning-based classifiers including DNN, RNN, and CNN_RNN to improve fraud detection accuracy. The Bayesian optimization algorithm is utilized to select the best hyperparameters of the employed models. The results of experiments indicated that the proposed approach improves the performance of the employed deep learning-based classifiers. Specifically, AE-DNN outperforms DNN in terms of all performance measures. Furthermore, AE-RNN and AE-CNN_RNN methods perform superior to their baseline counterparts. Also, we investigated the dimensionality reduction ability of the deep Autoencoder model with the ones of the PCA method. According to the experimental results on two fraud detection datasets, models created using deep AE outperform those created using PCA-obtained dataset in terms of F1 measure. This indicates that the utilized deep AE can learn better representation than PCA. Additionally, AE-DNN, AE-RNN, and AE-CNN_NN methods all performed better on European Cardholders Dataset than the best existing models. On German Credit Dataset, the AE-DNN model outperforms existing models. The results demonstrate the usefulness of using deep representation learning techniques in improving the performance of the credit card fraud detection approach. In future research, we will utilize supervised autoencoders as a data preprocessing method to enhance the accuracy of fraud detection system.

Table 21
The results of the utilized models on original dataset and extracted dataset using holdout method (German dataset).

	Model	Recall	Precision	F1-Measure (Bad)	F1-Measure (Good)	F1-Measure (macro avg)	AUC-ROC	AUC-PR
24-Dimension	DNN	0.7576	0.5435	0.6329	0.7943	0.7136	0.7424	0.5056
18-Dimension	**AE-DNN**	0.7273	0.5714	0.6400	0.8138	0.7269	0.7467	0.5228

Table 22
The results of the utilized models on original dataset and extracted dataset using 5-fold cross validation (German dataset).

	Model	Recall	Precision	F1-Measure (Bad)	F1-Measure (Good)	F1-Measure (macro avg)	AUC-ROC	AUC-PR
24-Dimension	DNN	0.58	0.6420	0.6094	0.8439	0.7267	0.7207	0.5406
18-Dimension	**AE-DNN**	0.57	0.6867	0.6229	0.8573	0.7401	0.7292	0.5673

Table 23
PCA-DNN and AE-DNN and Original DNN using holdout method (German dataset).

# Dimensions (D)	F1	Recall	Precision	AUC-ROC	AUC-PR
14 (PCA)	0.5389	0.5253	0.5532	0.6717	0.4692
15 (PCA)	0.5319	0.5051	0.5618	0.6681	0.4702
16 (PCA)	0.4375	0.3535	0.5738	0.6205	0.4398
17 (PCA)	0.5729	0.5758	0.57	0.6948	0.491
18 (PCA)	0.5263	0.5051	0.5495	0.6638	0.4626
19 (PCA)	0.5943	0.6364	0.5575	0.71	0.4946
20 (PCA)	0.5926	0.6465	0.547	0.7085	0.4888
21 (PCA)	0.5628	0.5657	0.56	0.6876	0.4821
24 (original dataset)	0.6329	0.7576	0.5435	0.7424	0.5056
18 (extracted from AE)	0.6400	0.7273	0.5714	0.7467	0.5228

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Abbasimehr, H., & Paki, R. (2021). Improving time series forecasting using LSTM and attention models. *Journal of Ambient Intelligence and Humanized Computing*. https://doi.org/10.1007/s12652-020-02761-x

Abbasimehr, H., Shabani, M., & Yousefi, M. (2020). An optimized model using LSTM network for demand forecasting. *Computers & Industrial Engineering, 106435*. https://doi.org/10.1016/j.cie.2020.106435

Bao, Y., Hilary, G., & Ke, B. (2022). Artificial Intelligence and Fraud Detection. In V. Babich, J. R. Birge, & G. Hilary (Eds.), *Innovative Technology at the Interface of Finance and Operations* (Volume I, pp. 223–247). Cham: Springer International Publishing.

Benchaji, I., Douzi, S., El Ouahidi, B., & Jaafari, J. (2021). Enhanced credit card fraud detection based on attention mechanism and LSTM deep model. *Journal of Big Data, 8*(1), 151. https://doi.org/10.1186/s40537-021-00541-8

Bengio, Y., Courville, A. C., & Vincent, P. (2012). Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR, abs/1206.5538, 1*(2665), 2012.

Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J. C. (2011). Data mining for credit card fraud: A comparative study. *Decision Support Systems, 50*(3), 602–613. https://doi.org/10.1016/j.dss.2010.08.008

Brabazon, A., Cahill, J., Keenan, P., & Walsh, D. (2010, 18-23 July 2010). *Identifying online credit card fraud using Artificial Immune Systems.* Paper presented at the IEEE Congress on Evolutionary Computation.

Chollet, F. (2015). Keras. Retrieved January 12, 2020, from https://github.com/fchollet/keras.

Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K., & Bengio, Y. (2015). Attention-based models for speech recognition. *Advances in neural information processing systems, 28*.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Correa Bahnsen, A., Aouada, D., Stojanovic, A., & Ottersten, B. (2016). Feature engineering strategies for credit card fraud detection. *Expert Systems with Applications, 51*, 134–142. https://doi.org/10.1016/j.eswa.2015.12.030

Dal Pozzolo, A., Caelen, O., Le Borgne, Y.-A., Waterschoot, S., & Bontempi, G. (2014). Learned lessons in credit card fraud detection from a practitioner perspective. *Expert Systems with Applications, 41*(10), 4915–4928. https://doi.org/10.1016/j.eswa.2014.02.026

Dev, K., Khowaja, S. A., Bist, A. S., Saini, V., & Bhatia, S. (2021). Triage of potential covid-19 patients from chest x-ray images using hierarchical convolutional networks. *Neural Computing and Applications*, 1–16.

Fiore, U., De Santis, A., Perla, F., Zanetti, P., & Palmieri, F. (2019). Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. *Information Sciences, 479*, 448–455. https://doi.org/10.1016/j.ins.2017.12.030

Forough, J., & Momtazi, S. (2021). Ensemble of deep sequential models for credit card fraud detection. *Applied Soft Computing, 99*, Article 106883. https://doi.org/10.1016/j.asoc.2020.106883

Forough, J., & Momtazi, S. (2022). Sequential credit card fraud detection: A joint deep neural network and probabilistic graphical model approach. *Expert Systems, 39*(1), e12795.

Fournier, Q., & Aloise, D. (2019, 3-5 June 2019). *Empirical Comparison between Autoencoders and Traditional Dimensionality Reduction Methods.* Paper presented at the 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE).

Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques: Concepts and Techniques*. Waltham, USA: Elsevier Science.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Jurgovsky, J., Granitzer, M., Ziegler, K., Calabretto, S., Portier, P.-E., He-Guelton, L., & Caelen, O. (2018). Sequence classification for credit-card fraud detection. *Expert Systems with Applications, 100*, 234–245. https://doi.org/10.1016/j.eswa.2018.01.037

Laurens, R., Jusak, J., & Zou, C. C. (2017, 4-8 Dec. 2017). *Invariant Diversity as a Proactive Fraud Detection Mechanism for Online Merchants.* Paper presented at the GLOBECOM 2017 - 2017 IEEE Global Communications Conference.

Luong, T., Pham, H., & Manning, C. D. (2015, September). *Effective Approaches to Attention-based Neural Machine Translation*, Lisbon, Portugal.

Ngai, E. W. T., Hu, Y., Wong, Y. H., Chen, Y., & Sun, X. (2011). The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems, 50*(3), 559–569. https://doi.org/10.1016/j.dss.2010.08.006

Pradhan, S. K., Rao, N. V. K., Deepika, N. M., Harish, P., Kumar, M. P., & Kumar, P. S. (2021, 2-4 Dec. 2021). *Credit Card Fraud Detection Using Artificial Neural Networks and Random Forest Algorithms.* Paper presented at the 2021 5th International Conference on Electronics, Communication and Aerospace Technology (ICECA).

Prechelt, L. (2012). Early Stopping — But When? In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Neural Networks: Tricks of the Trade* (Second Edition, pp. 53–67). Berlin, Heidelberg: Springer, Berlin Heidelberg.

