УДК 004.01/004.046

ANOMALY DETECTION USING AUTOENCODER FOR DATA QUALITY MONITORING IN CLOUD

C.S.Dzik
graduate student bsuir, utech solutions, software and data engineer

I.I. Piletski
phd, associate professor of belarusian state university of informatics and Radioelectronics

Belarusian State University of Informatics and Radioelectronics, Republic of Belarus
E-mail: kanstantind@gmail.com, ianmenski@gmail.com

C.S.Dzik

Graduate student BSUIR, Utech Solutions, software and data engineer, conduct scientific research of anomaly detection using autoencoder artificial neural network

I.I. Piletski

PhD, Associate Professor of Belarusian State University of Informatics and Radioelectronics. In the field of IT for over 50 years. Participation in the development of several large projects.

Abstract. Many works are dedicated to solving the data quality problem, a number of standards have been developed, but the problem has not been solved for decades. Moreover, this problem now requires a more complex solution due to processing large amounts of unstructured data in the cloud. This work presents the original project Autoencoder that focuses on the technology of analysis, detection and forecasting poor-quality data transmission based on machine learning and the use of neural networks.

Keywords: anomaly detection, autoencoder, artificial neural network, MLP, deep learning, AWS, S3, unsupervised learning

Introduction.

In today's world, cloud computing is a rapidly evolving technology that many organizations are adopting to enable their digital transformation. Cloud technology is opening up new competitive opportunities for companies globally and re-defining how they do business. The cloud makes resources, applications, platforms, and data available anytime, anywhere. Transferring data from outdated systems to the cloud enables you to scale your business, make your data productive, and make it more accessible.

Today most of the company's operations and strategic solutions rely heavily on the cloud data, so data quality is becoming an increasingly important characteristic. Data quality issues that arise when data and data applications are transferred to the cloud have a particular position among the challenges companies face. Cloud computing assumes new types and resources for potential data quality errors. In general, poor quality data can affect productivity, total and overall return on investment. Data quality is significantly affected by data that differs from the data contained in the data set, the so-called anomalous data.

Data quality and anomalous data.

According to data quality experts, data is of high quality when it satisfies the requirements of its intended use. In other words, companies know that they have good quality data when they are able to use it to communicate effectively with their constituents, determine clients' needs, and find effective ways to serve their client base [1-4].

This data quality definition is broad enough to help companies with varying products, markets, and missions to understand if their data is up to standards.

Data quality is not good or bad, high or low. It is a range or an indicator of operability of the data that passes through a company. Data quality management ensures the context-dependent process of improvement of suitability of the data, which is used for analysis and decision-making. The goal is to provide the vision of the "health" of the data by applying different processes and technologies to the increasingly complex data sets [1-4].

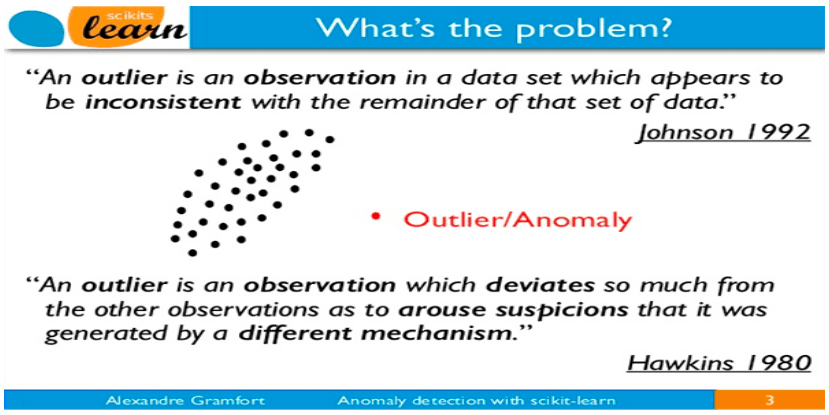
We want to be sure that when we take advantage of the cloud to help data managing, we define data quality parameters at the same time.

The most obvious and compelling way to achieve the goal is to make sure we perform automatic data quality checks for all our data, wherever they are - in the cloud or elsewhere. We must always perform an on-site data quality check.

