# Towards an interpretable autoencoder: A decision-tree-based autoencoder and its application in anomaly detection

Diana Laura Aguilar, Miguel Angel Medina-Pérez, Octavio Loyola-González, Kim-Kwang Raymond Choo, *Senior Member, IEEE*, Edoardo Bucheli-Susarrey

**Abstract**—The importance of understanding and explaining the associated classification results in the utilization of artificial intelligence (AI) in many different practical applications (e.g., cyber security and forensics) has contributed to the trend of moving away from black-box / opaque AI towards explainable AI (XAI). In this paper, we propose the first interpretable autoencoder based on decision trees, which is designed to handle categorical data without the need to transform the data representation. Furthermore, our proposed interpretable autoencoder provides a natural explanation for experts in the application area. The experimental findings show that our proposed interpretable autoencoder is among the top-ranked anomaly detection algorithms, along with one-class Support Vector Machine (SVM) and Gaussian Mixture. More specifically, our proposal is on average 2% below the best Area Under the Curve (AUC) result and 3% over the other Average Precision scores, in comparison to One-class SVM, Isolation Forest, Local Outlier Factor, Elliptic Envelope, Gaussian Mixture Model, and eForest.

**Index Terms**—Interpretable artificial intelligence, Autoencoder, Decision tree, Anomaly detection, Explainable artificial intelligence (XAI).

## 1 Introduction

Deep Neural Networks (DNNs) have been utilized in detection and classification tasks for a large variety of applications, such as facial recognition [1], [2], palmprint recognition [3], visual classification [4], [5], traffic safety [6], object detection [7], [8], video captioning [9], speech recognition [10], and fault diagnosis [11]. However, a limitation of DNNs is their inability to provide insights into their complex behavior or reasoning behind the underlying resolutions [12]. Furthermore, due to the various transformations to the input data, these architectures are difficult to understand even for machine learning experts [13].

The need for explainability is more pronounced in sensitive applications such as health care monitoring [14], where the decisions have real-world physical consequences. Thus, there has been recent interest in achieving model transparency and accountability in the artificial intelligence (AI; broadly defined to also include machine and deep learning techniques) literature [12]. Furthermore, Gilpin et al. [15] suggested that these explanations ensure the correct behavior of the algorithm, and they concluded that machine learning systems would be more widely accepted once they are capable of providing satisfactory explanations for their decisions.

Rudin [12] pointed out that there is an increasing number of works on post-hoc models created to explain an original black-box. Recent reviews on the topic [15], [16] also echoed Rudin's observation [12]. Rudin also raised a number of concerns in the post-hoc-model approach. First, uncertainty about explanations leads to uncertainty about the original model. Second, post hoc explanations are not faithful to what the original model computes. Third, post hoc explanations may leave out so much information that they may end up making little or no sense. Reasoning from these facts, inherent interpretability seems to be a viable option.

Doshi-Velez and Kim [17] defined interpretability as the ability to provide meaning in such a way that humans can understand it. Rudin [12] stated that an inherently interpretable model is capable of producing its faithful explanations. In other words, interpretable models do not need a second model to explain themselves as they are designed to output decisions and explanations. Since these explanations are an indication of the algorithm accuracy and impartiality [15], such algorithms can potentially be utilized in high-stake / sensitive applications, such as anomaly detection [18], [19]. Chandola et al. [20] defined anomalies as patterns in data that behave unexpectedly, and anomalies

*D. L. Aguilar is with the Tecnologico de Monterrey, Carretera al Lago de Guadalupe Km. 3.5, Atizapán, Estado de México 52926, México (e-mail: a01751168@tec.mx).*
*M. A. Medina-Pérez is with Altair Management Consultants, Calle de José Ortega y Gasset 22–24, 5th Floor, 28006 Madrid, Spain; and Tecnologico de Monterrey, Carretera al Lago de Guadalupe Km. 3.5, Atizapán, Estado de México 52926, México (e-mail: mmp@altair.consulting, migue@tec.mx). Corresponding author.*
*O. Loyola-González is with Altair Management Consultants, Calle de José Ortega y Gasset 22–24, 5th Floor, 28006 Madrid, Spain (e-mail: olg@altair.consulting).*
*K.-K. R. Choo is with the Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249, USA (e-mail: raymond.choo@fulbrightmail.org)*
*E. Bucheli-Susarrey is with the Tecnologico de Monterrey, Carretera al Lago de Guadalupe Km. 3.5, Atizapán, Estado de México 52926, México (e-mail: A01016080@itesm.mx).*

in data suggest actionable information in applications such as health care monitoring.

In recent years, there has been a growing interest in using autoencoders [21] for anomaly detection. In such an approach, an encoder transforms an input into a hidden representation, and a decoder maps the hidden representation of the input back into the original space [22]. Autoencoders are generally built as neural networks, where these deep architectures consist of encoding and decoding blocks of non-linear layers [23]. Although these neural networks are efficient for anomaly detection, their non-linearity and lack of interpretability make it debatable whether they should be used for high-stake / sensitive applications [21]. Autoencoding can also be performed using decision trees [24], [25], which can potentially leverage the inherent interpretability of traditional decision trees [26].

In this paper, we propose a novel approach for anomaly detection. Specifically, our proposed model is a decision tree-based autoencoder (hereafter referred to as DTAE) that can detect anomalies on categorical data. Furthermore, to the best of our knowledge, it is the first interpretable autoencoder for anomaly detection, in the sense that it outputs not only decisions but also faithful explanations for them. A comparative summary with six state-of-the-art classifiers, (i.e., One-class SVM [27], Isolation Forest [28], Local Outlier Factor [29], Elliptic Envelope [30], Gaussian Mixture Model, and eForest [25]) shows the utility of our proposed DTAE.

The remainder of this paper is organized as follows. Section 2 briefly reviews the extant literature. Section 3 introduces our proposed approach, before we present the evaluation findings in Section 4. Finally, Section 5 concludes this paper.

## 2 Related Literature

We will now review the related literature on explainability (see Section 2.1) and autoencoders (see Section 2.2).

### 2.1 Explainable DNNs

Explainable DNNs can be broadly categorized based on their explanatory capability, for example in terms of processing (Section 2.1.1), representation (Section 2.1.2), generation (Section 2.1.3) [15]. We will also summarize the challenges associated with these approaches, their application domains, the datasets that have been used, the evaluation metrics, and the availability of the source code in Table 1.

#### 2.1.1 Explaining their processing

Gilpin et al. [15] reported that one fundamental approach to explaining DNNs is by their processing. Specifically, systems try to explain a specific output given a specific input, by summarizing its decision. The fundamental idea is to create a post hoc model, which the authors referred to as the *proxy model*. This allows the explanation of the original black-box (or referred to as opaque) model in a way that is easier to understand. Moreover, the authors proposed using linear models [31], decision trees [32], automatic-rule extraction [33], and salience mapping [34] to explain DNN processing. In other words, the input-output relations are determined by emulating the processing of the information in the network. The explanations created by these techniques will reveal information about the network processing. However, a limitation is the reliance on a post hoc model to explain the original network; post hoc techniques do not explain the actual reasoning process behind the network outputs [12]. Moreover, according to Li et al. [35], post hoc approaches often create explanations that are difficult for humans to understand, such as the examples presented in [32].

#### 2.1.2 Explaining their representation

According to Gilpin et al. [15], another approach is to explain their representation; that is to say, this approach aims to explain the data flow through the network. In other words, data to be studied is divided by layers [36], single units [37], or representation vectors. While such approaches that focus on the internal workings of DNNs allow us to interpret activation data through the network, it is difficult to evaluate the performance of these methods directly [15]. Additionally, their results are difficult to understand as they focus on the data inside the network and not to explain particular network decisions nor reasoning.

#### 2.1.3 Generating explanations

The third approach is to create more transparent DNNs, in the sense that they are built to explain themselves [15]. Example approaches include attention-based networks [38], networks trained to learn disentangled representations [39], and generated explanations [40]. Vaswani et al. [38], for example, proposed a neural architecture based on self-attention for machine translation. Furthermore, they noted that self-attention could help to build more interpretable models. However, a significant drawback of this approach is that it does not generate explanations that humans can understand. Zhang et al. [39] introduced interpretable CNNs. Specifically, each filter in a high convolutional layer encodes different object parts per object category, and they are activated by a single part of the object only. Their experiments showed that their interpretable CNNs encoded more meaningful information than their traditional non-interpretable counterparts. However, this approach is only applicable to different types of CNNs, and it requires datasets with ground-truth annotations of object landmarks.

