MACHINE LEARNING FOR ANOMALY DETECTION IN PHYSICS

MAKS KONCILJA

Fakulteta za matematiko in fiziko
Univerza v Ljubljani

Anomaly detection is a broad field with diverse applications in physics, varying from medical diagnostics, detection of seismographic activity to exploring new physics in high-energy particle collider data and observation of gravitational waves, among others. A short review of anomaly detection is provided. The article begins by examining various factors of anomaly detection, including data types, categories of anomalies, label availability, and potential outputs of anomaly detection algorithms. The core of the article is dedicated to convolutional autoencoders, where their application in detecting gravitational waves is demonstrated.

DETEKCIJA ANOMALIJ V FIZIKI Z UPORABO STROJNEGA UČENJA

Detekcija anomalij je široko področje s številnimi implikacijami na področju fizike, te segajo od diagnostike v medicini, zaznavanja potresne aktivnosti, do odkrivanja nove fizike visokih energij v podatkih velikega hadronskega trkalnika in opazovanja gravitacijskih valov. V članku je zapisan kratek pregled problema detekcije anomalij. Članek se začne z obravnavo različnih dejavnikov detekcije anomalij, vključno s podatkovnimi tipi, kategorijami anomalij, razpoložljivostjo oznak in vrstami rezultatov algoritmov za detekcijo anomalij. Ena od številnih metod, s široko aplikativnostjo, so konvolucijski avtoenkoderji, katerih uporaba je predstavljena na primeru detekcije gravitacijskih valov.

1. Introduction

Anomaly detection involves identifying data points that are outliers of a given statistical distribution. In other words anomaly detection can be defined as a problem of finding data patterns that that do not behave as expected. Outliers and anomalies are the two most commonly used terms in the context of anomaly detection in articles and textbooks. Most of the application domains of anomaly detection have specialised methods for detecting anomalies that are tailored to their specific needs while more universal algorithms are also being developed. The beginning of the research on this topic dates back to the era before modern computers when most of the work was done by statistics community [1]. Since then, technological advances have enabled the processing of large amounts of data with shorter computational times. Computer scientists have become important contributors to the development of this research field. Among the interesting examples of anomaly detection are:

- Medical diagnosis from the unusual data patterns in positron emission tomography, magnetic resonance imaging, electrocardiogram signals and other medical devices [2].
- Measurements of the seismographic activity for critical infrastructure. One of the newest projects using state-of-the art algorithms is South Korean high-speed railway system, which measures seismographic activity with the onboard sensors, reducing the cost of seismic network. When an earthquake with a magnitude above a certain threshold is measured, a signal is transmitted to nearby trains to initiate braking to prevent derailment [3].
- The work of the LIGO-Virgo Collaboration is centered around observing gravitational waves. However, the collected data is noise dominated, which is caused by seismic activity, environmental factors, thermal fluctuations of the detector, and the noise of the laser beam. For this reason, deep learning anomaly detection models are being developed to distinguish the gravitational wave signal from the dominant noise signal [4].

- Machine learning for anomaly detection in high-energy particle collider data is employed to search for rare events that would enhance our understanding of physics beyond the Standard Model. Anomalies typically occur in the tails of the distribution and are crucial for the understanding of interactions between particles. On the other hand, a high concentration of localized points that deviates from the expected distribution may hint at the presence of new possible particles or processes [5].

- One of the newest methods of anomaly detection is the so-called Quantum anomaly detection, which promises a reduction in computational time by many orders of magnitude. Working commercialized quantum computers do not yet exist, however quantum units as a part of classical computers are in research stage. Such machines have the potential to surpass classical computers in specific tasks, such as anomaly detection [6].

Anomalies are described as outliers with respect to the expected distribution. For better comprehension, let's showcase an example using the image (1) of a two-dimensional data set with anomalous and regular data points. The two most concentrated regions of points are labeled as N_1 and N_2. The points that are far from these two normal regions are considered anomalous; in this case point anomalies are labeled as A_1, A_2, and a region of anomalies as A_3. From this example, the task