Virtually every company that works with data has a certain data quality (DQ) monitoring system. Some companies even hire an entire department that deals with the issue. This option is very expensive. In addition, most data quality checks are hard-coded and rule-based. In the event of a failure, the system notifies you of the risk indicator. Such rules are often critical to business continuity. For example, we cannot have a missing customer ID or a "risk profile" variable with an incorrect value. As the amount of data grows, you cannot specify a rule for work with each attribute; not to mention the difficulty of working with hard-coded multidimensional control checks.

The best option is automated DQ (data quality) checks using Machine Learning to detect anomalies that we don't even need to explicitly program.

Identifying anomalies (or outliers) in data is a challenge for scientists and engineers from various fields of science and technology. Although the detection of anomalies (objects suspiciously different from the main data set) has been engaged for, a long time and the first algorithms were developed back in the 60s of the last century figure 1 [5].


What's the problem?

"An **outlier** is an **observation** in a data set which appears to be **inconsistent** with the remainder of that set of data."

Johnson 1992

"An **outlier** is an **observation** which **deviates** so much from the other observations as to **arouse suspicions** that it was generated by a **different mechanism**."

Hawkins 1980

Alexandre Gramfort Anomaly detection with scikit-learn 3

Figure 1. Example of an anomaly in data

In this area, there are many unresolved issues and problems that people face in such areas as consulting, bank scoring, information security, financial transactions and healthcare. So, for example, various algorithms and methods for detecting anomalies are used in the following areas: Fraud Detection, Cyber-Intrusion Detection, Medical Anomaly Detection, Sensor Networks Anomaly Detection, Internet Of Things (IoT) Big-data Anomaly Detection, Log-Anomaly Detection, Video Surveillance, Industrial Damage Detection.

In connection with the rapid development of deep learning algorithms over the past few years, many modern approaches to solving this problem have been proposed for various types of studied data, be it images, tabular data (about financial transactions), etc.

Anomaly detection.

Anomaly detection (or outlier detection) is the identification of rare items, events or observations which raise suspicions by differing significantly from the majority of the data. Typically, abnormal data may be associated with a problem or rare event such as data quality, bank fraud, health problems, structural defects, faulty equipment, etc. This relationship is interesting in terms of the possibility to identify data points that can be considered anomalies, since the detection of such events is interesting from the point of view of sustainable business development [1]. This brings us to one of the key goals of this study: how to determine whether data points are normal or abnormal? In some simple cases, this can be determined at once (see figure 2.).

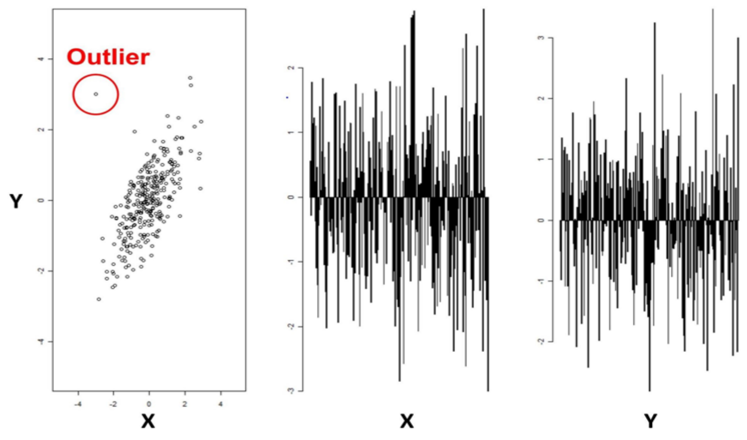
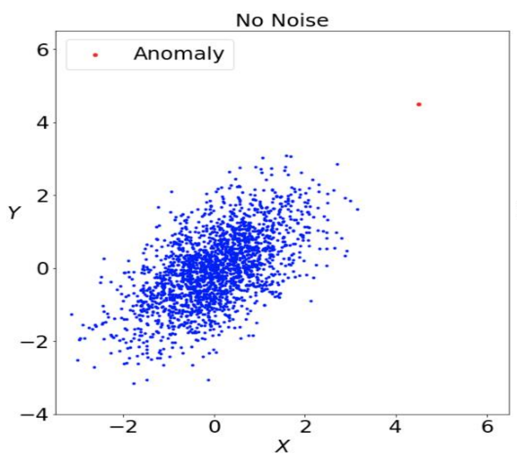


Figure 2: Anomaly detection for two variables

Let us consider the case of two-dimensional data (X and Y): it is quite easy to visually identify anomalies through data points located outside the typical distribution. However, looking at the numbers on the right, it is not possible to identify the outlier directly from investigating one variable at the time.