Kanehira et al. [40] presented a system that classifies an input object and outputs linguistic explanations and examples that justify it. This architecture has four components, namely: an explainer, a selector, a reasoner, and a predictor. The predictor is the target model of the explanations, and it is not trained; while the explainer, the selector, and the reasoner are trained to explain the output of the predictor in a post-hoc manner. Although the model is capable of providing human-readable explanations, its post-hoc nature raises questions regarding this approach that we have already discussed. Furthermore, it requires a dataset with attributes assigned to be trained, such as the work in [39].

### 2.2 Autoencoders

Autoencoders are models that can transform dimensionality and aim to reconstruct the original input data. These architectures work with encoding and decoding functions,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3148331, IEEE Transactions on Dependable and Secure Computing



TABLE 1: Current approaches to explaining DNNs. In the *Approach to explanation* column, *I* stands for *Processing*, *II* for *Representation*, and *III* for *Generating explanations*. In the *Disadvantage* column, *1* stands for *post hoc model that does not explain the actual decisions of the DNN*, *2* for *explanations are difficult to understand*, *3* for *it requires dataset with ground-truth annotations of object landmarks*.

| Authors | Approach to explanation | Problem | Application domain | Datasets | Pub. | Evaluation metrics | Aval. | Dis. |
|---|---|---|---|---|---|---|---|---|
| Ribeiro et al. [31] | I<br>Linear model | Binary classification | Text classification<br>Image classification | Sentiment analysis dataset<br>Hand-selected images | Yes<br>No | Faithfulness of explanations<br>Trustworthiness of the model | Yes | 1 |
| Zilke et al. [32] | I<br>Decision tree | Binary classification | Image classification<br>Artificial problem<br>XOR problem | MNIST dataset<br>Letter Recognition<br>Artif I<br>Artif II | Yes<br>Yes<br>No<br>No | Successful attempts to rule extraction Comprehensibility and fidelity of rules | No | 1,2 |
| Augasta and Kathirvalavakumar [33] | I<br>Automatic-rule extraction | Binary classification<br>Multi-class classification | Plant classification<br>Medical diagnosis<br>Credit risk Signal processing | Iris dataset<br>Breast Cancer Wisconsin<br>Pima Indian diabetes<br>German credit card<br>Hepatitis dataset<br>Ionosphere dataset | Yes | Accuracy, comprehensibility, and fidelity of rules | No | 1 |
| Selvaraju et al. [34] | I<br>Saliency mapping | Multi-class classification | Image classification<br>Image captioning<br>VQA | ImageNet dataset<br>PASCAL VOC 2007 | Yes | Localization capability Class-discriminative visualizations<br>Trustworthiness of explanations | Yes | 1 |
| Shin et al. [36] | II<br>By layers | Binary classification | Medical imaging classification | Lymph node detection datasets<br>Multimedia database of interstitial lung diseases | Yes | Explanations were not directly evaluated | No | 2 |
| Yosinski et al. [37] | II<br>By units | Multi-class classification | Image classification | ImageNet dataset | Yes | Explanations were not directly evaluated | Yes | 2 |
| Vaswani et al. [38] | III<br>Attention-based | Multi-class classification | Machine translation | WMT 2014 English-German and English-French datasets | Yes | Explanations were not directly evaluated | Yes | 2 |
| Zhang et al. [39] | III<br>Disentangled representation | Single-class classification<br>Multi-class classification | Image classification | ImageNet dataset<br>CUB200-2011 dataset<br>PASCAL VOC 2007 | Yes | Part interpretability<br>Location instability | Yes | 3 |
| Kanehira et al. [40] | III<br>Generated explanations | Binary classification<br>Multi-class classification | Image classification | Aesthetics with Attributes Database<br>CUB200-2011 dataset | Yes | Accuracy and consistency of predictions<br>Complementarity of explanations | No | 1, 3 |

where the encoder computes a representation $h(x)$ of an input $x$ and the decoder maps the representation $h(x)$ back into the input space. This creates a reconstruction $g(h(x))$ of the input [22] – see Fig. 1.

The potential of using autoencoders for anomaly detection is explained in a recent literature review [21]. Existing approaches generally use unsupervised learning methods [21] although there are also approaches that use semi-supervised learning [41] and supervised learning [42]. However, autoencoders are usually non-linear because of the activation functions used in the neural network paradigm. This limits its ability to provide interpretability, which is an increasingly valued property in anomaly detection.

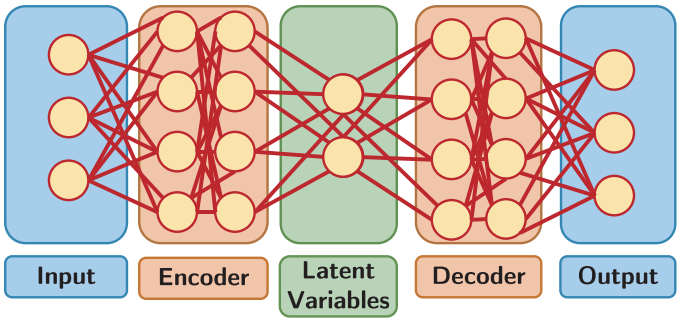


Fig. 1: Autoencoder architecture.

There have also been recent attempts to design autoencoding models using decision trees [24], [25]. Irsoy and Alpaydin [24], for example, presented an autoencoder based on soft trees. Classification with soft trees is achieved by following every path from the root node to the leaves, unlike traditional trees. In soft decision trees, a logistic model is used in every leaf. Although soft trees are reported to be more accurate, there is not a simple mechanism to explain how they work (unlike traditional decision trees that can be explained as a set of rules, which can be easily understood by the users).

Another autoencoder based on decision trees is *eForest*

[25], which is a tree ensemble that can encode and decode data. Encoding is performed by sending the input data through all the decision trees in the ensemble to find out in which leaf node the input data ends up. Decoding is achieved by using the Maximal Compatible Rule (MCR). MCR is defined by the decision paths the input data follows when it traverses down from the root nodes to all the leaves in all decision trees, that is, when being encoded. The authors showed that decision trees can outperform neural networks in some autoencoding tasks, namely: image reconstruction using MNIST and CIFAR-10. Moreover, the authors speculated that decision-tree-based autoencoders can handle categorical data without transforming it into numerical (i.e., no supporting evidence since their study focused on numerical data only).

We observe that existing autoencoders are not capable of working with categorical data. Consequently, they need to transform all categorical data into a numerical representation by using an encoding method, such as one hot, ordinal, sum, and binary [43], [44], [45]. However, when transforming data from categorical to numerical, the new representation contains significantly more attributes than the original representation, and this translates into higher computational complexity. In addition, all proposed encoding methods ignore every relation among values of categorical attributes, which can produce poor classification results [46].

## 3 A NOVEL INTERPRETABLE AUTOENCODER BASED ON DECISION TREES

As discussed earlier, existing autoencoders can handle numerical data only. When working with categorical data, a common solution is to use one-hot encoding. However, such an approach has two key limitations [46]. First, they are high-dimensional and sparse, and they ignore the relations among different values of categorical attributes. Second, existing autoencoders lack interpretability. Our proposed interpretable autoencoder is designed to mitigate these two limitations.

In our approach, given the value of an attribute, it correlates with some other values of the other attributes. For example, $weather$ value $very\_cold$ is correlated with $hemisphere$ value $north$ when $month$ is $December$. Hence, our hypothesis is that we can accurately model the structure of a dataset with an interpretable autoencoder if we automatically learn to encode and decode the values of the attributes based on their correlations with values of other attributes.

Fig. 2 shows the architecture of our proposed approach. In the training phase, our input is a dataset $T = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}\}$, in which every object $\mathbf{x}^{(i)}$ is represented by a tuple of attributes values $(x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}) \in D_1 \times D_2 \times \ldots \times D_n$. Here, $D_i$ is the definition domain of the i-$th$ categorical attribute, e.g. $D_1 = \{north, south, east, west\}$ and $D_2 = \{true, false\}$. Then, for each i-$th$ attribute, a new decision tree trains with the dataset $T$ (projected into the attributes different from the i-$th$ one) using the i-$th$ attribute as the target class.

