Density-Approximating Neural Network Models for Anomaly Detection

Martin Flusser
Faculty of Nuclear Sciences and Physical Engineering
Czech Technical Univ., Prague
flussmar@fjfi.cvut.cz

Tomáš Pevný[*]
Faculty of Electrical Engineering
Czech Technical Univ., Prague
pevnytom@fel.cvut.cz

Petr Somol[†]
Cisco Systems
Cognitive Research
Charles Square, Prague
psomol@cisco.com

ABSTRACT

We propose an alternative use of neural models in anomaly detection. Traditionally, in anomaly detection context the common use of neural models is in form of auto-encoders. Through the use of auto-encoders the true anomality is proxied by reconstruction error. Auto-encoders often perform well but do not guarantee to perform as expected in all cases. A popular more direct way of modeling anomality distribution is through k-Nearest Neighbor models. Although kNN can perform better than auto-encoders in some cases, their applicability can be seriously impaired by their space and time complexity especially with high-dimensional large-scale data. The alternative we propose is to model the distribution imposed by kNN using neural networks. We show that such neural models are capable of achieving comparable accuracy to kNN while reducing computational complexity by orders of magnitude. The de-noising effect of a neural model with limited number of neurons and layers is shown to lead to accuracy improvements in some cases. We evaluate the proposed idea against standard kNN and auto-encoders on a large set of benchmark data and show that in majority of cases it is possible to improve on accuracy or computational cost.

CCS Concepts

•Theory of computation → Nearest neighbor algorithms; •Computing methodologies → Anomaly detection; •Computer systems organization → Neural networks;

Keywords

anomaly detection; neural network; nearest neighbor

[*]Tomáš Pevný is also the Technical Leader at Cisco Systems, Prague

[†]Petr Somol is also with SALOME laboratory at Institute of Information Theory, Czech Academy of Sciences, Prague

ACM SIGKDD 2018 Workshop August 20, 2018, London, UK
© 2018 ACM. ISBN xxx-xxxx-xx-xxx/xx/xx...$xx.00
DOI: xx.xxx/xxx_x

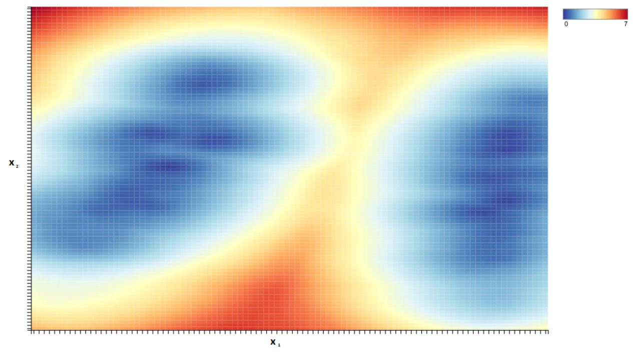

Figure 1: Anomaly scores on a 2D projection of *Iris* data set inferred by distance to k-nearest neigbours. Warmer color depicts higher anomaly.

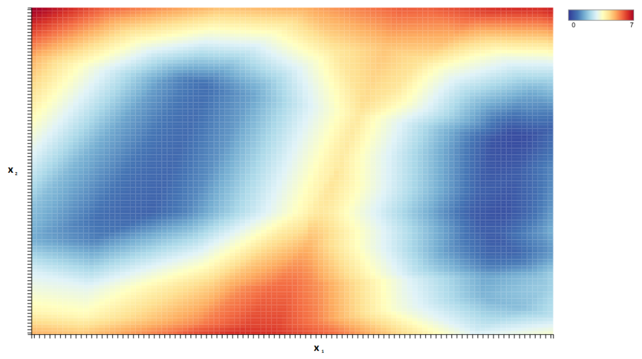

Figure 2: Anomaly scores on a 2D projection of *Iris* data set. Anomalousness inferred by the proposed model. Warmer color depicts higher anomaly.

1. INTRODUCTION

Anomaly detection (AD) is gaining on importance with the massive increase of data we can observe in every domain of human activity. In many applications the goal is to recognize objects or events of classes with unclear definition and missing prior ground truth, while the only assumed certainty is that these entities should be different from what we know well. The problem can thus be seen as the problem of modeling what is common, and then identifying outliers.

Applications of anomaly detection are extensive. Anomaly detection is inherent in cyber security, is successfully applied in industrial quality control, banking and credit card fraud detection, in medicine it can help raise alarms when a patient's condition deteriorates, etc.

Anomaly detection as a general problem has been widely studied as we overview in Section 1.1. Current state of the art is, however, less satisfactory than in supervised learning. Specifically, the recent rapid advances in neural networks (for overview see, e.g., [18], [9], [14]) seem to not have been replicated as successfully in anomaly detection.

The primary neural model use in anomaly detection is through *auto-encoders* (AE). Auto-encoders, however, do not model distribution of anomalies, they optimize a proxy criterion, usually in form of reconstruction error. This fact can limit the success of AEs in some problem areas.

Among traditional anomaly detection principles the *k-nearest neighbor* (kNN) remains among the best performing models. Distance-based detectors directly model the density but can become computationally expensive or even prohibitive in on-line and embedded systems.

In this paper we propose to take use of distance-based kNN principle to enable training of neural models with multiple potential advantages: better robustness against noise as well as low computational complexity leading to high detection speed - an important parameter especially in on-line and embedded anomaly detection applications.