In addition, it still remains a difficult task to differentiate whereas the difference between anomalies and noise in some sort of data. This is illustrated in figures 3.1 and 3.2:



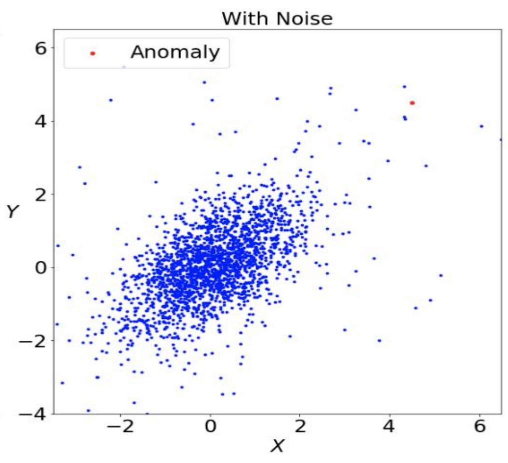


Figure 3.1. Data without Noise *Figure 3.2. Data with noise*

In both figures, the main distribution is the same. In Figure 3.1, the anomalous point marked in red seems to be obvious as it deviates significantly from the rest. However, in Figure 3.2 it is difficult to distinguish the anomaly point from the other points in the sparse space. This example shows that the difficulty to distinguish between anomalies and noises depends on the dataset. Thus, a deep understanding of the dataset is necessary to distinguish between anomalies and noises.

Various methods and algorithms of traditional machine learning are used to detect anomaly research: Supervised, Unsupervised, Hybrid Models, and various Neural Networks. However, at present, the Deep learning for anomaly detection method has found great application.

Deep learning for anomaly detection.

Deep Anomaly Detection (DAD) - allows you to resolve the following set of limitations:

Uncertainty of new data: since anomalies are objects that are not similar to all others, the algorithm must be able to generalize information about the object with the information already available from previously obtained data and determine the lack of similarity, if any;

Heterogeneity of different classes of objects: the dissimilarity can be different;

The rarity of the appearance of anomalies: imbalance of classes in training.

Various types of anomalies: single objects, ordinary objects, groups of objects in abnormal conditions (too dense graph of fake social network accounts).

At the same time, from the point of view of machine learning, the following main difficulties in the task at hand can be named:

Low values of precision metrics in the problem of classification into abnormal / normal (frequent false positives of algorithms on normal data);

The problem of large data dimensions;

Lack or lack of labeled data;

The instability of algorithms to noisy objects;

Detection of anomalies of a whole group of objects;

Low interpretability of results.

Categorization of approaches, in article [6] G. Pang gives the following classification of existing approaches to solving the problem (see figure 4.).