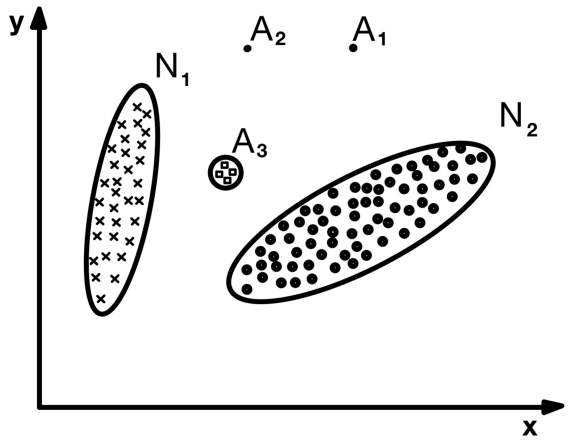

Figure 1. Two-dimensional dataset, with two normal regions labeled as N_1 and N_2, where most of the data points are located. Other points that are far from these two normal regions, such as A_1, A_2, and the collection of points A_3, are anomalies. Adapted from Ref. [7].

of anomaly detection doesn't seem too difficult. It involves identifying region of data points that represent predicted behaviour and labeling the remaining points as anomalies. However, there are many obstacles that can complicate anomaly detection, such as determining the boundary between the normal region and anomalous data points. Another common problem is scarce availability of labeled data for model training and validation, since hiring specialised personnel for hand-labeling data can be expensive and time-consuming. Furthermore, the precise definition of what should be perceived as an anomaly is domain-specific, making it challenging to generalize anomaly detection techniques across all application domains.

2. Aspects of an Anomaly Detection

Few different factors determine anomaly detection problem [8]: type of input data, availability of labels and domain-specific requirements. The data set is a collection of data instances that are represented as vectors, vector components are called features. Features can be binary, categorical or continuous. Each data instance can either be univariate, consisting of only one feature, or

multivariate when it has many features. Multivariate data instance can have all features of the same type or a combination of different feature types. Types of features of the dataset determine which anomaly detection methods can be used.

The choice of the algorithm for anomaly detection also depends on the category of the anomaly. Anomaly categories are

- *Point anomalies* refer to individual data instances that are outliers of a given statistical distribution. The points A_1 and A_2 in the previously described example, shown in figure (1), are examples of point anomalies.

- *Collective anomalies* occur when a group of related data instances forms a collective anomaly. While individual data instances, that are a part of a collective anomaly, may not be identified as anomalies on their own, the collection of related data instances as a whole is considered anomalous. A common example is electrocardiogram data, as shown in Fig. (2), where the red sequence on the left graph represents a collective anomaly.

- *Contextual anomalies* occur when a data instance is considered anomalous based on context, which depends on both the data structure and the specific problem at hand. Data instances are described with two sets of features: contextual features (*e.g.*, time in the case of time series data) and behavioral features (*e.g.*, signal amplitude at a given time in the case of time series data). An example is shown in Fig. (2), where the black dot on the right graph represents a contextual anomaly. It is important to emphasize that a data instance with certain behavioral features might be considered regular or anomalous depending on contextual features.

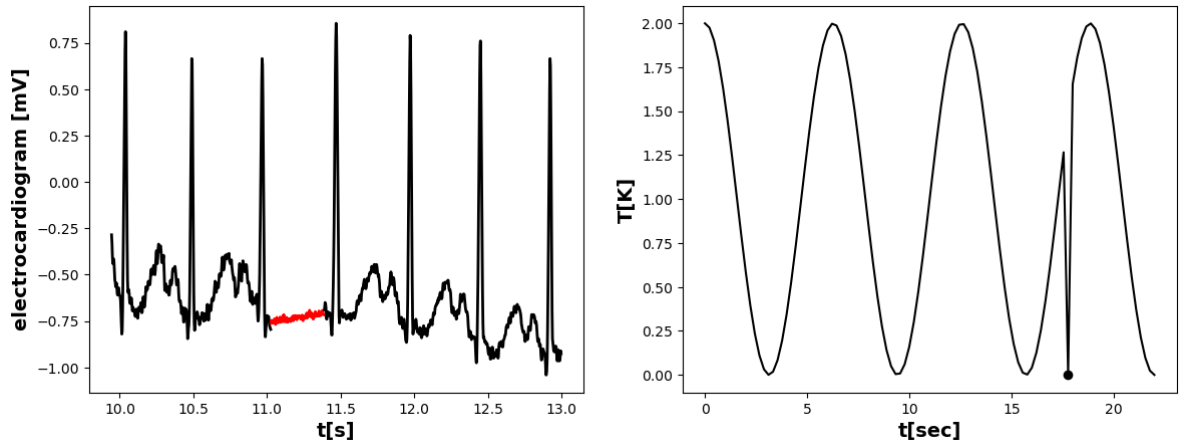

Figure 2. The graph on the left depicts the electrocardiogram signal over time, with a highlighted collective anomaly in red. The graph on the right illustrates the temperature dependence of time, with a back dot marking a contextual anomaly. Adapted from Ref. [7].