The paper is structured as follows: in Section 1.1 we review existing anomaly detection methodology, in Section 2 we introduce the proposed method, in Section 3 we cover the experimental evaluation of the proposed method and comparison to kNN and AEs on a large body of benchmark data sets, in Sections 4 and 5 we provide discussion and conclusion.

1.1 Anomaly detection

Anomaly detection is also known as one-class classification. The goal is to detect a sample that is somehow different from expected patterns or other observations, without knowing the exact definition of *different.* Hence, anomaly detection techniques focus on modeling what is expected, and subsequently to mark as anomaly anything sufficiently different from the expected.

Contrary to the other machine learning tasks such as classification, the anomaly detection is more difficult because the character of the anomalous data is unknown when the model is trained. In addition to that, the decision how much the sample must be different from others, to be detected as anomalous, is a problem. To solve the anomaly detection problem, we need to address the following concerns: 1) Choice of the model/method with properties suitable for the problem, 2) conceptual problems including thresholding and evaluation.

There is a number of methods for anomaly detection the survey of which is given, e.g., in [7], [20], [23]. This paper focuses on nearest neighbor based techniques [17] and neural models (see Section 1.1.1) and consequently investigates the question of how to find synergy between both. Nearest neighbor techniques are beneficial for their performance (under certain conditions) and adaptability to various data types. Their computational complexity, however, grows rapidly with both the dimensionality and size of the training data. Supporting structures thus have been proposed especially in form of *k-d trees* [3, 12, 4] and *ball trees* [29, 4] to mitigate the problem. But the problem of complexity has thus not disappered as we also illustrate in Section 3.4.

The standard anomaly detection knowledge base also includes *kernel PCA* methods [21], *kernel density estimation (KDE)* including *robust KDE* [16] and *one-class support vector machines (SVM)* [27] that all have been compared to and partly outperformed by neural models, see, e.g., [33]. Neural network models are used for anomaly detection in two different ways: 1) fully unsupervised, i.e., neural network is trained on the regular data only and produces anomaly score or any other similar metric which can be thresholded (see Section 1.1.1), 2) supervised to some extent, i.e., knowledge about possible outliers or other indirect information about anomality apart from the mere density is utilized during training (see, e.g., [6], [24], [26], [13], [34], [22]). In the following we consider only the standard approach to anomaly detector training where no additional information is assumed available apart from the unlabeled data.

1.1.1 Auto-Encoders in Anomaly Detection

Auto-encoders have been used in anomaly detection with success (see, e.g. [1], [28], [8]), yet the application of AEs can lead to disappointing results due to multiple problems: 1) as with any neural model, the open question is the selection of meta-parameters including architecture depth, numbers of neurons, choice of activation function, etc. (addressed, e.g., in [15]), 2) the robustness of neural models in general tends to be questionable unless there is large number of training samples available (addressed successfully in *de-noising auto-encoders*, see, e.g., [31], [25]), 3) the notion of anomality implied by AE may not correspond to the expectation of what an outlier is, as AEs do not model density but a proxy criterion instead, e.g., reconstruction error [see the criterion (1) below], reconstruction probability [2] or minimization of training set energy [33]. The 3rd problem and a proposal of its alternative solution is the primary subject of this paper (see Section 2).

1.1.2 Auto-Encoder Definition

The auto-encoder, also known as replicator neural network or auto-associative neural network is a feed-forward neural network that encodes the input to a compressed form and then decodes back to replicate the input.

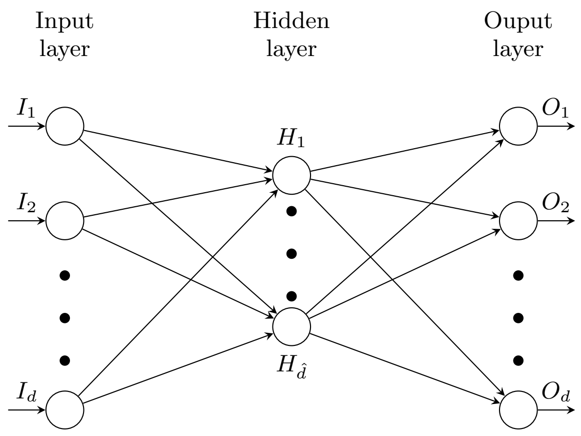

Figure 3: Structure of an auto-encoder that encodes d-dim. vectors into $\hat{d}$-dim., $\hat{d} < d$, and back

The auto-encoder is composed of the encoder and the decoder such that the encoder observes and performs nonlinear dimensionality reduction (from dimension d to $\hat{d}$) with minimal loss of information and similarly the decoder performs a projection from the reduced space back to the original one.

Let us have training set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ such that $\mathbf{x}_i \in \mathbb{R}^d$, $\forall i \in \{1, .., n\}$, where n is number of training samples and d is dimensionality. The input vector $\mathbf{x} \in \mathbb{R}^d$ is encoded to $\mathbf{z} \in \mathbb{R}^{\hat{d}}$ which is projected consequently to $\mathbf{x}' \in \mathbb{R}^d$. The encoding is performed as:

$$\mathbf{z} = f_\theta(\mathbf{x}) = c(\mathbf{W}\mathbf{x} + \mathbf{b})$$

where f is parameterized by $\theta = \{\mathbf{W}, \mathbf{b}\}$, c is an activation function, $\mathbf{W}$ is a $d' \times d$ weight matrix and $\mathbf{b}$ is a bias vector. Similarly the decoding (reconstruction) is performed as:

$$\mathbf{x}' = g_{\theta'}(\mathbf{z}) = c(\mathbf{W}'\mathbf{z} + \mathbf{b}')$$