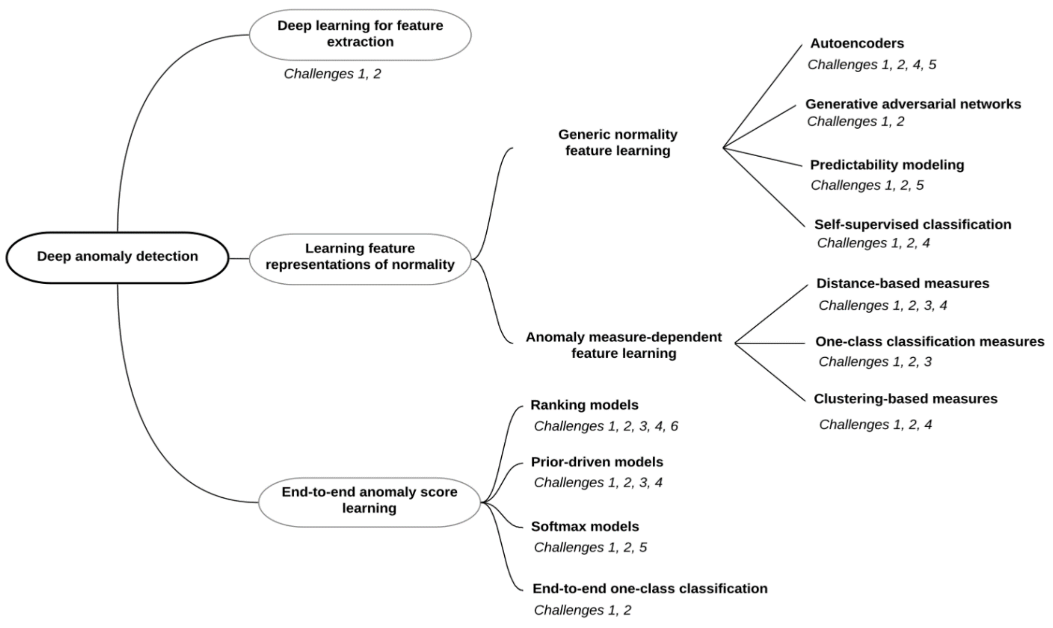


Figure 4. Deep anomaly detection

The author breaks down all algorithms into three large groups:

1. Deep learning for feature extraction;
2. Learning feature representation of normality;
3. End-to-end anomaly score learning.

Deep learning for feature extraction - in fact, the separate problem of obtaining a new domain of features of a smaller size than the original one (in this case, it will be possible to use pre-trained models from other deep learning problems), and the solution of the classification problem already on the data of a new domain using classical anomaly detection methods. Here the two parts of the solution are in no way related to each other, and only the first part can be attributed to DAD. An the figure 5 [6] schematically shows the pipeline for this approach. First, we use a neural network φ (): X → Z to translate the original feature space into a low-dimensional space Z, and then independently scoring the presence of anomalies using classical methods.

Learning feature representation of normality - now the φ (): X → Z neural network is not an independent extractor of new features, but is trained together with the scoring system of anomalies, that is, the Z space will be formed with an eye to the final task.

However, we need end-to-end anomaly score learning - end-to-end pipeline, where the neural network will immediately predict anomaly score, see figure 6.

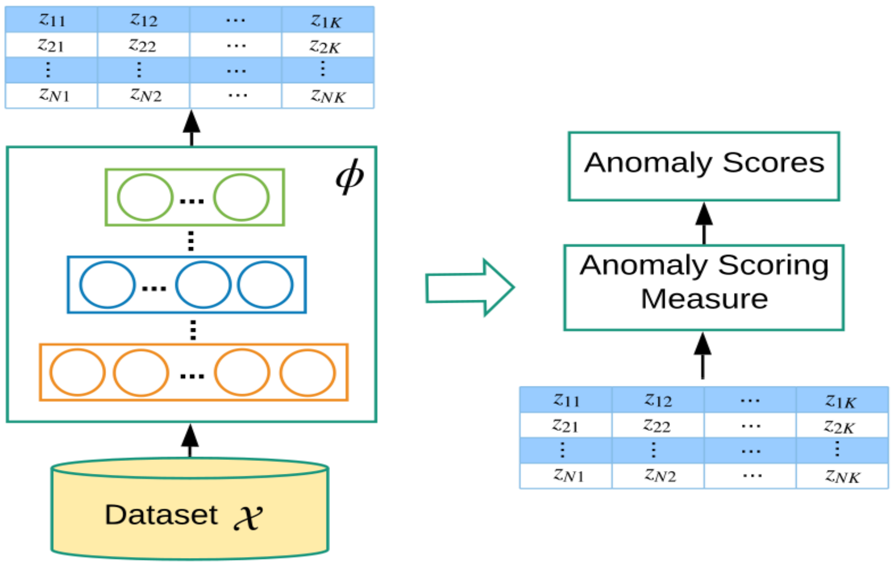

z_{11} z_{12} … z_{1K}
z_{21} z_{22} … z_{2K}
z_{N1} z_{N2} … z_{NK}
ϕ
Dataset $\mathcal{X}$
Anomaly Scores
Anomaly Scoring Measure

Figure 5. Deep learning for feature extraction

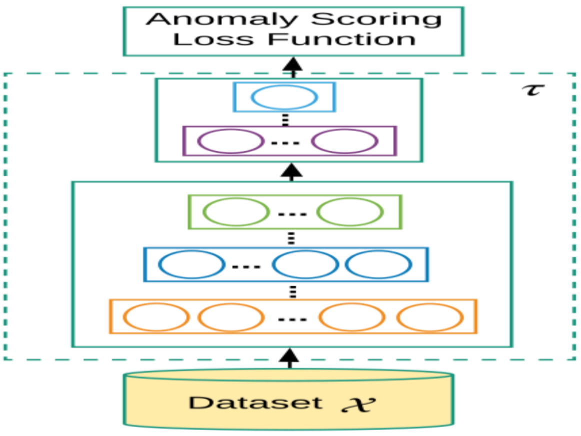


Figure 6. End-to-end anomaly score learning

Autoencoder.