Any dataset can have point anomalies while collective anomalies can occur only in data with related instances (*e.g.*, sequence data, spatial data, and graph data). Both point and collective anomalies can be transformed to contextual anomalies by adding contextual features to the data set.

Another crucial factor that has a strong influence on the choice of an anomaly detection algorithm is the availability of labels. Each data instance can be labeled to indicate whether it is regular or anomalous. However, accurately labeled data is expensive, as it requires experts to label the data manually. Furthermore, the scarcity of anomalous events presents another challenge, as it is highly improbable to have all possible types of anomalies labeled in a given dataset. Depending on the

availability of labels for dataset, we categorize anomaly detection algorithms into three different categories:

- *Supervised anomaly detection* models are trained using data with labeled instances for both anomalous and regular data. These models are trained with labeled data to predict whether new instances belong to the regular or anomalous class. However, these models suffer from a lack of available datasets. Since anomalous events are rare, many labeled datasets contain artificial anomalies [9]. Most classical prediction models assume a balanced class distribution, meaning an equal proportion of anomalous and regular instances in datasets. Prepossessing data to meet the criteria of balanced class distribution notably reduces the quantity of instances, as the majority of regular instances have to be discarded. Furthermore, this leads to poor prediction accuracy, as the likelihood ratio of anomalies in training and testing data are not the same.

- *Semi-supervised anomaly detection* models are usually trained using data that has labels for a subset of regular instances. This approach does not require classified anomaly instances, which are difficult to obtain or model in a representative manner.

- *Unsupervised anomaly detection* models are the most widely applicable since no labels are needed. These models assume that anomalies are less frequent. Semi-supervised models can be transformed to unsupervised models by training them on unlabeled data sets with proportion of anomalous instances close to zero.

We have discussed the structure of datasets, various categories of anomalies, and different types of anomaly detection models. To complete the overall picture of anomaly detection, let's describe the possible outputs of anomaly detection models:

- *Scores* can be ascribed to each data instance, representing the degree to which that instance is perceived as an anomaly. The decision where to set a cut-off threshold, which distinguishes between regular and anomalous instance scores, depends on finding a balance between false positives and false negatives. This decision depends on the nature of the task at hand. For example, in medical models, it's preferable to minimize false negative classifications. This is because medical professionals can later re-evaluate patients who were incorrectly diagnosed with a disease as healthy.

- *Labels* are binary classes, instance can be anomalous or regular.

3. Convolutional Autoencoders

In previous section all the key aspects of an anomaly detection are covered. Convolutional autoencoders (CNN-AE) are unsupervised, which makes them applicable to a variety of different anomaly detection tasks. This section will introduce the concept of CNN-AE. Additionally, an example of its application in the context of physics will be discussed, specifically demonstrating its use in the detection of gravitational waves [4].

3.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) are artificial neural networks that consist of multiple layers processing information from input to output. CNNs use convolution filters that preform inner multiplication with input data arrays. For example, consider data instances represented as vectors of length n, where each convolutional filter is of lower dimension, typically denoted as $p < n$.

These filters slide over the elements of the data instance, replacing each segment of length p in the original vector with the result of the inner multiplication, reducing the dimensionality of the data. A schematic representation of this process is shown in Fig. (3). CNNs typically consist of multiple layers with various filters. In CNN architectures, it is common to include specialised layers between convolutional layers, with the most common being the max pooling layer, which reduces data dimensionality by retaining the maximal value within small regions. The final layer of a typical CNN is mapped with an activation function to produce a vector. Each vector value represents the probability of the processed instance belonging to a specific class. Instance is assigned to the class with the highest probability.