The parameters of the model are optimized with a training set thus each vector $\mathbf{x_i} \in \mathbf{X}$ can be projected to $\mathbf{z_i}$ and $\mathbf{x_i'}$ such that the average reconstruction error is minimized:

$$\begin{aligned} \theta^*, \theta'^* &= \arg\min_{\theta', \theta} \frac{1}{n} \sum_{i=1}^{n} L\Big(\mathbf{x_i}, \mathbf{x_i'}\Big) = \\ &= \arg\min_{\theta', \theta} \frac{1}{n} \sum_{i=1}^{n} L\Big(\mathbf{x_i}, g_{\theta'}\big(f_\theta(\mathbf{x_i})\big)\Big) \end{aligned} \tag{1}$$

where L represents a loss function which may be defined in many ways, however, the squared error $L(x, x') = ||x - x'||^2$ is the most common [30]. This is also called reconstruction error which is used for the representation of the anomaly score.

2. PROPOSED METHOD

The proposed method aims to make nearest neighbor based anomaly detection efficient utilizing a neural network. The main idea is simple: train a neural network that estimates kNN score. The algorithm does it in two logical steps. First, it creates an auxiliary dataset covering the input space, and for each point of this auxiliary set a kNN anomaly score is computed. Then, this auxiliary dataset is used as a training set to train the neural network-based estimator.

Having the training set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}, \mathbf{x}_i \in \mathbb{R}^d$, $\forall i \in \{1, .., n\}$, let us denote $\mathbf{A}$ the auxiliary data set of m samples where $\mathbf{A} = \{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_m\}$, $\mathbf{a}_i \in \mathbb{R}^d$, $\forall i \in \{1, .., m\}$ and Y the vector of respective anomaly scores, where $Y = \{y_1, y_2, ..., y_m\}$, $y_i \in \mathbb{R}$, $\forall i \in \{1, .., m\}$. We will consider the size of the proposed neural network's hidden layers to be $d \cdot p$ where p is a parameter.

2.1 Computing the auxiliary dataset

The auxiliary set $\mathbf{A}$ is computed from the training set $\mathbf{X}$ as described in the following steps:

1. A bounding hyper-block of $\mathbf{X}$ is observed. Such hyper-block is defined with the vector of lower bounds $\mathbf{h}_l$ and upper bounds $\mathbf{h}_u$ such that $\mathbf{h}_l^{(j)} \leqslant \mathbf{x}_i^{(j)} \leqslant \mathbf{h}_u^{(j)} \quad \forall i \in \{1, .., n\} \quad \forall j \in \{1, .., d\}$ where $\mathbf{x}_i^{(j)}$ represents j-th element of i-th vector from $\mathbf{X}$

2. The hyper-block is filled with randomly generated and uniformly distributed samples $\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_m\}$. By default we consider uniform random sampling. Note that the choice of m for concrete problem may depend on n and d (see also Sec. 3.2.3).

3. The anomaly score vector Y is constructed so that for each auxiliary sample $\mathbf{a}_i, i \in \{1, .., m\}$ the respective $y_i \in Y$ is computed as k-Nearest Neighbor mean distance $G(\cdot)$:

$$y_i = G(\mathbf{a}_i) = \frac{1}{k} \sum_{j=1}^{k} D_j(\mathbf{a}_i) \tag{2}$$

where $D_j(\mathbf{a}_i)$ represents the j-th smallest distance of $\mathbf{a}_i$ to samples from $\mathbf{X}$. Note that the number of neighbors k is a parameter [35, 19].

2.2 Training of the model

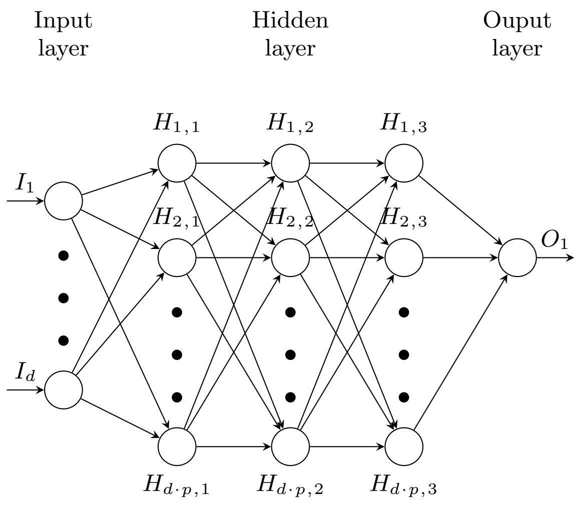

Figure 4: Structure of the utilized network

The feed forward multi-layer neural network (see Fig.4) is trained with $\mathbf{A}$ and Y to be able to predict the anomaly score. In other words, the input vector $\mathbf{a}_i \in \mathbb{R}^d$ is projected to $y_i' \in \mathbb{R}$ as follows:

$$y_i' = f_\theta(\mathbf{a}_i) = f^{(4)}_{\theta^{(4)}}(f^{(3)}_{\theta^{(3)}}(f^{(2)}_{\theta^{(2)}}(f^{(1)}_{\theta^{(1)}}(\mathbf{a}_i)))) \tag{3}$$

where $f^{(j)}_{\theta^{(j)}}$ represents the j-th layer of the NN and the layer propagation is defined as:

$$f^{(j)}_{\theta^{(j)}}(\mathbf{a}_i) = c(\mathbf{W}^{(j)}\mathbf{a_i} + \mathbf{b}^{(j)}) \tag{4}$$

thus ($f^{(j)}$ is parameterized by $\theta^{(j)} = \{\mathbf{W}^{(j)}, \mathbf{b}^{(j)}\}$, c is an activation function, $\mathbf{W^{(j)}}$ is a weight matrix and $\mathbf{b^{(j)}}$ is a bias vector of the j-th layer.