For the DAD task the normal data easily reconstructed by the autoencoder, while the anomalous object for the model will be difficult to reconstruct. An autoencoder is a type of artificial neural network used to learn efficient data codings in an unsupervised manner.

Autoencoder are can: accept an input set of data; internally compress the data into a latent-space (low dimensional) representation; reconstruct the input data from the latent representation.

To accomplish this task, an autoencoder uses two components: an encoder and a decoder.

The encoder accepts the input data and compresses it into the latent-space representation. The decoder then attempts to reconstruct the input data from the latent space [5].

The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for dimensionality reduction. Along with the reduction side, a reconstructing side is learnt, where the autoencoder tries to generate from the reduced encoding a representation as close as possible to its original input [5, 7], see figure 7.

Here, x is the input to the encoder, E is the encoder, the data transformation is z=E(x). x^1 output from decoder, restore data to decoder x^1 =D(z). Then autoencoder can be written as:

$$x^1 = D(E(x)).$$

To find anomalies/outliers using the autoencoder we should: take our pre-trained autoencoder; use it to make predictions (i.e., reconstruct the digits in our dataset), measure the MSE between the original input images and reconstructions, compute quantiles for the MSEs, and use these quantiles to identify outliers and anomalies.

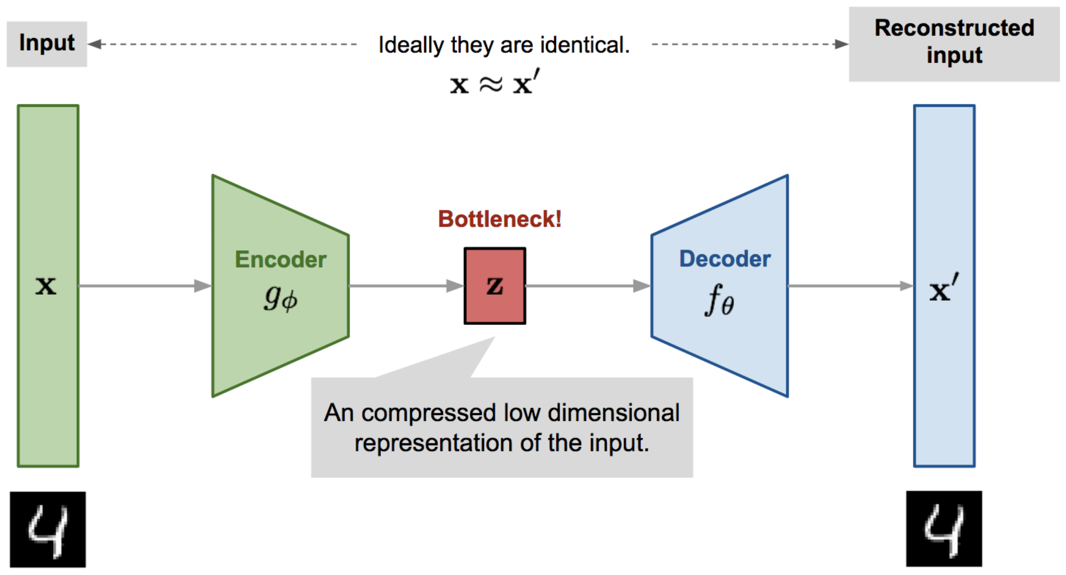

Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
x
Encoder
g_ϕ
Bottleneck!
z
Decoder
f_θ
$\mathbf{x}'$
An compressed low dimensional representation of the input.