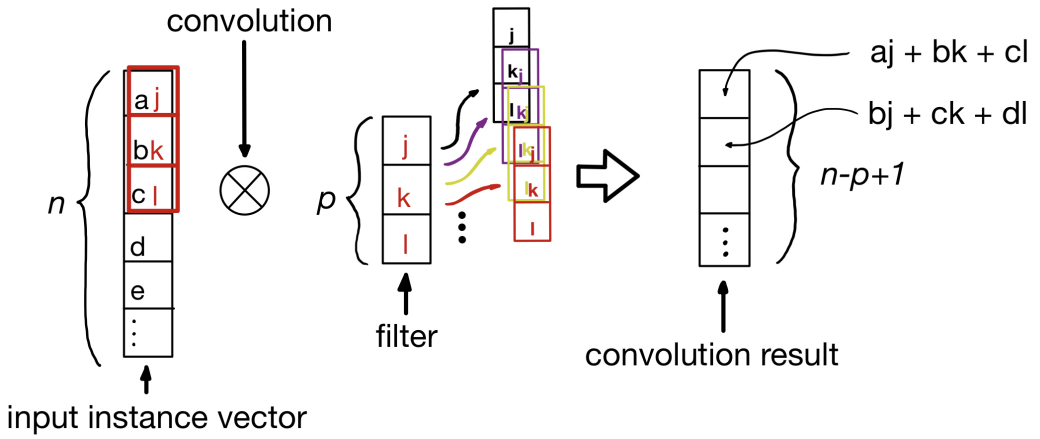

Figure 3. A schematic representation of convolution illustrates the process where the input instance vector, of length n, undergoes inner multiplication with a filter of length p, resulting in a convolution result of length $n-p+1$. Adapted from Ref. [4].

3.2 Autoencoders

An autoencoder (AE) is a deep neural network which is designed to learn the underlying structure of the input data. AE achieves it by compressing the input, encoding it into a latent space — represented by the last layer of the encoder — and than attempting to reconstruct the original data using the decoder, as illustrated in Fig. (4). An ideal AE should accurately reconstruct the input instances without overfitting the training data. The objective of the latent layer is to capture only the information relevant to the variations present in the training data, to ensure optimal reconstruction. The dimensionality of the latent space is almost always smaller than of the input data to prevent overfitting. This ensures that the autoencoder model doesn't directly copy the input data in the output.

Autoencoder is defined by the following components:

- Three sets: the space of input data $\mathcal{Y} \in \mathbb{R}^n$, the space of encoded data (latent space) $\mathcal{H} \in \mathbb{R}^l$, and the space of decoded data $\widetilde{\mathcal{Y}} \in \mathbb{R}^n$. All three sets, $\mathcal{Y}, \mathcal{H}, \widetilde{\mathcal{Y}}$, are typically Euclidean spaces, where $l < n$.

- Two parameterized families of functions, first is the encoder family of functions f_ϕ, parameterized by ϕ, and the second is the decoder family of functions g_θ, parameterized by θ. The encoder family of functions maps from the space of input data to the latent space $f_\phi : \mathcal{Y} \to \mathcal{H}$, and the decoder family of functions maps from the latent space to the space of decoded data: $g_\theta : \mathcal{H} \to \widetilde{\mathcal{Y}}$.

From now on let's consider the simplest AE architecture, as illustrated in Fig. (4), consisting only of the input layer, latent layer, and output layer. The encoder activation function maps the original data from $\mathbb{R}^n$ to the latent space $\mathbf{h} = f(\mathsf{W}\mathbf{y}+\mathbf{b})$, where W is a weight matrix and $\mathbf{b}$ is a bias vector. Similarly, the decoder activation function maps from the latent space to the space of decoded data $\widetilde{\mathbf{y}} = g(\widetilde{\mathsf{W}}\mathbf{h}+\widetilde{\mathbf{b}})$, where $\widetilde{\mathsf{W}}$ is a weight matrix and $\widetilde{\mathbf{b}}$ is a bias vector. Both, weight matrix and bias vector, of encoder and decoder, are randomly initialized and continuously updated during model training. The training of the AE model is proceeded by the minimization of the difference between the original and reconstructed data, commonly achieved trough the mean squared error of the difference [10]:

$$\mathcal{L}(\mathbf{y},\widetilde{\mathbf{y}}) = |\mathbf{y}-\widetilde{\mathbf{y}}|^2 = \left|\mathbf{y} - g\left(\widetilde{\mathsf{W}} f\left[\mathsf{W}\mathbf{y}+\mathbf{b}\right]+\widetilde{\mathrm{b}}\right)\right|^2. \tag{1}$$

The training ends when the loss function $\mathcal{L}(\mathbf{y},\widetilde{\mathbf{y}})$ Eq. (1) plateaus through iterative updates of parameters $\mathsf{W},\mathbf{b},\widetilde{\mathsf{W}}$, and $\widetilde{\mathbf{b}}$. These parameters are usually updated using the adaptive moment estimation algorithm (ADAM) [7]. The model becomes non-linear, by replacing the simple activation functions g and f with more complex CNN layers. This substitution enables CNN-AE to learn more abstract relations between different features.