The parameters of the model are optimized with $\mathbf{A}$ and Y such that the average loss function is minimized:

$$\theta^* = \arg\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} L\Big(y_i, y_i'\Big) \tag{5}$$

where L represents a loss function.

3. EXPERIMENTAL EVALUATION

We compare the proposed methodology to standard kNN based anomaly detecion and then to auto-encoder based

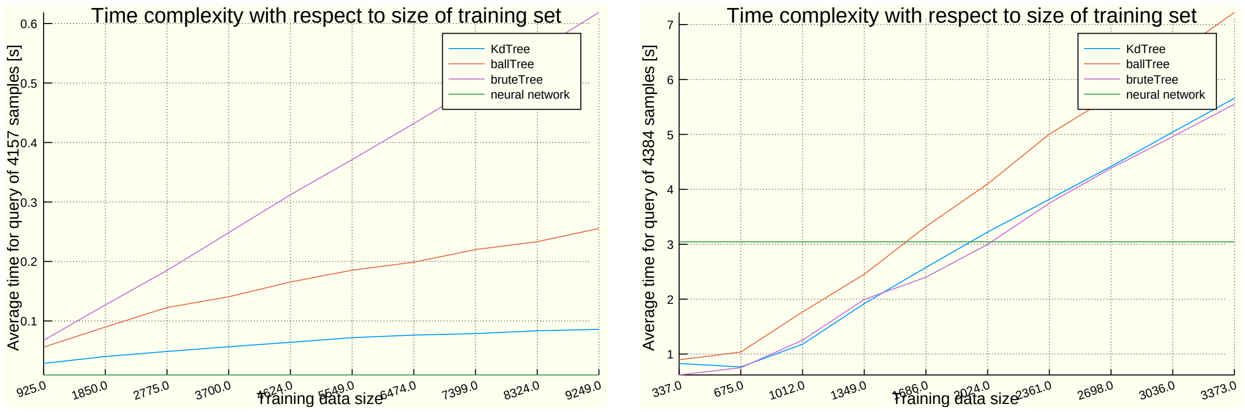

Figure 5: Anomaly detectors' prediction time dependence on training data size in application phase. Tested on *magic telescope* and *Isolet* data sets. Neural model prediction speed does not depend on training data size (note the close-to-zero time in *magic telescope* case)

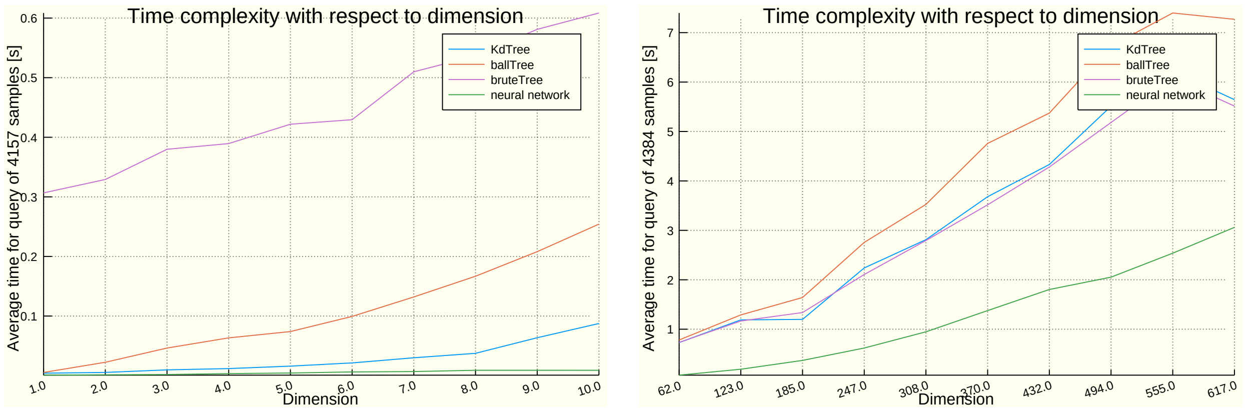

Figure 6: Anomaly detectors' prediction time dependence on dimensionality in application phase. Tested on *magic telescope* and *Isolet* data sets

anomaly detection. We compare area under the curve of receiver operating characteristics (AUC of ROC) [5] of the three approaches on a body of benchmark data. Additionally, we compare computational complexity of the proposed method to kNN with respect to increasing dimensionality and training data size[1].

3.1 Data Sets

Our aim is to compare the methods on a variety of data sets of various properties, ideally such that are widely used in literature.

To do so we have adopted the experimental protocol of Emmott [11], who has introduced a methodology of creating general AD benchmarking sets using multi-class data from the UCI repository. For each source dataset one selected class is used to form the non-anomalous data and some of the others are used as source of anomalies. The concrete choice of anomaly representing classes leads up to four different datasets of four levels of detection' difficulty. Emmott demonstrated the utility of such approach by creating a number of carefully selected sets and using them to evaluate performance of six popular AD method. There is a large number of various multi-class data sets usually based on a real-world data in the UCI repository hence the constructed benchmark sets provide a reasonable reality check. In 2014 Dau considered Emmott's methodology as the most advanced [8].