In the classification phase, given an unseen object $\mathbf{x}$ represented by a tuple of attributes values $(x_1, x_2, \ldots, x_n)$, *decision tree 1* receives tuple $\mathbf{x}_{-1} = (x_2, x_3 \ldots, x_n)$ and outputs a vector $(h_{1,1}^{(1)}, h_{1,2}^{(1)} \ldots, h_{1,k_1}^{(1)})$ that represents how probable the obtained class is for every value in the attribute domain $D_1$. This procedure is repeated for each decision tree and eventually the argmax pooling layer outputs the values of the attributes corresponding to the highest probabilities output by each decision tree. The more similar the output object $\mathbf{x}'$ is to $\mathbf{x}$, the more similar $\mathbf{x}$ is to the knowledge learned from the training dataset.

TABLE 2: Synthetic objects to exemplify (in Fig. 3) how to build and use the model of our autoencoder. Each row represents an object, and each column represents an attribute. The first row includes the name of the attributes.

| $att1$ | $att2$ | $att3$ | $att4$ |
|---|---|---|---|
| $a$ | $p$ | $n$ | $u$ |
| $a$ | $p$ | $n$ | $u$ |
| $a$ | $p$ | $n$ | $u$ |
| $a$ | $p$ | $n$ | $u$ |
| $b$ | $p$ | $n$ | $u$ |
| $b$ | $q$ | $m$ | $w$ |
| $b$ | $q$ | $m$ | $w$ |
| $b$ | $q$ | $m$ | $w$ |
| $b$ | $q$ | $m$ | $w$ |
| $c$ | $q$ | $m$ | $w$ |
| $c$ | $q$ | $m$ | $u$ |
| $c$ | $q$ | $m$ | $u$ |
| $c$ | $q$ | $m$ | $u$ |

We will explain how our proposed approach works using the dataset in Table 2. In Fig. 3, one can observe that the top box has three outgoing connections to the top output neuron, while the rest of the boxes have only two outgoing connections. This is because the top box contains a tree with class $att1$, whose domain contains three values ($\{a, b, c\}$), while the trees in the other boxes use, as classes, attributes that only have two possible values each.

Let us focus on the decision tree at the top. In the root node, there is a vector $[4, 5, 4]$ of length $|D_1|$ since its target is $att1$. Each number represents how many objects in the dataset in Table 2 have a particular attribute value for $att1$. Initially, there are four objects with $att1 = a$, five with $att1 = b$, and four with $att1 = c$. As we go deeper in the tree, we observe that there are four objects with $att1 = a$ and $att2 = p$, one with $att1 = b$ and $att2 = p$, and none with the combination $att1 = c$ and $att2 = p$. This explanation applies to the other nodes in the tree, and for the other trees as well.

Fig. 3 shows that when inputting the tuple $(a, p, m, w)$, the autoencoder encodes it to $(0.8, 0.2, 0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0)$, and it decodes it to $(a, q, n, u)$, which is similar to the input only in $att1$. When we study the training dataset in Table 2, we observe that there is no combination of three attribute values matching the input tuple $(a, p, m, w)$. This indicates that the input tuple is an anomaly of the training dataset. In order to interpret how the autoencoder decode this tuple, we find the rules that classify the tuple in each tree as follows:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3148331, IEEE Transactions on Dependable and Secure Computing



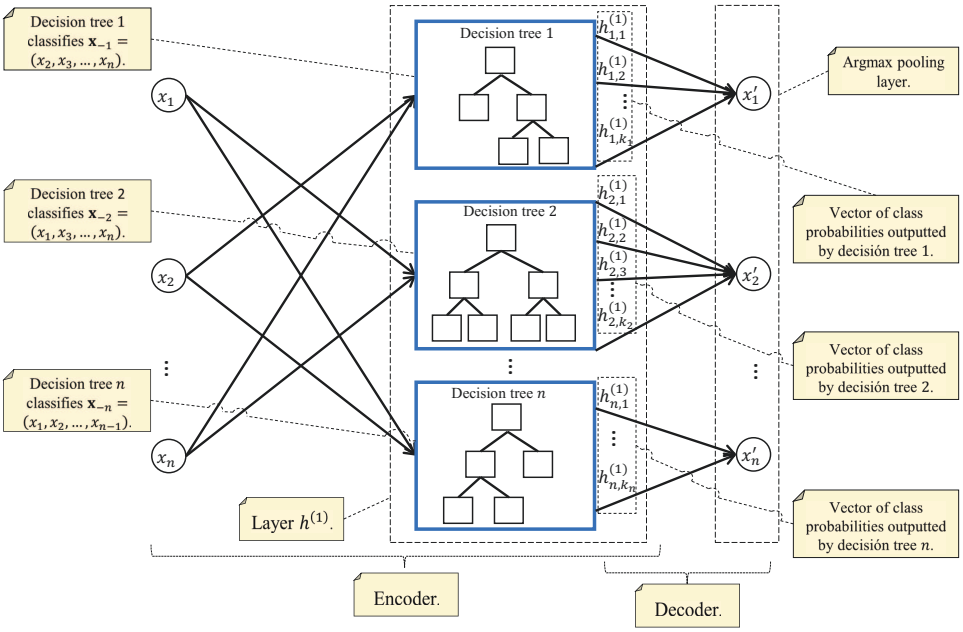


Fig. 2: Architecture of our proposed decision tree-based autoencoder for categorical attributes. In the architecture, the i-th decision tree trains with all the attributes except the i-th attribute. In the classification phase, the i-th decision tree receives a tuple $\mathbf{x}_{-i}$ that contains all the attributes values except $x_i$, and outputs a probability vector which the argmax pooling layer uses to infer $x'_i$. The length of the i-th vector depends on the number of possible values in the definition domain $D_i$ of the i-th categorical attribute. The end-user can look at the path traversed by the object $\mathbf{x}_{-i}$ into the i-th tree, because this path indicates the pattern learned from the training dataset. Hence, the classification result is interpretable.

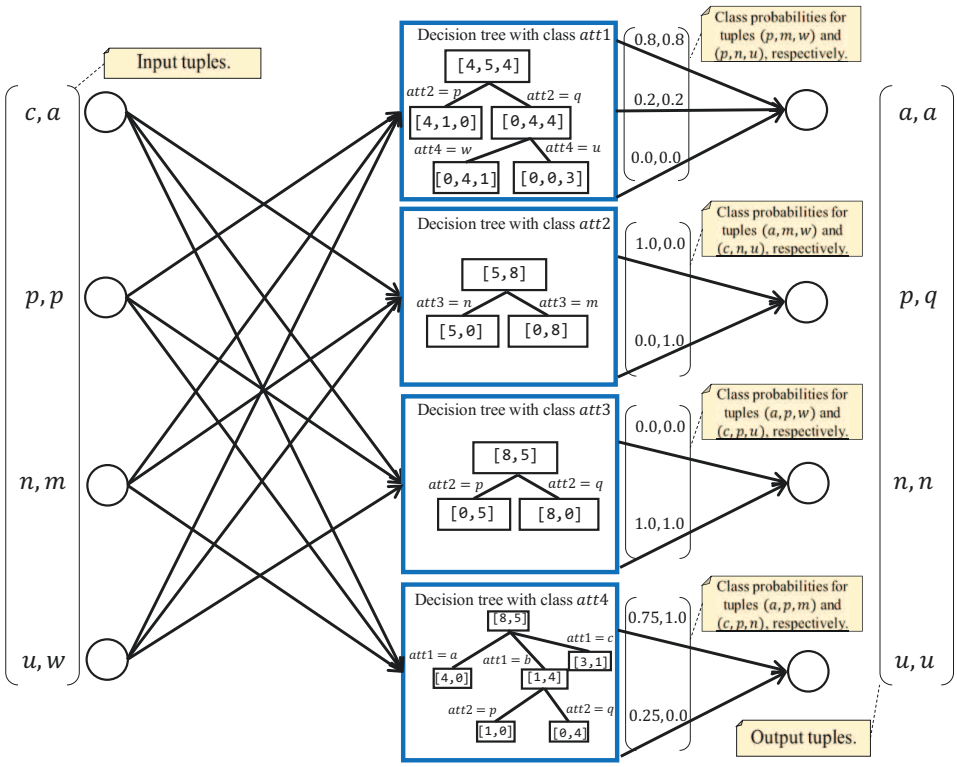


Fig. 3: Model of architecture in Fig. 2, which is trained with the dataset in Table 2. The figure shows the outputs of layer 1 and the decoded tuples when inputting the objects $(a, p, m, w)$ and $(c, p, n, u)$ in that order.

1) The top tree classifies $(p, m, w)$ according to the rule $(att2 = p) \implies (att1 = a)$. This is the only rule that correctly decodes one attribute value $(att1 = a)$.