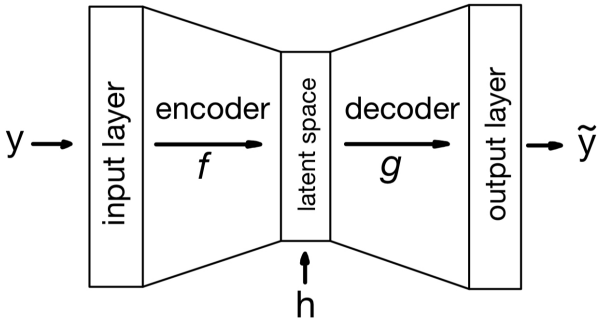

Figure 4. A schematic representation of the simplest autoencoder architecture, comprising of an activation function f mapping the input layer to the latent space, followed by activation function g mapping form the latent space to the output layer. Adapted from Ref. [8].

3.3 Anomaly Detection Implementation of CNN-AE

The architecture of CNN-AE typically consists of convolutional layers with max-pooling layers in the encoder, to reduce the dimensionality. Conversely, the decoder consists of convolutional layers and upsampling layers. These upsampling layers are necessary to increase the dimensionality of the encoded data, effectively repeating values within small regions to gradually upscale the data to the size of the original input data.

In case of anomaly detection, with instances described as vectors, CCN-AE learns two key aspects. A well trained CNN-AE can accurately reconstruct regular instances, resulting in a close resemblance between the original input and the reconstructed output vector. Contrariwise, in case of anomalous instances, the CNN-AE is trained to disregard the anomaly and reconstruct the anomalous input as regular data. Comparison of the anomalous input instance with the CNN-AE reconstructed vector can provide further information about the anomaly. According to the definition, the value of the loss function $\mathcal{L}(\mathbf{y},\widetilde{\mathbf{y}})$, (see Eq. (1)), will be higher in the case of anomalous input, since the reconstructed value $\widetilde{\mathbf{y}}$ will differ more from the original data in comparison to regular inputs.

4. Anomaly detection in Gravitational Waves

After providing a short overview of the anomaly detection task and describing one of the broadly applicable algorithms, Convolutional Autoencoders, let's explore its application in the detection of gravitational waves [4].

Up to now, the LIGO-Virgo Collaborations, have detected over 80 candidate signals of gravitational waves (GW) with their interferometers. These detections put theoretical models of GW sources (*e.g.*, binary systems of neutron stars and black holes) to the test. The reliability of GW observations depends to a large degree on data analysis methods. Gravitational wave data is primarily characterized by noise, which includes sesmic activity, thermal fluctuations of the detector and the noise of the laser beam.

The classical approach to recognising GW signals buried in the detector noise involves scanning the data to match an optimal model of the gravitational waveform. However, this method restricts the GW search to known sources of GW and is computationally expensive. Methods for searching GWs from unknown astrophysical sources are based on models of short-duration GW bursts. However, the reliability of such methods is compromised by the possibility of rapid glitches that may affect multiple detectors simultaneously. To address this issue, a larger number of diverse detectors is required, resulting in vast amounts of data. These circumstances make deep learning methods, such as convolutional autoencoders, particularly promising.

GW interferometers measure time series data that can be processed in real-time with CNN-AE. The authors of the paper [4] proposed an anomaly detection method for analyzing model-independent GW searches. In the case of anomaly detection in gravitational waves data, instances perceived as regular represent the measured noise of the experiment, while anomalies can either be instrumental errors or gravitational waves. In the study [4], GWs were modeled as signals resulting from binary black hole system mergers.

The model's performance was tested on the data with added signals of simulated binary black hole (BBH) mergers. After trying different CNN-AE architectures, the one with the lowest converged value of the loss function $\mathcal{L}(\mathbf{y}, \widetilde{\mathbf{y}})$ had encoding, latent and decoding layers with three convolutional layers in between, each with 256, 128 and 256 neurons respectively. The size of one-dimensional filters was set to three ($p = 3$). The model was trained on two types of datasets: the first consisted of simulated detector signals (artificial LIGO and artificial Virgo datasets), while the second was derived from the realistic second run observations of the LIGO-Virgo collaboration (LIGO Livingston, LIGO Hanford and Virgo datasets). An example of simulated time series data, labeled as artificial LIGO (aLIGO) and artificial Virgo (aVirgo) is presented on the left side of Fig. 5.