Saia, R., & Carta, S. (2019). Evaluating the benefits of using proactive transformed-domain-based techniques in fraud detection tasks. *Future Generation Computer Systems, 93*, 18–32.

Singh, A., & Jain, A. (2019). *Adaptive credit card fraud detection techniques based on feature selection method Advances in computer communication and computational sciences* (pp. 167–178). Springer.

Sohony, I., Pratap, R., & Nambiar, U. (2018). *Ensemble learning for credit card fraud detection.* Paper presented at the Proceedings of the ACM India Joint International Conference on Data Science and Management of Data, Goa, India. 10.1145/3152494.3156815.

Soltani Halvaiee, N., & Akbari, M. K. (2014). A novel model for credit card fraud detection using Artificial Immune Systems. *Applied Soft Computing, 24*, 40–49. https://doi.org/10.1016/j.asoc.2014.06.042

Sultana, J., Usha Rani, M., & Farquad, M. A. H. (2020, 2020//). *An Extensive Survey on Some Deep-Learning Applications.* Paper presented at the Emerging Research in Data Engineering Systems and Computer Communications, Singapore.

Tingfei, H., Guangquan, C., & Kuihua, H. (2020). Using Variational Auto Encoding in Credit Card Fraud Detection. *IEEE Access, 8*, 149841–149853. https://doi.org/10.1109/ACCESS.2020.3015600

Tran, D. T., Iosifidis, A., Kanniainen, J., & Gabbouj, M. (2018). Temporal attention-augmented bilinear network for financial time-series data analysis. *IEEE Transactions on Neural Networks and Learning Systems, 30*(5), 1407–1418.

Turner, R., Eriksson, D., McCourt, M., Kiili, J., Laaksonen, E., Xu, Z., & Guyon, I. (2021). *Bayesian Optimization is Superior to Random Search for Machine Learning Hyperparameter Tuning: Analysis of the Black-Box Optimization Challenge 2020.* Paper presented at the Proceedings of the NeurIPS 2020 Competition and Demonstration Track, Proceedings of Machine Learning Research. https://proceedings.mlr.press/v133/turner21a.html.

Vaughan, G. (2020). Efficient big data model selection with applications to fraud detection. *International Journal of forecasting, 36*(3), 1116–1127. https://doi.org/10.1016/j.ijforecast.2018.03.002

Wang, T., & Zhao, Y. (2022, 20-22 Jan. 2022). *Credit Card Fraud Detection using Logistic Regression.* Paper presented at the 2022 International Conference on Big Data, Information and Computer Network (BDICN).

Wang, Y., Yao, H., Zhao, S., & Zheng, Y. (2015). *Dimensionality reduction strategy based on auto-encoder.* Paper presented at the Proceedings of the 7th International Conference on Internet Multimedia Computing and Service, Zhangjiajie, Hunan, China. 10.1145/2808492.2808555.

Xu, W., Jang-Jaccard, J., Singh, A., Wei, Y., & Sabrina, F. (2021). Improving Performance of Autoencoder-Based Network Anomaly Detection on NSL-KDD Dataset. *IEEE Access, 9*, 140136–140146. https://doi.org/10.1109/ACCESS.2021.3116612

Xue, Y., & Qin, J. (2022). Partial Connection Based on Channel Attention for Differentiable Neural Architecture Search. *IEEE Transactions on Industrial Informatics, 1–10*. https://doi.org/10.1109/TII.2022.3184700

Xue, Y., Tong, Y., & Neri, F. (2022). An ensemble of differential evolution and Adam for training feed-forward neural networks. *Information Sciences, 608*, 453–471. https://doi.org/10.1016/j.ins.2022.06.036

Xue, Y., Wang, Y., Liang, J., & Slowik, A. (2021). A Self-Adaptive Mutation Neural Architecture Search Algorithm Based on Blocks. *IEEE Computational Intelligence Magazine, 16*(3), 67–78. https://doi.org/10.1109/MCI.2021.3084435

Yuksel, B. B., Bahtiyar, S., & Yilmazer, A. (2020). *Credit Card Fraud Detection with NCA Dimensionality Reduction*. Paper presented at the 13th International Conference on Security of Information and Networks, Merkez, Turkey. 10.1145/3433174.3433178.