2) The second tree, from top to bottom, classifies $(a, m, w)$ according to the rule $(att3 = m) \implies (att2 = q)$. This explains why the autoencoder incorrectly decodes the

attribute value $att2 = p$ as $att2 = q$.

3) The third tree, from top to bottom, classifies $(a, p, w)$ according to the rule $(att2 = p) \implies (att3 = n)$. This explains why the autoencoder incorrectly decodes the attribute value $att3 = m$ as $att3 = n$.
4) The fourth tree, from top to bottom, classifies $(a, p, m)$ according to the rule $(att1 = a) \implies (att4 = u)$. This explains why the autoencoder incorrectly decodes the attribute value $att4 = w$ as $att4 = u$.

Fig. 3 shows that when inputting the tuple $(c, p, n, u)$, the autoencoder encodes it to $(0.8, 0.2, 0.0, 1.0, 0.0, 0.0, 1.0, 0.75, 0.25)$, and it decodes it to $(a, p, n, u)$ which is similar to the input in three out of four attributes. In the training dataset (Table 2), the combination $(p, n, u)$ of the input tuple appears five times. In order to interpret how the autoencoder decodes this tuple, we find the rules that classify the tuple in each tree as follows:

1) The top tree classifies $(p, n, u)$ according to the rule $(att2 = p) \implies (att1 = a)$. This explains why the autoencoder decodes the value $att1 = c$ as $att1 = a$.
2) The second tree, from top to bottom, classifies $(c, n, u)$ according to the rule $(att3 = n) \implies (att2 = p)$. This explains why the autoencoder correctly decodes the attribute value $att2 = p$.
3) The third tree, from top to bottom, classifies $(c, p, u)$ according to the rule $(att2 = p) \implies (att3 = n)$. This explains why the autoencoder correctly decodes the attribute value $att3 = n$.
4) The fourth tree, from top to bottom, classifies $(c, p, n)$ according to the rule $(att1 = c) \implies (att4 = u)$. This explains why the autoencoder correctly decodes the attribute value $att4 = u$.

During the induction procedure, our approach builds $n$ decision trees from the $m$ objects of the training dataset projected into $n - 1$ attributes. The time complexity for building an unpruned decision tree is $O(m*log(m))$. Hence, the time complexity for building the model of the proposed autoencoder is $O(n * m * log(m))$.

Testing an autoencoder depends on the application domain. The next section introduces the experimental framework to evaluate the utility of the proposed autoencoder in anomaly detection. However, it is necessary to add an additional layer in our architecture to allow us to return a single value (see Fig. 4). Each weight $w_{i,j}^{(1)}$ is the recall of the i-th decision tree for the j-th attribute value in the definition domain $D_i$ (the recall measure is computed from the confusion matrix obtained after five-fold cross-validation). Each weight $w_i^{(2)}$ is the AUC [47] of the i-th decision tree computed from the same confusion matrix obtained after five-fold cross-validation. The algorithm source code in C# and Python can be found at https://github.com/miguelmedinaperez/DTAE and https://github.com/ebucheli/DT-Autoencoder, respectively.

## 4 Evaluation

Here, we will present the selected datasets, the k stratified cross-validation procedure, the selected classifiers for anomaly detection, and the performance metrics used to evaluate the selected classifiers. We will also present the findings in this section.

### 4.1 Setup

Table 3 describes the 28 datasets used in our experimental setup, which were obtained from the UCI Machine Learning Repository [48]. For each dataset, Table 3 shows the number of attributes (#Att.) and the number of objects by class (# Obj. by Class). The tested datasets range from four to 69 for the number of attributes, from 20 to 12,960 for the number of objects, and from one to 47.79 for the class imbalance ratio, see (1) [49].

$$IR = \frac{\mid Class_{maj} \mid}{\mid Class_{min} \mid} \tag{1}$$

In the above equation, $\mid Class_{maj} \mid$ denotes the number of objects belonging to the majority class and $\mid Class_{min} \mid$ is the number of objects belonging to the minority class. Furthermore, most of the datasets contain imbalanced classes [49], and the numbers of attributes and objects are heterogeneous, which is typical of anomaly detection [50], [51].

We transformed each dataset with two balanced classes or with multiple imbalanced classes into several datasets (the obtained datasets can be differentiated by a suffix in the dataset name), changing the minority class every time because we want to measure the performance of the algorithms for anomaly detection. Then, for each dataset in Table 3, we executed a five-fold *Distribution Optimally Balanced Stratified Cross-Validation* (DOB-SCV) procedure, as suggested by [52], [49]. Lastly, we moved the objects of the minority class from each training partition to its respective testing partition. Therefore, we transformed every binary classification dataset into an anomaly detection dataset.

We compared the performance of our proposed DTAE with six other state-of-the-art classifiers, namely: One-class SVM [27], Isolation Forest [28], Local Outlier Factor [29], Elliptic Envelope [30], Gaussian Mixture Model, and eForest [25]. In the performance evaluation, we leveraged the scikit-learn library [53] using the parameter values recommended by their authors (see also Table 4). For each of the six classifiers, we used the one-hot encoding method [54] to transform the values of all categorical attributes because it provides better results than other state-of-the-art encoding methods [43], [44], [45]. By using the one-hot encoding method, all categorical attributes were encoded by creating a binary column for each value, and this procedure returns a sparse matrix or dense array [54].

Some existing literature suggests that a threshold should be set so as to differentiate between normal objects and anomalies [21]. However, this necessitates the involvement of human experts because of their knowledge and experience [55]. Thus, we chose two evaluation metrics that take into account all possible thresholds to assess the selected classifiers. First, we utilized the Area Under the Receiver Operating Characteristic Curve (AUC) measure [56]. AUC was selected because it is an objective measure (i.e., not

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3148331, IEEE Transactions on Dependable and Secure Computing



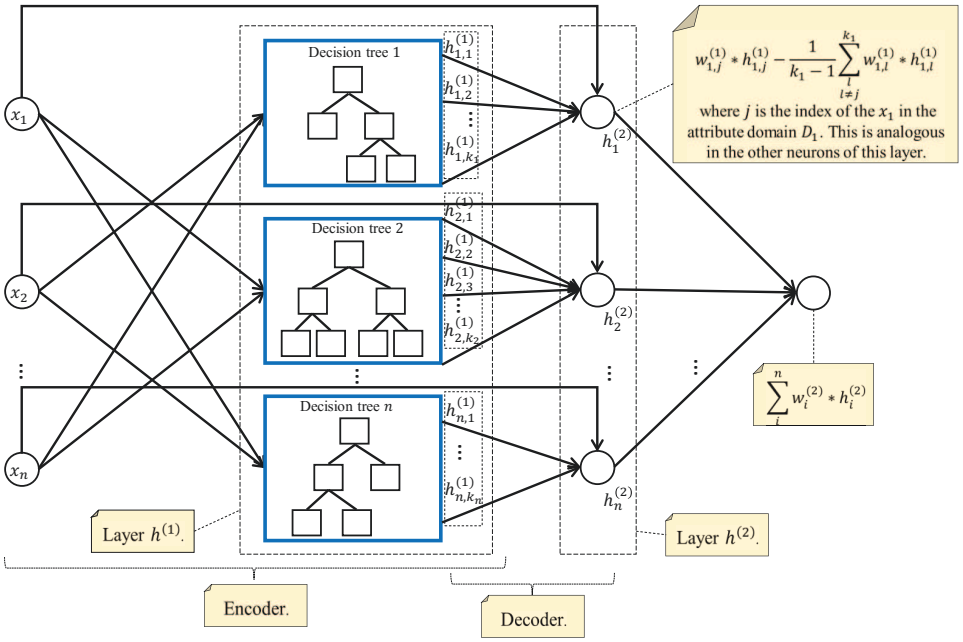


Fig. 4: An anomaly detection architecture from the autoencoder in Fig. 2.