Before feeding the two types of datasets into CNN-AE, the data was preprocessed to remove stationary detector noise, resulting in a uniformly distributed amplitude of spectral density. Signals of simulated BBH mergers were additionally added to both datasets, taking into account the spectra of frequencies detectable by LIGO and Virgo spectrometers. The middle graph in Fig. (5) displays three examples of generated binary black hole gravitational wave waveforms that were additionally injected into the two types of datasets.

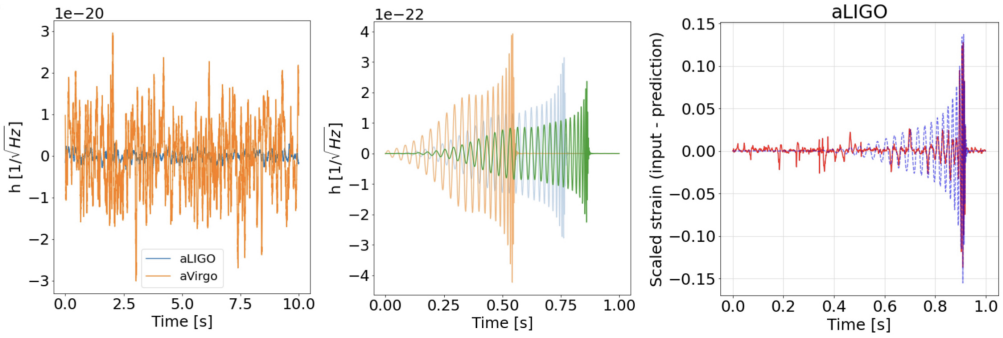

Figure 5. The left graph displays examples of artificial LIGO (aLIGO) and artificial Virgo (aVirgo) time series data. The middle graph illustrates examples of generated binary black hole gravitational wave waveforms. The right graph displays an example of CNN-AE reconstructed signal on artificial LIGO data. The red curve represents the difference between the output and the input of CNN-AE, while the blue curve depicts the injected simulated BBH GW waveform. Adapted from Ref. [4].

When training the CNN-AE, a lower value of the loss function was reached for the artificial Virgo data set. For the same set of injected GW, the artificial Virgo dataset achieved lower values than the artificial LIGO dataset, as Virgo's interferometers are less sensitive. GW waveforms were correctly reconstructed in many more cases for LIGO compared to the Virgo artificial data set, see table (1). While GW waveforms in the artificial LIGO dataset were distinguishable form signal's noise, reconstruction of the artificial Virgo dataset was noise-dominated. The datasets were labeled since anomalies were additionally injected into regular data. The authors of the article [4] set anomaly detection threshold so that 5% of regular data instances were perceived as anomalous by the CNN-AE. Real datasets preformed worse compared to artificial ones, as artificial datasets were constructed based on anticipated improvements in measuring devices, which were not yet implemented in the second observational run of the LIGO-Virgo collaborations. Around half of the GW were detected in the Livingston and Hanford LIGO datasets, while only a third were detected in the real Virgo dataset, as seen on table 1.

The CNN-AE is trained to reconstruct the detectors' noise from the input data despite the presence of GW signals. In the right graph of Fig. (5), we observe an example of the reconstructed GW signal (red line), obtained as the difference between the output and input artificial LIGO data from the trained CNN-AE. For comparison, the expected GW signal is plotted using blue dashed line.

Figure 6 shows the distribution of MSE between the input instances and CNN-AE predictions. The blue histogram represents regular data (detector noise), and the red histogram anomalous data. Setting the anomaly detection threshold to a 5% false positive rate establishes the boundary for CNN-AE predictions.

	artificial LIGO		artificial Virgo		LIGO Livingston		LIGO Hanford		Virgo	
	inj. GW	noise	inj. GW	noise	inj. GW	noise	inj. GW	noise	inj. GW	noise
anom.	96 %	5 %	41 %	5 %	52 %	5%	50 %	5 %	27 %	5 %
reg.	4 %	95 %	59 %	95 %	48 %	95 %	50 %	95 %	73 %	95 %

Table 1. Anomaly detection results for real LIGO and Virgo datasets of CNN-AE. Rows correspond to predictions and columns to the ground truth.