For our comparison we thus include 64 data sets generated using Emmott's methodology from 18 source data sets representing real-world data. To give more insight into methods' performance under various conditions the data sets are grouped according to their difficulty. We thus perform our evaluation on *easy* (see Table 1), *medium* (Table 2), *hard* (Table 3) and *very hard* (Table 4) problems. Numbers of samples in data sets vary from 66 to 12332, dimensionalities vary from 4 to 90. For details on individual data sets see [11]. See also comments in Section 4.

3.2 Evaluation Setup

To construct training and testing sets, in all cases random sampling is used such that 75 % of normal (non-anomalous) samples is used for training and the rest 25 % for testing. The anomalous samples are only used for testing. The accuracy score is measured by AUC of ROC [5] as is common in

[1]The utilized hardware for this experiment is provided by the supercomputer services of the computing and information center at the Czech Technical University and Cognitive Research at Cisco Systems, Prague.

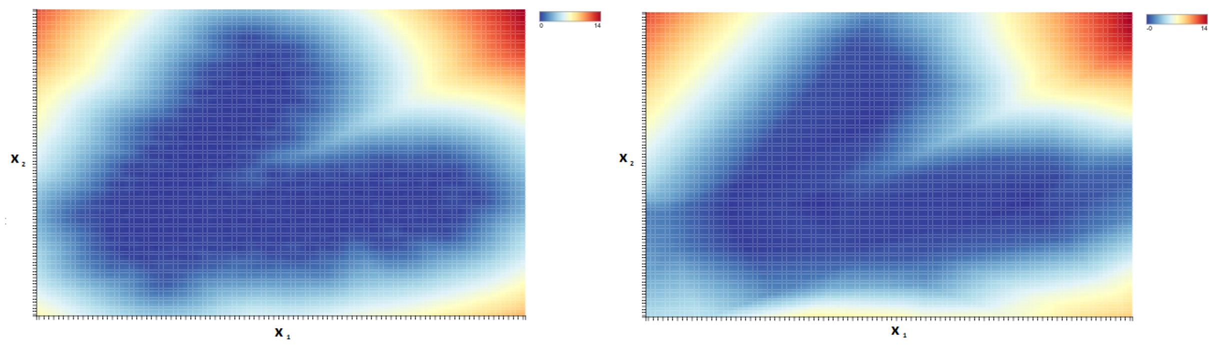

Figure 7: Anomaly scores on a 2D projection of *waveform* data set. Left: anomaly scores obtained by kNN. Right: anomaly scores obtained by the proposed model. Warmer color depicts higher anomaly

literature. The advantage of this metric is the independence on specific thresholding.

3.2.1 *k-Nearest Neighbors Setup*

To evaluate kNN accuracy we compute AUC according to the anomaly score obtained as mean distance $G(\cdot)$ introduced in Equation (2).

To evaluate kNN computational complexity we consider multiple kNN versions: the *basic kNN*, which is implemented as brute tree, *k-d tree* and *ball tree*. The *k-d tree* [3, 12, 4] is a special type of binary search tree that uses hyperplanes to divide the space to accelerate search. Similarly, the *ball-tree* [29, 4] uses hyperspheres to cover the space recursively. The usage of the supporting structure does not affect the precision of the method. However, the time complexity may differ significantly.

The optimal choice of the parameter k which is essential for kNN is not addressed in this paper. However, we observed $k = 5$ as the best performing on average for all the sets thus it is used for all presented experiments. Remark: note that the proposed method is applicable for any k.

3.2.2 *Auto-Encoders Setup*

We evaluate the denoising three-layer auto-encoder according to [8, 30] (see Section 1.1.1). When computing AUC, anomaly score is proxied by reconstruction error. AEs are subject to parametrization; for the tests we needed to decide on the number of neurons per hidden layer, initialization and the type and magnitude of noise. To ensure unified approach across all data sets and to avoid the worst local maxima we opted for a meta-optimization procedure. For each benchmark data set we trained multiple AE models with varying parameters to eventually retain the one with best loss function result.

We observed that *Gaussian noise* worked better than *salt and pepper noise* across the considered data sets. Four different magnitudes of noise are tried with deviations between 0.01 and 0.2 while the samples were scaled to $[0, 1]$ for each dimension.

To set the number of hidden neurons we propose to take guidance from pre-analysis of the data. The number of hidden neurons is selected empirically among six various setups implied by expected relative variance. First, we define a relative variance (0.7, 0.8, 0.9, 0.95, 0.97, 0.98) that we want to preserve with the encoding. Then the required number of hidden neurons is estimated using *principal component analysis* (PCA) [32] as the lowest number of dimensions required to preserve the relative variance.

We observed only negligible improvement from repeated random initialization. All models were trained in 300 000 iterations; varying the number of iterations also proved to have only negligible impact.

Hence the eventual meta-optimization procedure consists from building the 24 models (4 noise parameters, 6 hidden layer size parameters) and retaining the one with best achieved loss function result.

3.2.3 *Proposed Method Setup*

To evaluate the accuracy of the proposed model we compute AUCs using the neural network introduced in Section 2.2. The method is subject to parametrization: its performance can be affected by the properties of auxiliary data set as well as by the standard neural model parametrization (number of layers, number of neurons in layers etc).