TABLE 3: All datasets used in our experimental setup, taken from the UCI Machine Learning Repository [48].

| Name | # Att. | # Obj. by Class | Name | # Att. | # Obj. by Class |
|---|---|---|---|---|---|
| audiology | 69 | 207/19 | balloons | 4 | 12/8 |
| breast-cancer | 9 | 201/85 | car | 6 | 1663/65 |
| chess | 36 | 1669/1527 | hayes-roth | 4 | 129/31 |
| hiv-1 | 8 | 5230/1360 | lenses | 4 | 15/9 |
| lymphography | 18 | 81/67 | molecular_Promoter− | 57 | 53/53 |
| molecular_Promoter+ | 57 | 53/53 | monks-1-0 | 6 | 278/278 |
| monks-1-1 | 6 | 278/278 | mushroom | 22 | 4208/3916 |
| nursery | 8 | 12630/330 | postoperative-patient-data | 8 | 64/26 |
| primary-tumor | 17 | 318/21 | solar-flare1 | 12 | 294/29 |
| solar-flare2 | 12 | 1023/43 | soybean-l | 35 | 669/14 |
| soybean-s-D1 | 35 | 37/10 | soybean-s-D2 | 35 | 37/10 |
| soybean-s-D3 | 35 | 37/10 | spect | 22 | 212/55 |
| splice | 60 | 2423/767 | sponge | 44 | 70/6 |
| tic-tac-toe | 10 | 626/332 | vote | 16 | 267/168 |

TABLE 4: **Parameters used for the tested classifiers.**

| Classifier | Parameters |
|---|---|
| One-class SVM [27] | kernel='rbf', degree=3, gamma='scale', coef0=0.0, tol=0.001, nu=0.5, shrinking=True, cache_size=200, verbose=False, max_iter=-1 |
| Isolation Forest [28] | n_estimators=100, max_samples='auto', contamination='auto', max_attributes=1.0, bootstrap=False, n_jobs=None, behaviour='new', random_state=970, verbose=0, warm_start=False |
| Local Outlier Factor [29] | n_neighbors=20, algorithm='auto', leaf_size=30, metric='minkowski', p=2, metric_params=None, contamination='auto', novelty=False, n_jobs=None |
| Elliptic Envelope [30] | store_precision=True, assume_centered=False, support_fraction=None, contamination=0.1, random_state=None |
| Gaussian Mixture Model | n_components=1, covariance_type='full', tol=0.001, reg_covar=1e-06, max_iter=100, n_init=1, init_params='kmeans', weights_init=None, means_init=None, precisions_init=None, random_state=None, warm_start=False, verbose=0, verbose_interval=10 |
| eForest [25] | n_trees=1000 |

affected by subjective indicators) and it is insensitive to changes in the distribution of both training and testing datasets [47], [57], [58]. The AUC was computed directly from the ROC curve (True Positive Rate versus False Positive Rate) [47]. Second, we evaluated the classifiers with the Area Under the Precision vs. Recall Curve (AUC-PR) [59]. Furthermore, we used the Average Precision Score as an estimate of AUC-PR [60]. We utilized this metric as it has been recommended as an alternative for domains with highly imbalanced classes [59], [60].

Finally, to know if there are statistical differences among the tested classifiers, we performed a comparison among all obtained classification results using the Friedman non-parametric test, which provides a ranking. Additionally, the Finner post-hoc procedure was applied, as suggested in [61], [49].

### 4.2 Findings

Now, we will present the evaluation findings, including the statistical tests performed to determine whether there are statistical differences between the classification results.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3148331, IEEE Transactions on Dependable and Secure Computing



Fig. 5 and Fig. 6 are the box-plots of all AUC and Average Precision values. The box-plot shows the minimum and maximum values, the median (black line inside the box), and the first and third quartiles (top and bottom sides of the box, respectively) for the evaluation metric. Small boxes and whiskers closer to the median indicate lower variability in the measure; the lower variability and the higher median value, the better and more consistent the results. Values considered as outliers are usually shown as dots outside the whiskers. The best possible value for AUC and Average Precision is one that corresponds to perfect classification.

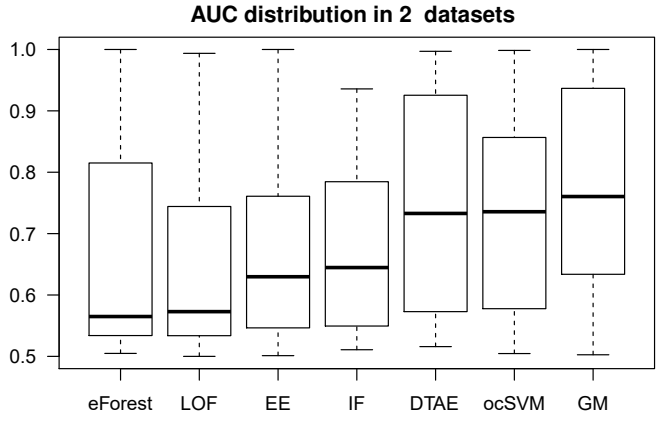


Fig. 5: A box-plot showing the distribution of the AUC [56] obtained by all seven classifiers using the 28 datasets of UCI Machine Learning repository [48]. The algorithms are sorted from left to right in ascending order of the median: eForest [25], LOF (Local Outlier Factor) [29], EE (Elliptic Envelope) [30], IF (Isolation Forest) [28], proposed DTAE, ocSVM (One-class SVM) [27], and GM (Gaussian Mixture).

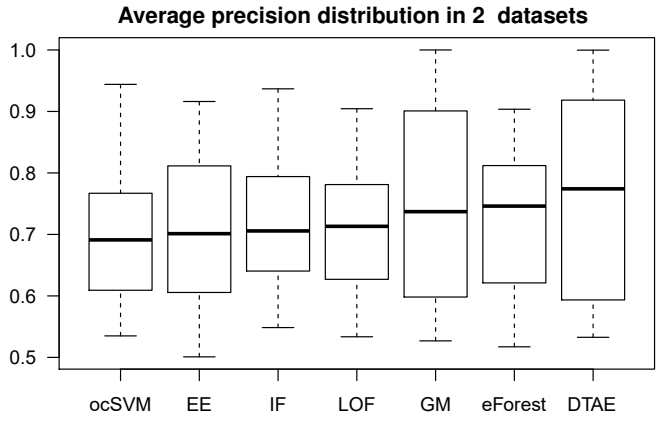


Fig. 6: A box-plot showing the distribution of the Average Precision Score [59] obtained by all seven tested classifiers in 28 datasets of UCI Machine Learning repository [48]. The algorithms are sorted from left to right in ascending order of the median: ocSVM (One-class SVM) [27], EE (Elliptic Envelope) [30], IF (Isolation Forest) [28], LOF (Local Outlier Factor) [29], GM (Gaussian Mixture), eForest [25], and DTAE.

One can observe from Fig. 5 that in terms of AUC, DTAE outperforms Elliptic Envelope, Isolation Forest, and the other decision-tree-based autoencoder from the literature (i.e., eForest). Besides, the third quartile of One-class SVM is above that of DTAE. However, the first quartile of DTAE is far above the one of One-class SVM. Yet both quartiles of the Gaussian Mixture Model outperform those of the other classifiers.

For the Average Precision Score, one can observe from Fig. 6 that the distribution of DTAE is similar to that of the Gaussian Mixture Model, but the first quartile of DTAE is better. Additionally, our autoencoder is better than eForest as well. Hence, one can conclude that DTAE outperforms all the classifiers. A closer inspection of the results presented in Table 6 also reveals that DTAE achieves good AUC and Average Precision.

Table 7 and Table 8 show the average of AUC and Average Precision, respectively. They also include the corresponding standard deviation (SD), the average ranking according to the Friedman's test, and the adjusted $p$-value of the Finner's procedure for all tested classifiers (based on all the datasets outlined in Table 3). The findings in Table 7 and Table 8 are ordered according to the average of the Friedman's ranking value, and the thin horizontal line indicates the point after which there is a statistically significant difference with the best result in the Friedman's ranking ($p$-value $< 0.05$).

Table 7 shows that DTAE significantly outperforms eForest, Isolation Forest, Elliptic Evelop, and Local Outlier Factor. Although Gaussian Mixture Model and One-class SVM have better performance than DTAE, these results show that there are no statistical differences among them. Considering Table 8, DTAE has the best position based on Friedman's ranking, without statistical differences.

In summary, Table 5 shows that our proposal is, on average, 2% below the best AUC result, and according to Table 7 that there are no statistical differences. Table 6 shows that our proposal is, on average, 3% over the other classifiers as to Average Precision without statistically significant differences (see Table 8). Moreover, based on our earlier discussion in Section 3, we can conclude that our proposal is the first autoencoder working with categorical attributes that does not require us to transform the data from categorical into numerical. Additionally, our proposal can be understood by experts in the application area in a language close to the one they use.