Figure 7. Example autoencoder network

The model contains an encoder function g(.) parameterized by ϕ and a decoder function f(.) parameterized by θ. The low-dimensional code learned for input x in the bottleneck layer is z=g_ϕ g_ϕ(x) and the reconstructed input is x′=fθ($g_\phi g_\phi$(x)). The parameters (θ,ϕ) are learned together to output a reconstructed data sample same as the original input, x≈fθ($g_\phi g_\phi$(x)), or in other words, to learn an identity function. There are various metrics to quantify the difference between two vectors, such as cross entropy when the activation function is sigmoid, or as simple as MSE loss:

$$L_{AE}(\theta, \phi) = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - f_{\theta}(g_{\phi}(x^{(i)})))^{2}$$

To find anomalies/outliers using the autoencoder we should: take our pre-trained autoencoder; use it to make predictions (i.e., reconstruct the digits in our dataset), measure the MSE between the original input images and reconstructions, compute quantiles for the MSEs, and use these quantiles to identify outliers and anomalies.

In terms of architecture, the simplest form of an autoencoder is a feedforward, non-recurrent neural network very similar to the many single layer perceptron's which makes a multilayer perceptron (MLP) — having an input layer, an output layer and one or more hidden layers connecting them — but with the output layer having the same number of nodes as the input layer, and with the purpose of reconstructing its own inputs.

Autoencoders are a type neural network architecture that allows unsupervised learning, the basic principle of which is to get the response closest to the input at the output layer.

Autoencoders are relevant for DQ because they can model whole data tables. They map all relevant fields of a data table to the input and the output layer of the network. Hidden layers between input and output layers learn the regular behaviour of the data. As usually, we need to have "good" data to train the model [8].

If something unusual happened to the data, and we didn't expect that. Even if the system conforms to all the classic hard-coded rules, an autoencoder trained to handle regular data will be completely malfunctioning and predicting incorrect data outputs. We will observe a reconstruction error – a significant difference between predicted and actual values – and detect a data anomaly. The data doesn't behave in the way that the autoencoder learned.

Such function is very useful when something new is happening in a dataset. For example, there are suddenly much more missing values (NA) than usual, or levels of categorical variables change or shift. The autoencoder allows you to omit the encoding of these rules. This feature also helps you to identify the source of the problem. The described properties of autocoders allow the detection of anomalies at an early stage of their occurrence and reduce the cost of manual research [9].

Development and application of autoencoder.

We've developed an anomaly detection solution that processes and transforms metadata from S3. Using Deep Learning algorithms, our application detects data load anomalies of S3 files.

For a given S3 path, our application fetches all the metadata, prepares it for ML model consumption, trains the ML model, detects anomalies by identifying unusual patterns, unexpected behaviors, or events.

Application components:

1. Metadata Fetch
2. Data Preparation
3. Anomaly Detection Model

What does application do?

1. Targets data sets on S3

Metadata fetch: Fetch raw S3 metadata and manipulate it to filter out size/timestamp fields for given S3 objects Data Preparation: transform fetched metadata ad extract necessary features required for the model training component.

2. Identifies anomalies in the metadata as files load

Model training: training the ML Models for anomaly detection and forecasting, tune the models and publish as executables.

Model execution: execute the ML models using prepared data and generate labels for each row(anomaly/not anomaly).

Components details:

Metadata Fetch: the driving force behind our application is S3 metadata. The purpose of our application is to be able to detect/forecast anomalies based on load patterns for data files in S3. In order for the framework to be able to understand the data and detect anomalies, the data needs to be made available for the framework utilities. This is where the Metadata Fetch process comes in.

The Metadata Fetch process is the first step in the application pipeline. As its name suggests, it fetches the metadata from objects in S3 and prefroms transformations & formatting on top of the fetch metadata so that it can be consumed further down the pipeline. The key pieces of information that are collected during this fetch process are the object load timestamp.