We fixed the auxiliary set construction parameters for all experiments as follows. We fixed $k = 5$ in kNN used for auxiliary data set construction to get results comparable to the standalone kNN anomaly detector. The auxiliary data set is constructed as described in Section 2.1 with the total number of auxiliary samples set to $m = n \cdot d^2 \cdot l$, where l is set to 150. The choice of the parameter l is empirical and reflects a trade-off between model accuracy and computational complexity of the training.

ReLU $(f(x) = \max(0, x))$ activation function is used for all neurons (except input). Size of batch is set always to 80. We opted for a simple meta-optimization of neural model parameters so as to avoid the worst local optima. The same procedure is applied across all benchmark data. For this purpose we train for each training data set multiple models, to eventually retain the version with best loss function result. The variation across training runs consist in: 2 or 3 hidden layers, hidden layer size $3d$ or $5d$, multiple random weight initializations, number of iterations thresholded by six values between 1000 and 300000.

3.3 Detection Accuracy Results

Assessing results of three methods over multiple datasets can be done in multiple ways [10]. We focus on pair-wise comparison of the *proposed method* separately to *kNN* and *auto-encoder*.

We summarize the best achived AUC accuracies in four tables, each covering one problem difficulty level: Table 1 for

Table 1: AUC scores for *easy* problems. Table provides two pairwise comparisons of proposed NN vs kNN and auto-encoder

Set	*d*	NN	kNN	NN	AE
abalone	10	0.972	**0.994**	**0.972**	0.961
blood-transfusion	4	0.955	**0.987**	0.955	**0.991**
breast-cancer-wis.	30	**0.989**	0.969	**0.989**	0.978
breast-tissue	9	**1.000**	0.997	**1.000**	0.996
cardiotocography	27	**0.884**	0.566	**0.884**	0.617
ecoli	7	**0.927**	0.918	**0.927**	0.876
glass	10	**0.816**	0.785	**0.816**	0.803
haberman	3	**0.996**	0.972	**0.996**	0.911
ionosphere	33	0.811	**0.962**	0.811	**0.984**
iris	4	**1.000**	0.936	**1.000**	0.569
libras	90	0.500	**0.768**	0.500	**0.562**
magic-telescope	10	0.901	**0.946**	**0.901**	0.896
page-blocks	10	**0.990**	0.981	**0.990**	0.976
parkinsons	22	**0.895**	0.830	**0.895**	0.862
pendigits	16	0.975	**0.994**	**0.975**	0.945
pima-indians	8	0.890	**0.920**	0.890	**0.892**
sonar	60	0.500	**0.605**	0.500	**0.664**
spect-heart	44	**0.500**	0.277	0.500	**0.502**

Table 2: AUC scores for *medium* problems. Table provides two pairwise comparisons of proposed NN vs kNN and auto-encoder

Set	*d*	NN	kNN	NN	AE
abalone	10	0.871	**0.935**	**0.871**	0.825
blood-transfusion	4	0.800	**0.806**	0.800	**0.902**
breast-cancer-wis.	30	**0.973**	0.918	**0.973**	0.954
breast-tissue	9	**1.000**	0.964	**1.000**	0.958
cardiotocography	27	**0.855**	0.561	**0.855**	0.622
ecoli	7	0.837	**0.843**	**0.837**	0.790
glass	10	**0.711**	0.685	**0.711**	0.703
haberman	3	**0.963**	0.955	**0.963**	0.865
ionosphere	33	0.961	**0.993**	0.961	**0.988**
iris	4	**0.980**	0.924	**0.980**	0.858
libras	90	0.500	**0.628**	0.500	**0.532**
magic-telescope	10	0.868	**0.898**	**0.868**	0.850
page-blocks	10	0.956	**0.983**	**0.956**	0.952
parkinsons	22	**0.638**	0.525	**0.638**	0.551
pendigits	16	0.869	**0.974**	0.869	**0.873**
pima-indians	8	0.787	**0.789**	**0.787**	0.735
sonar	60	0.502	**0.717**	0.502	**0.701**
spect-heart	44	**0.589**	0.236	**0.589**	0.434

easy, Table 2 for *medium*, Table 3 for *hard* and Table 4 for *very hard*. Note that we report results rounded to 4 decimal places. Pair-wise better results are emphasised in bold.

To obtain a global picture we then aggregate results over data sets and use Wilcoxon signed rank test to verify the statistical significance (at 0.05 level) of one method's win over the other.

When compared to kNN, Table 5 shows that the proposed method performed better on notable majority of *very hard* problems. Overall, the proposed method performed better on 36 problems while the kNN on 28. However, the statistical significance is not approved for any of the problem difficulty groups. Nevertheless, we have achieved the primary goal of providing a neural model that achieves comparable or better accuracy when compared to kNN with considerably lower computational complexity in application phase (see also Section 3.4).

When compared to auto-encoders, Table 6 shows that the proposed method outperformed AEs in all difficulty levels. The most notable success occurs for *medium* and *very hard* problem level where the statistical significance is achieved. In summary, the NN performed better for 43 problems while the auto-encoder for 21.

Remark: Note that the Wilcoxon test could not be performed across difficulty groups, because it requires independent results.

3.4 Time and Space Complexity Results

The time complexity of the proposed method in detection phase is its main expected advantage over kNN, including advanced kNN forms that utilize search trees. When measuring the detection time we assume that the neural network is already trained and the kNN search tree (if any) built.