## 5 Conclusions and Future Work

Given the potential of deep learning algorithms to solve real-world problems such as anomaly detection and forensic investigation, it is not surprising that designing more efficient and effective deep learning algorithms is an active research area. One existing trend is to design models that allow users to understand the associated classification results. For example, how we can develop the explanatory capability of opaque models such as autoencoders to provide the reasoning behind their decisions. In this paper, we introduced DTAE, the first interpretable autoencoder for anomaly detection using categorical data. DTAE is a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3148331, IEEE Transactions on Dependable and Secure Computing



TABLE 5: Results, in terms of to AUC [56], for all the tested classifiers, considering all the tested databases. The best results per database appear in bold.

| Dataset | eForest [25] | Local Outlier Factor [29] | Elliptic Envelope [30] | Isolation Forest [28] | ocSVM [27] | Gaussian Mixture Model | DTAE |
|---|---|---|---|---|---|---|---|
| audiology | 0.7336 | 0.6470 | 0.7140 | 0.6444 | 0.7395 | **0.8093** | 0.6700 |
| balloons | 0.8750 | 0.5750 | 0.8500 | 0.5125 | 0.8333 | 0.8500 | **0.9417** |
| breast-cancer | 0.5270 | 0.6965 | 0.6396 | 0.6803 | **0.6999** | 0.6433 | 0.6643 |
| car | 0.5959 | 0.5000 | 0.7253 | 0.5567 | 0.7520 | 0.7530 | **0.7703** |
| chess | 0.5055 | 0.7358 | 0.6084 | 0.6450 | 0.6809 | 0.7384 | **0.7647** |
| hayes-roth | **1.0000** | 0.9937 | 0.9673 | 0.8081 | 0.9985 | 0.9967 | 0.6840 |
| hiv-1 | 0.5619 | 0.5110 | 0.5012 | **0.5778** | 0.5173 | 0.5026 | 0.5552 |
| lenses | 0.7037 | 0.5963 | 0.7222 | **0.7556** | 0.6000 | **0.7556** | 0.5333 |
| lymphography | 0.7499 | 0.7901 | 0.7724 | 0.7673 | **0.8181** | 0.7974 | 0.7303 |
| molecular_Promoter+ | 0.8238 | 0.9660 | 0.6091 | 0.8141 | **0.9787** | 0.9394 | 0.9283 |
| molecular_Promoter- | 0.5427 | 0.5318 | 0.5067 | 0.5197 | 0.6105 | **0.6298** | 0.5490 |
| monks-1-0 | 0.8063 | 0.7869 | 0.5261 | 0.7344 | 0.7922 | **0.8183** | 0.5261 |
| monks-1-1 | 0.5513 | 0.5708 | 0.6060 | 0.5270 | 0.5046 | 0.5422 | **0.9765** |
| mushroom | **0.9637** | 0.9129 | 0.6261 | 0.8019 | 0.9002 | 0.9339 | 0.9598 |
| nursery | 0.5327 | 0.5000 | 0.6174 | 0.6290 | **0.7905** | 0.7595 | 0.6929 |
| postoperative-patient-data | 0.5218 | 0.5072 | 0.5310 | 0.5108 | 0.5303 | 0.5319 | **0.5648** |
| primary-tumor | 0.5352 | 0.5074 | **0.5618** | 0.5341 | 0.5503 | 0.5488 | 0.5356 |
| solar-flare1 | 0.5051 | 0.5612 | 0.5064 | 0.5201 | 0.5553 | 0.5911 | **0.6216** |
| solar-flare2 | 0.5048 | 0.5395 | 0.7402 | **0.7518** | 0.7318 | 0.6376 | 0.7507 |
| soybean-l | 0.9458 | 0.6145 | 0.5741 | 0.8826 | 0.9778 | 0.9761 | **0.9970** |
| soybean-s-D1 | 0.9900 | 0.6763 | 0.8183 | 0.5646 | 0.8800 | **1.0000** | 0.9550 |
| soybean-s-D2 | **1.0000** | 0.9825 | **1.0000** | 0.8758 | 0.9975 | **1.0000** | 0.9675 |
| soybean-s-D3 | 0.5493 | 0.5449 | 0.8158 | 0.7576 | 0.5562 | **1.0000** | 0.9225 |
| spect | 0.5679 | 0.7525 | 0.7494 | **0.8128** | 0.7910 | 0.7620 | 0.7806 |
| splice | 0.5100 | 0.5356 | 0.5207 | 0.5873 | **0.5992** | 0.5232 | 0.5159 |
| sponge | **0.7452** | 0.5381 | 0.6333 | 0.5738 | 0.6452 | 0.7167 | 0.5810 |
| tic-tac-toe | 0.5436 | 0.5026 | 0.5211 | 0.5422 | 0.5217 | **0.9720** | 0.7354 |
| vote | 0.5414 | 0.5590 | 0.7826 | 0.9358 | **0.9411** | 0.7613 | 0.9022 |

TABLE 6: Results, in terms of Average Precision [59], for all the tested classifiers, considering all the tested databases. The best results per database appear in bold.

| Dataset | eForest [25] | Local Outlier Factor [29] | Elliptic Envelope [30] | Isolation Forest [28] | ocSVM [27] | Gaussian Mixture Model | DTAE |
|---|---|---|---|---|---|---|---|
| audiology | 0.5624 | 0.7378 | 0.7546 | 0.7403 | 0.7687 | **0.9123** | 0.8165 |
| balloons | 0.8034 | 0.7677 | 0.7061 | 0.7959 | 0.7144 | 0.7333 | **0.8900** |
| breast-cancer | **0.6810** | 0.6006 | 0.6079 | 0.6064 | 0.5988 | 0.5397 | 0.5719 |
| car | 0.8756 | 0.8365 | 0.7912 | 0.7265 | 0.5418 | 0.7407 | **0.9413** |
| chess | **0.7992** | 0.7237 | 0.7430 | 0.7730 | 0.7556 | 0.5934 | 0.5738 |
| hayes-roth | 0.7132 | 0.6474 | 0.6444 | 0.5983 | 0.6479 | **1.0000** | 0.6604 |
| hiv-1 | 0.5172 | 0.5979 | 0.5569 | **0.6476** | 0.5569 | 0.5532 | 0.6052 |
| lenses | 0.6212 | **0.7739** | 0.6804 | 0.6556 | 0.7659 | 0.5878 | 0.5417 |
| lymphography | **0.8657** | 0.6858 | 0.6887 | 0.6943 | 0.6651 | 0.5440 | 0.5817 |
| molecular_Promoter+ | **0.8909** | 0.6573 | 0.8819 | 0.7091 | 0.6538 | 0.8125 | 0.7832 |
| molecular_Promoter- | 0.7696 | 0.8423 | **0.8661** | 0.8275 | 0.8146 | 0.7195 | 0.7866 |
| monks-1-0 | 0.8969 | 0.7389 | 0.8547 | **0.9269** | 0.7681 | 0.6030 | 0.7020 |
| monks-1-1 | 0.7884 | 0.8252 | 0.8091 | 0.8345 | 0.8312 | 0.5417 | **0.9590** |
| mushroom | **0.9036** | 0.6653 | 0.7227 | 0.6934 | 0.6659 | 0.8926 | 0.8520 |
| nursery | 0.8792 | 0.8845 | 0.5104 | 0.6676 | 0.6153 | 0.7949 | **0.9221** |
| postoperative-patient-data | 0.6181 | 0.6848 | 0.6964 | **0.7024** | 0.6913 | 0.6185 | 0.5578 |
| primary-tumor | 0.7469 | 0.7326 | 0.7224 | 0.7089 | 0.7039 | **0.7572** | 0.7352 |
| solar-flare1 | 0.7030 | 0.5335 | 0.5009 | 0.5485 | 0.5992 | 0.6776 | **0.7629** |
| solar-flare2 | 0.8205 | 0.8104 | 0.8623 | 0.8744 | 0.8781 | 0.9092 | **0.9281** |
| soybean-l | 0.7907 | 0.9045 | 0.9162 | 0.9368 | 0.9440 | 0.9976 | **0.9997** |
| soybean-s-D1 | 0.7194 | 0.5860 | 0.5103 | 0.6069 | 0.5350 | **1.0000** | 0.9633 |
| soybean-s-D2 | 0.7211 | 0.6004 | 0.6033 | 0.5588 | 0.6030 | **1.0000** | 0.9671 |
| soybean-s-D3 | 0.5433 | 0.6735 | 0.5129 | 0.7922 | 0.6307 | **1.0000** | 0.9147 |
| spect | 0.5566 | 0.7883 | 0.8138 | **0.8424** | 0.8272 | 0.6897 | 0.6879 |
| splice | 0.6007 | 0.6065 | 0.6166 | 0.6852 | **0.7007** | 0.5269 | 0.5557 |
| sponge | 0.6211 | 0.5717 | 0.6865 | 0.6332 | 0.6909 | **0.8651** | 0.7651 |
| tic-tac-toe | 0.7469 | 0.7026 | 0.8150 | 0.7390 | 0.7361 | **0.8787** | 0.5327 |
| vote | 0.7452 | 0.7522 | 0.5908 | 0.5673 | 0.5654 | 0.7281 | **0.8566** |

decision-tree-based autoencoder, designed to detect anomalies and provide the explanations behind its decisions by finding the correlations among different attribute values. Performance evaluation of DTAE and six other state-of-the-art classifiers demonstrated that our classifier achieves good performance, with a minimal trade-off between per-