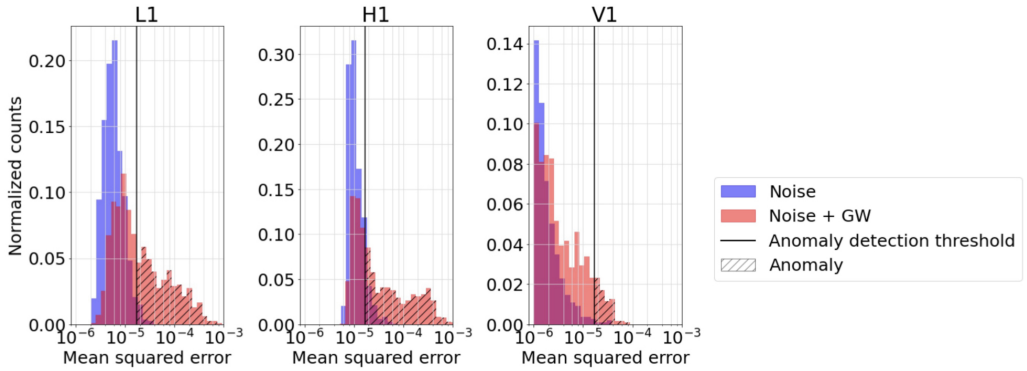

Figure 6. The distribution of $\mathcal{L}(\mathbf{y}, \widetilde{\mathbf{y}})$, where $\widetilde{\mathbf{y}}$ represents the predictions of the CNN-AE for input instances $\mathbf{y}$, is depicted. The distribution of regular instances (detector noise) is shown in blue, the distribution of injected GW is shown in red. The anomaly detection threshold is marked as a vertical line. The first plot corresponds to the LIGO Livingston (L1) dataset, the second to the LIGO Hanford (H1) dataset, and the third to the Virgo (V1) dataset. Adapted from Ref. [4].

5. Conclusion

Anomaly detection involves identifying events that are outliers of a given statistical distribution and finds diverse applications in physics, ranging form medical diagnostics to the observation of gravitational waves. The choice of method depends on the data structure, anomaly types (point, collective, or contextual), and label availability. Anomaly detection algorithms are categorized based on label availability: supervised, semi-supervised and unsupervised models. Each model may provide either a score or a label.

Convolutional autoencoders (CNN-AE) are versatile unsupervised anomaly detection methods with the ability of learning nonlinear relations between input features using convolutional filters. Autoencoders compress input data into a lower-dimensional latent space and than reconstruct it to its original size. The use of CNN-AE was showcased on the example of anomaly detection in gravitational waves of unknown physical background. Presented, rather simple, CNN-AE architecture described in [4] yielded promising results with optimistic future prospects.

REFERENCES

[1] D. M. Hawkins, *Identification of Outliers*, Springer Netherlands, 1980.

[2] T. Fernando, H. Gammulle, S. Denman, S. Sridharan and C. Fookes, *Deep Learning for Medical Anomaly Detection – A Survey*, ACM Computing Surveys **54** (2021), 1–37.

[3] J. Seo, Y. Kim, J. Ha, D. Kwak, M. Ko, and M. Yoo, *Unsupervised anomaly detection for earthquake detection on Korea high-speed trains using autoencoder-based deep learning models*, Scientific Reports **14** (2024).

[4] F. Morawski, M. Bejger, E. Cuoco and L. Petre, *Anomaly detection in gravitational waves data using convolutional autoencoders*, Machine Learning: Science and Technology **2** (2021).

[5] V. Belis, P. Odagiu and T. K. Aarrestad, *Machine learning for anomaly detection in particle physics*, Reviews in Physics **12** (2024).

[6] L.Moro and E. Prati, *Anomaly detection speed-up by quantum restricted Boltzmann machines*, Communications Physics **6** (2023).

[7] V. Chandola, A. Banerjee and V. Kumar, *Anomaly detection: A survey*, ACM Computing Surveys **41** (2009), 1-58.

[8] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, O'Reilly Media, Inc. (2019).

[9] B. Krawczyk, *Learning from imbalanced data: open challenges and future directions*, Progress in Artificial Intelligence **5** (2016), 221-–232.

[10] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016.