We illustrate the advantage on two data sets: *magic telescope*, (10-dim., 12332 samples) and *Isolet* (617-dim., 4497 samples), both of medium difficulty. Figure 5 compares the detection speed of proposed neural model to various forms of kNN with respect to dependency on training data size. To construct the graph we run a series of tests on the same data

Table 3: AUC scores for *hard* problems. Table provides two pairwise comparisons of proposed NN vs kNN and auto-encoder

Set	*d*	NN	kNN	NN	AE
abalone	10	0.520	**0.550**	0.520	**0.529**
blood-transfusion	4	**0.716**	0.521	**0.716**	0.714
breast-cancer-wis.	30	**0.735**	0.627	0.735	**0.742**
breast-tissue	9	**0.542**	0.463	0.542	**0.559**
cardiotocography	27	**0.780**	0.372	**0.780**	0.526
ecoli	7	0.702	**0.736**	**0.702**	0.666
glass	10	**0.607**	0.556	0.607	**0.651**
haberman	3	**0.937**	0.922	**0.937**	0.796
ionosphere	33	0.542	**0.863**	0.542	**0.782**
iris	4	**0.880**	0.842	**0.880**	0.574
magic-telescope	10	0.835	**0.850**	**0.835**	0.807
page-blocks	10	0.937	**0.963**	0.937	**0.941**
parkinsons	22	**0.432**	0.365	0.432	**0.515**
pendigits	16	0.886	**0.977**	**0.886**	0.878
pima-indians	8	0.606	**0.647**	**0.606**	0.606
spect-heart	44	**0.500**	0.085	**0.500**	0.129

Table 4: AUC scores for *very hard* problems. Table provides two pairwise comparisons of proposed NN vs kNN and auto-encoder

Set	*d*	NN	kNN	NN	AE
abalone	10	**0.521**	0.447	**0.521**	0.498
blood-transfusion	4	**0.506**	0.366	**0.506**	0.471
breast-tissue	9	**0.504**	0.413	**0.504**	0.450
cardiotocography	27	**0.790**	0.307	**0.790**	0.437
ecoli	7	**0.567**	0.504	**0.567**	0.566
glass	10	0.321	**0.568**	0.321	**0.529**
haberman	3	**0.555**	0.529	**0.555**	0.498
iris	4	**0.740**	0.632	**0.740**	0.493
magic-telescope	10	**0.667**	0.643	**0.667**	0.590
page-blocks	10	0.854	**0.874**	0.854	**0.872**
pendigits	16	0.857	**0.924**	**0.857**	0.817
pima-indians	8	**0.470**	0.428	0.470	**0.471**

Table 5: Counts of wins of the *proposed method* versus *kNN*, grouped by problem difficulty. Wilcoxon signed rank test at 0.05 level is used to verify statistical significance of wins

	Easy	Medium	Hard	V. Hard	Sum
Proposed NN	10	8	9	9	36
kNN	8	10	7	3	28
Significance	no	no	no	no	–

Table 6: Counts of wins of the *proposed method* versus *auto-encoder*, grouped by problem difficulty. Wilcoxon signed rank test at 0.05 level is used to verify statistical significance of wins

	Easy	Medium	Hard	V. Hard	Sum
Proposed NN	12	13	9	9	43
Auto-encoder	6	5	7	3	21
Significance	no	yes	no	yes	–

set while gradually removing samples. Figure 6 compares the same anomaly detectors with respect to dependency on data dimensionality. To construct the graph we run a series of tests on the same data set while gradually (randomly) removing features.

Note that the neural model has constant complexity with respect to the number of training samples. This makes it potentially very useful whenever a kNN would perform well only with large sample sets. Figure 6 illustrates that even with respect to growing dimensionality the neural model can be expected to keep the edge over kNN models.

We do not provide space complexity measurements but it is clear that larger dimensionality increases space complexity for all models. Neural models can be considered particularly practical to optimize the trade-off between model size and accuracy due to great parametrization variability. Larger data sets severely increase space complexity for kNN but leave neural model size unaffected. This fact can prove crucial, e.g., in embedded applications.

Remark: Note that to measure time complexity of kNN and of the proposed method we set the test environment to use a single CPU core on a computer in controlled environment to minimize unwanted system influences.

4. DISCUSSION

To give more insight into how the proposed model replicates kNN-induced anomaly distribution we provide heat-maps in Figures 1, 2 and 7. To construct the heat-maps the data sets were transformed into 2D space using PCA. The respective anomaly in each pixel position is marked by color on a scale from blue (lowest anomaly) to red (highest anomaly).

Note that due to the extent of performed tests we have not obtained full results for very-high-dimensional data sets from UCI Repository. However, our partial results allow for the following observation: for the proposed neural network it gets more difficult to keep up AUC with kNN with increasing dimensionality. This is not surprising as neural networks are known to require large numbers of samples; with increasing dimensionality this effect gets more pronounced. If training complexity is not the limiting factor, then the effect can be compensated for by increased size of the auxiliary set.

Note also that we have not exhausted all parametrization options of the neural network when comparing the proposed model to kNN. It should be also noted that the main idea behind our proposed method does not actually depend on neural networks. Once an auxiliary set is constructed, it should be possible to apply any predictor capable of learning from samples with labels from $\langle 0, 1\rangle$.

The accuracy of the proposed method thus depends crucially on the number and distribution of auxiliary samples. In the present paper we assume only the simplest definition of the auxiliary set (cf. Section 3.2.3). Expectably the accuracy of the proposed model can be improved further by weighted sampling and optimization of the auxiliary set size with respect to data set properties. This is the subject of our next effort.