TABLE 7: Statistical results, according to AUC [56], for all the tested classifiers, considering all the tested databases. The algorithms are sorted from top to bottom, according to the Friedman ranking [62]. The last column shows the adjusted p-value obtained with the Finner post-hoc [62] when comparing the best-ranked algorithm with the rest of the algorithms. The thin horizontal line indicates the point after which there is a statistically significant difference with the best result in Friedman's ranking ($p$-value $\leq 0.05$).

| Classifier | Average of AUC | Standard deviation | Ranking | Adjusted $p$-value |
|---|---|---|---|---|
| Gaussian Mixture | 0.7675 | 0.1654 | 2.7143 | - |
| One-class SVM [27] | 0.7319 | 0.1637 | 3.1071 | 0.4962 |
| DTAE | 0.7420 | 0.1660 | 3.3571 | 0.3094 |
| eForest [25] | 0.6761 | 0.1799 | 4.3571 | 0.0066 |
| Isolation Forest [28] | 0.6722 | 0.1323 | 4.6607 | 0.0015 |
| Elliptic Envelope [30] | 0.6695 | 0.1388 | 4.7321 | 0.0014 |
| Local Outlier Factor [29] | 0.6476 | 0.1585 | 5.0714 | 0.0003 |

TABLE 8: Statistical results, according to Average Precision [59], for all the tested classifiers, considering all the tested databases. The algorithms are sorted from top to bottom, according to the Friedman ranking [62]. The last column shows the adjusted p-value obtained with the Finner post-hoc [62] when comparing the best-ranked algorithm with the rest of the algorithms.

| Classifier | Average of Average Precision | Standard deviation | Ranking | Adjusted $p$-value |
|---|---|---|---|---|
| DTAE | 0.7647 | 0.1561 | 3.4286 | - |
| eForest [25] | 0.7321 | 0.1167 | 3.7857 | 0.6022 |
| Isolation Forest [28] | 0.7176 | 0.1071 | 3.7857 | 0.6022 |
| Gaussian Mixture | 0.7577 | 0.1651 | 3.8571 | 0.6009 |
| Elliptic Envelope [30] | 0.7023 | 0.1243 | 4.2500 | 0.3253 |
| One-class SVM [27] | 0.6953 | 0.1046 | 4.3929 | 0.3253 |
| Local Outlier Factor [29] | 0.7118 | 0.0994 | 4.5000 | 0.3253 |

formance and interpretability.

However, like any study, our proposed approach is not without limitation. First, our architecture should be used in datasets with less than a thousand attributes because it builds a tree for each attribute. Building thousands of trees is time-consuming, although this limitation may be overcome with access to better computing resources. Second, our proposal may fail to build decision trees for attributes with tens of different values in the definition domain. In the near future, our plan is to extend our proposal to work simultaneously with categorical and numerical data. Furthermore, we plan to train the weights of the anomaly detector using stochastic gradient descent.

## Acknowledgments

This work is partly supported by the National Council of Science and Technology of Mexico under the scholarship grant 1006864. The authors also thank Drs. Bárbara Cervantes, Raúl Monroy, Jose E. Ramirez-Marquez, and Aythami Morales-Moreno because our discussions motivated this study.

## References

[1] A. George, Z. Mostaani, D. Geissenbuhler, O. Nikisins, A. Anjos, and S. Marcel, "Biometric face presentation attack detection with multi-channel convolutional neural network," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 42–55, 2020.

[2] H. Chen, G. Hu, Z. Lei, Y. Chen, N. M. Robertson, and S. Z. Li, "Attention-based two-stream convolutional networks for face spoofing detection," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 578–593, 2020.

[3] A. Genovese, V. Piuri, K. N. Plataniotis, and F. Scotti, "Palmnet: Gabor-pca convolutional networks for touchless palmprint recognition," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 12, pp. 3160–3174, 2019.

[4] H. Shi, Y. Zhang, Z. Zhang, N. Ma, X. Zhao, Y. Gao, and J. Sun, "Hypergraph-induced convolutional networks for visual classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 10, pp. 2963–2972, Oct 2019.

[5] F. Liu, L. Jiao, and X. Tang, "Task-oriented gan for polsar image classification and clustering," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2707–2719, Sep. 2019.

[6] Z. Gao, X. Wang, Y. Yang, C. Mu, Q. Cai, W. Dang, and S. Zuo, "Eeg-based spatio–temporal convolutional neural network for driver fatigue evaluation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2755–2763, Sep. 2019.

[7] Z. Zhao, P. Zheng, S. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, Nov 2019.

[8] Y. Chen, J. Wang, B. Zhu, M. Tang, and H. Lu, "Pixelwise deep sequence learning for moving object detection," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 9, pp. 2567–2579, 2019.

[9] J. Song, Y. Guo, L. Gao, X. Li, A. Hanjalic, and H. T. Shen, "From deterministic to generative: Multimodal stochastic rnns for video captioning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 10, pp. 3047–3058, Oct 2019.

[10] H. Kwon, Y. Kim, H. Yoon, and D. Choi, "Selective audio adversarial example in evasion attack on speech recognition system," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 526–538, 2020.

[11] L. Wen, L. Gao, and X. Li, "A new deep transfer learning based on sparse auto-encoder for fault diagnosis," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 1, pp. 136–144, 2019.

[12] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[13] O. Loyola-González, "Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view," *IEEE Access*, vol. 7, pp. 154 096–154 113, 2019.

[14] N. D. Truong, A. D. Nguyen, L. Kuhlmann, M. R. Bonyadi, J. Yang, S. Ippolito, and O. Kavehei, "Convolutional neural networks for seizure prediction using intracranial and scalp electroencephalogram," *Neural Networks*, vol. 105, pp. 104 – 111, 2018.

[15] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, 2018, pp. 80–89.

[16] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 93:1–93:42, 2018.

[17] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *ArXiv*, vol. abs/1702.08608, 2017.

[18] J. Pereira and M. Silveira, "Learning representations from healthcare time series data for unsupervised anomaly detection," in *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2019, pp. 1–7.

[19] B. Wang, D. Liu, X. Peng, and Z. Wang, "Data-driven anomaly detection of uav based on multimodal regression model," in *2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 2019, pp. 1–6.

[20] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, 2009.

[21] H. Wang, M. J. Bah, and M. Hammad, "Progress in outlier detection techniques: A survey," *IEEE Access*, vol. 7, pp. 107 964–108 000, 2019.

[22] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, 2010.

[23] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, vol. 7, pp. 53 040–53 065, 2019.

[24] O. Irsoy and E. Alpaydin, "Autoencoder trees," *ArXiv*, vol. abs/1409.7461, 2015.

[25] J. Feng and Z.-H. Zhou, "Autoencoder by forest," *ArXiv*, vol. abs/1709.09018, 2018.

[26] Y. Yang, I. G. Morillo, and T. M. Hospedales, "Deep neural decision trees," *ArXiv*, vol. abs/1806.06988, 2018.

[27] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, Jul. 2001.

[28] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, Dec 2008, pp. 413–422.

[29] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," *SIGMOD Rec.*, vol. 29, no. 2, p. 93–104, 2000.

[30] P. Rousseeuw and K. Driessen, "A fast algorithm for the minimum covariance determinant estimator," *Technometrics*, vol. 41, pp. 212–223, 1999.

[31] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. ACM, 2016, pp. 1135–1144.

[32] J. R. Zilke, E. Loza Mencía, and F. Janssen, "Deepred – rule extraction from deep neural networks," in *Discovery Science*, T. Calders, M. Ceci, and D. Malerba, Eds. Springer International Publishing, 2016, pp. 457–473.

[33] M. G. Augasta and T. Kathirvalavakumar, "Reverse engineering the neural networks for rule extraction in classification problems," *Neural Processing Letters*, vol. 35, no. 2, pp. 131–150, 2012.

[34] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 618–626.

[35] L. Oscar, L. Hao, C. Chaofan, and R. Cynthia, "Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 2018, pp. 3530–3537.