5. CONCLUSION

We propose a novel anomaly detection methodology. We take use of distance-based k-Nearest Neighbor principle to enable unsupervised training of neural networks that directly model the density of anomaly values. This is in contrast to auto-encoders where anomality is modeled indirectly through reconstruction error or other proxy criterion.

We compare the proposed method's accuracy and complexity to kNN and to auto-encoders on an extensive set of benchmark data. To obtain robust results we defined meta-optimization of parameters for all compared neural models. The evaluation shows that the proposed model exhibits multiple advantages. When compared to auto-encoders it often provides better accuracy. When compared to kNN it provides comparable accuracy with principally lower computational complexity - an important property especially in on-line and embedded anomaly detection applications.

Acknowledgment: The work has been supported by the grant of Czech grant agency GACR 18-21409S, with partial support from SALOME laboratory and OP VVV Res. Center for Informatics no. CZ.02.1.01/0.0/0.0/16_019/0000765.

6. REFERENCES

[1] E. Aleskerov, B. Freisleben, and B. Rao. Cardwatch: a neural network based database mining system for credit card fraud detection. In *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering*, pages 220–226, Mar 1997.

[2] J. An and S. Cho. Variational autoencoder based anomaly detection using reconstruction probability. Technical report, 2015.

[3] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[4] A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning*, pages 97–104. ACM, 2006.

[5] C. D. Brown and H. T. Davis. Receiver operating characteristics curves and related decision measures: A tutorial. *Chemometrics and Intelligent Laboratory Systems*, 80(1):24–38, 2006.

[6] J. Cannady. Artificial neural networks for misuse detection. In *National Information Systems Security Conference*, pages 368–81, 1998.

[7] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[8] H. A. Dau, V. Ciesielski, and A. Song. Anomaly detection using replicator neural networks trained on examples of one class. In *Asia-Pacific Conference on Simulated Evolution and Learning*, pages 311–322. Springer, 2014.

[9] H. B. Demuth, M. H. Beale, O. De Jess, and M. T. Hagan. *Neural Network Design*. Martin Hagan, 2014.

[10] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, 2006.

[11] A. F. Emmott, S. Das, T. Dietterich, A. Fern, and W.-K. Wong. Systematic construction of anomaly detection benchmarks from real data. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, ODD '13, pages 16–21, New York, NY, USA, 2013. ACM.

[12] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209–226, 1977.

[13] A. K. Ghosh, A. Schwartzbard, and M. Schatz. Learning program behavior profiles for intrusion detection. In *Workshop on Intrusion Detection and Network Monitoring*, volume 51462, pages 1–13, 1999.

[14] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[15] S.-J. Han and S.-B. Cho. Evolutionary neural networks for anomaly detection based on the behavior of a program. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cyber.)*, 36(3):559–570, 2005.

[16] J. Kim and C. D. Scott. Robust kernel density estimation. *Journal of Machine Learning Research*, 13(Sep):2529–2565, 2012.

[17] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: algorithms and applications. *The VLDB Journal*, 8(3):237–253, Feb 2000.

[18] B. Kosko. Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence/book and disk. *Vol. 1Prentice hall*, 1992.

[19] C. R. Loader. Local likelihood density estimation. *Ann. Statist.*, 24(4):1602–1618, 08 1996.

[20] D. Martinus and J. Tax. *One-class classification: Concept-learning in the absence of counterexamples*. PhD thesis, Delft University of Technology, 2001.

[21] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel PCA and de-noising in feature spaces. In *Advances in neural information processing systems*, pages 536–542, 1999.

[22] S. Mukkamala, G. Janoski, and A. Sung. Intrusion detection using neural networks and support vector machines. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 2, pages 1702–1707. IEEE, 2002.

[23] T. Pevný. Loda: Lightweight on-line detector of anomalies. *Machine Learning*, 102(2):275–304, 2016.

[24] J. Ryan, M.-J. Lin, and R. Miikkulainen. Intrusion detection with neural networks. In *Advances in Neural Information Processing Systems*, pages 943–949, 1998.

[25] M. Sakurada and T. Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2Nd Workshop on Machine Learning for Sensory Data Analysis*, MLSDA'14, pages 4:4–4:11, NY, USA, 2014. ACM.

[26] S. T. Sarasamma, Q. A. Zhu, and J. Huff. Hierarchical kohonenen net for anomaly detection in network security. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 35(2):302–312, 2005.

[27] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[28] B. B. Thompson, R. J. Marks, J. J. Choi, M. A. El-Sharkawi, M.-Y. Huang, and C. Bunje. Implicit learning in autoencoder novelty assessment. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 3, pages 2878–2883. IEEE, 2002.

[29] J. K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information processing letters*, 40(4):175–179, 1991.

[30] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[31] 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. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[32] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[33] S. Zhai, Y. Cheng, W. Lu, and Z. Zhang. Deep structured energy based models for anomaly detection. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1100–1109. JMLR.org, 2016.

[34] Z. Zhang, J. Li, C. Manikopoulos, J. Jorgenson, and J. Ucles. Hide: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification. In *Proc. IEEE Workshop on Information Assurance and Security*, pages 85–90, 2001.

[35] M. Zhao and V. Saligrama. Anomaly detection with score functions based on nearest neighbor graphs. In *Advances in neural information processing systems*, pages 2250–2258, 2009.