[36] H. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[37] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," 2015. [Online]. Available: https://arxiv.org/abs/1506.06579

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 5998–6008. [Online]. Available: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[39] Q. Zhang, Y. N. Wu, and S. Zhu, "Interpretable convolutional neural networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8827–8836.

[40] A. Kanehira and T. Harada, "Learning to explain with complemental examples," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[41] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, and L. Benini, "A semisupervised autoencoder-based approach for anomaly detection in high performance computing systems," *Eng. Appl. of AI*, vol. 85, pp. 634–644, 2019. [Online]. Available: https://doi.org/10.1016/j.engappai.2019.07.008

[42] Y. Kawachi, Y. Koizumi, and N. Harada, "Complementary set variational autoencoder for supervised anomaly detection," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 2366–2370.

[43] K. Potdar, T. S. Pardawala, and C. D. Pai, "A comparative study of categorical variable encoding techniques for neural network classifiers," *International Journal of Computer Applications*, vol. 175, no. 4, pp. 7–9, 2017.

[44] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks," *Neural Networks*, vol. 111, pp. 47 – 63, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608018303332

[45] S. Yu and J. C. Príncipe, "Understanding autoencoders with information theoretic concepts," *Neural Networks*, vol. 117, pp. 104 – 123, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608019301352

[46] C. Guo and F. Berkhahn, "Entity embeddings of categorical variables," *CoRR*, vol. abs/1604.06737, 2016. [Online]. Available: http://arxiv.org/abs/1604.06737

[47] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, Jun. 2006. [Online]. Available: http://dx.doi.org/10.1016/j.patrec.2005.10.010

[48] D. Dua and C. Graff, "UCI machine learning repository," 2019. [Online]. Available: http://archive.ics.uci.edu/ml

[49] O. Loyola-González, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, and M. García-Borroto, "Cost-Sensitive Pattern-Based classification for Class Imbalance problems," *IEEE Access*, vol. 7, no. 1, pp. 60 411–60 427, 2019.

[50] K. Guo, D. Liu, Y. Peng, and X. Peng, "Data-driven anomaly detection using ocsvm with boundary optimzation," in *2018 Prognostics and System Health Management Conference (PHM-Chongqing)*, Oct 2018, pp. 244–248.

[51] G. A. Susto, A. Beghi, and S. McLoone, "Anomaly detection through on-line isolation forest: An application to plasma etching," in *2017 28th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, May 2017, pp. 89–94.

[52] J. G. Moreno-Torres, J. A. Saez, and F. Herrera, "Study on the Impact of Partition-Induced Dataset Shift on k-Fold Cross-Validation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 8, pp. 1304–1312, Aug. 2012.

[53] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[54] W. Zhang, T. Du, and J. Wang, "Deep learning over multi-field categorical data," in *Advances in Information Retrieval*, N. Ferro, F. Crestani, M.-F. Moens, J. Mothe, F. Silvestri, G. M. Di Nunzio, C. Hauff, and G. Silvello, Eds. Cham: Springer International Publishing, 2016, pp. 45–57.

[55] T. Kieu, B. Yang, C. Guo, and C. S. Jensen, "Outlier detection for time series with recurrent autoencoder ensembles," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 2019, pp. 2725–2732.
[56] J. Huang and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 3, pp. 299–310, Mar. 2005.
[57] N. Japkowicz, "Assessment Metrics for Imbalanced Learning," in *Imbalanced Learning: Foundations, Algorithms, and Applications*, H. He and Y. Ma, Eds. John Wiley & Sons, Inc., 2013, ch. 8, pp. 187–206. [Online]. Available: http://dx.doi.org/10.1002/9781118646106.ch8
[58] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. on Knowl. and Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2008.239
[59] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. Association for Computing Machinery, 2006, p. 233–240.
[60] K. Boyd, K. H. Eng, and D. Page, "Area under the precision-recall curve: Point estimates and confidence intervals," in *ECML/PKDD*, 2013.
[61] L. Cañete-Sifuentes, R. Monroy, M. A. Medina-Pérez, O. Loyola-González, and F. Vera Voronisky, "Classification Based on Multivariate Contrast Patterns," *IEEE Access*, vol. 7, no. 1, pp. 55 744–55 762, 2019.
[62] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

**Diana Laura Aguilar** received a BSc degree in Bionics Engineering from the National Polytechnic Institute, México, in 2019, and an MSc degree in Computer Science at Tecnologico de Monterrey, Campus Estado de México in 2021. She has worked as a data engineer and is currently working as an IT project coordinator at a multinational company. She is a machine learning enthusiast and her research focuses on interpretable models.

**Miguel Angel Medina-Pérez** received a Ph.D. in Computer Science from the National Institute of Astrophysics, Optics, and Electronics, Mexico, in 2014. He is currently an Artificial Intelligence Manager at Altair Management Consultants. He has rank 1 in the Mexican Research System. He has published in "Information Fusion," "IEEE Transactions on Affective Computing," "Pattern Recognition," "IEEE Transactions on Information Forensics and Security," "Knowledge-Based Systems," "Information Sciences," "Expert Systems with Applications," etc. He has extensive experience developing software to solve Pattern Recognition problems. A successful example is a fingerprint and palmprint recognition framework, which has more than 1.3 million visits and 135 thousand downloads.

**Octavio Loyola-González** received his B.Eng. in Informatics Engineering in 2010 and his M.Sc. degree in Applied Informatics in 2012, both from University of Ciego de Ávila. After, he received his PhD degree in Computer Science from the National Institute for Astrophysics, Optics and Electronics, Mexico, in 2017. He received the Best Thesis Award "José Negrete" for the Doctoral Thesis Category on Artificial Intelligence sponsored by the Mexican Society for Artificial Intelligence (SMIA). Prizewinner in the XXXI National Contest of Computer Science Thesis (ANIEI). Prizewinner to the best PhD Thesis in the Computer Science Coordination at National Institute of Astrophysics, Optics and Electronics. Currently, he is an Artificial Intelligence Manager at Altair Management Consultants. Also, he is a member of the GIEE-ML (Machine Learning) research group at the Tecnologico de Monterrey, and the GIARP Group. These groups are devoted to research on pattern recognition, where we have developed a fingerprint verification framework and a data mining framework. Currently, he is a Member of the Mexican Researchers System (Rank 1).

**Kim-Kwang Raymond Choo** (Senior Member, IEEE) received the Ph.D. in Information Security in 2006 from Queensland University of Technology, Australia, in 2006. He currently holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio. He is the founding co-Editor-in-Chief of ACM Distributed Ledger Technologies: Research & Practice, founding Chair of IEEE TEMS Technical Committee on Blockchain and Distributed Ledger Technologies, an ACM Distinguished Speaker and IEEE Computer Society Distinguished Visitor (2021 - 2023), and a Web of Science's Highly Cited Researcher (Computer Science – 2021, Cross-Field – 2020). He is the recipient of the 2019 IEEE Technical Committee on Scalable Computing Award for Excellence in Scalable Computing (Middle Career Researcher), the British Computer Society's 2019 Wilkes Award Runner-up, the Fulbright Scholarship in 2009, the 2008 Australia Day Achievement Medallion, and the British Computer Society's Wilkes Award in 2008. He has also received best paper awards from IEEE Systems Journal in 2021, IEEE Computer Society's Bio-Inspired Computing Special Technical Committee Outstanding Paper Award for 2021, 2021 IEEE Conference on Dependable and Secure Computing, IEEE Consumer Electronics Magazine for 2020, Journal of Network and Computer Applications for 2020, EURASIP Journal on Wireless Communications and Networking in 2019, IEEE TrustCom 2018, and ESORICS 2015; the IEEE Blockchain 2019 Outstanding Paper Award; and Best Student Paper Awards from Inscrypt 2019 and ACISP 2005. He has received the Outstanding Editor Award for 2021 from Future Generation Computer Systems.

**Edoardo Bucheli-Susarrey** was born in Saltillo, Mexico, in 1992. He received a B.S. in sound engineering from Tecnologico de Monterrey, Campus Santa Fe, in Mexico City in 2015, and an MSc in Computer Science from Tecnologico de Monterrey Campus Estado de Mexico in the State of Mexico in 2019. He has worked as a professional musician and sound engineer and as a teacher in Mathematics and Information Technologies. His research interests lie at the intersection of Signal Processing, Feature Engineering, and Machine Learning. His recently finished Master's thesis is related to feature extraction techniques for audio classification using Fourier and Deep Learning Methods.