Metadata Fetch is built entirely in python. It utilizes the S3 API (specifically the list_objects_v2 function) to list the objects for a given S3 Bucket/Key pair and obtain metadata from the API response. A sample API response for the list_objects_v2 API call is shown below:

```
HTTP/1.1 200
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult>
   <IsTruncated>boolean</IsTruncated>
   <Contents>
      <ETag>string</ETag>
      <Key>string</Key>
      <LastModified>timestamp</LastModified>
      <Owner>
         <DisplayName>string</DisplayName>
         <ID>string</ID>
      </Owner>
      <Size>integer</Size>
      <StorageClass>string</StorageClass>
   </Contents>
   ...
   <Name>string</Name>
   <Prefix>string</Prefix>
   <Delimiter>string</Delimiter>
   <MaxKeys>integer</MaxKeys>
   <CommonPrefixes>
      <Prefix>string</Prefix>
   </CommonPrefixes>
   ...
   <EncodingType>string</EncodingType>
   <KeyCount>integer</KeyCount>
   <ContinuationToken>string</ContinuationToken>
   <NextContinuationToken>string</NextContinuationToken>
   <StartAfter>string</StartAfter>
</ListBucketResult>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Name>bucket</Name>
    <Prefix/>
    <KeyCount>205</KeyCount>
    <MaxKeys>1000</MaxKeys>
    <IsTruncated>false</IsTruncated>
    <Contents>
        <Key>my-image.jpg</Key>
        <LastModified>2009-10-12T17:50:30.000Z</LastModified>
        <ETag>"fba9dede5f27731c9771645a39863328"</ETag>
        <Size>434234</Size>
        <StorageClass>STANDARD</StorageClass>
    </Contents>
    <Contents>
       ...
    </Contents>
    ...
</ListBucketResult>
```

From the above response, the LastModified, Size and StorageClass fields are filtered output and pushed to the final output.

Data Preparation component.

Data Preparation takes the input json files (same as the output if metadata fetch process) which includes multiple files where each contains a list of scheduled load time metadata, and output all the same records with additional features extracted from the input attributes in csv/json format.

Sample features:

file_id: 11
file_size: 1235.0
size_percentage_change: -128.2
last_modified_timestamp: 2021-06-27 16:04:22.0
month: 6
day_of_month:27
week_of_month: 5
day_of_week: 4
time_of_day: 964.36
last_modified _cumulative_duration: 7905.45
last_modified_range: 7905.45

Formula for feature extraction:

File_identifier: unique id for each file
File_size: “total size” attribute from the input file
Filesize_percentage_change: based on total_size in each file_identifier,[(current-previous)/current*100]
Timestamp: last_modified_timestamp attribute from the input file
Month,day_of_month,week_of_month,day_of_week:extract from “last_modified_timestamp”
Time_of_day:based on “last_modified_timestamp”,unit: minute, range:(0,1440)
Cumulative_duration: based on “last_modified_timestamp” in each “file_identifier”,[(current-initial)], unit: minute
Time_range: based on “last_modified_timestamp” in each “file_identifier”, [(current-previous)], unit:minute

Anomaly Detection model.

Detection model is build using Autoencoder in Keras with a TensorFlow Backend. This Model allows to detect anomalies by identifying unusual patterns or behaviors and label output with the results.

ML implementation workflow:

Importing the required libraries
Read data
Preprocessing the data
Building and Training the model
Saving and Reload the model
Reconstruction error graph with thresholds labeled
Simulation of the process with full file
Analysis Results
Metrics

Autoencoders require three things:

Encoding function
Decoding function

Loss function describing the amount of information loss between the compressed and decompressed representations of the data examples and the decompressed representation. (i.e. “loss” function)

For our solution, we built an autoencoder with two hidden layers, with the number of units 9-6-3-6-9 and tanh and reLu as activation functions.

The autoencoder was then trained with Adam - an optimized version of backpropagation - on just legitimate transactions, for 75 epochs (see figure 9), against the MSE as a loss function, figure 10.

Activation functions are mathematical equations that determine the output of neural network. The function is attached to each neuron in the network, and determine whether it should be activated(“fired”) or not, based on whether each neuron’s input is relevant for the model’s prediction.

The general idea behind unsupervised anomaly detection approaches is to find an approximate model that can capture the normal behavior of complex systems. The approximate model can then be used to flag anomalies if the deviation of the predicted behaviors of the trained model from the actual observation exceeds some certain threshold. Training on the normal data, the autoencoder is expected to produce higher reconstruction error for the abnormal inputs than the normal ones, which is adopted as a criterion for identifying anomalies (on Figure 10). We have two columns – time range difference and file size difference. Orange line represent the training curve, while blue line represents new input curve. Green dots indicate normal data points, while red dotes indicates potential anomalies. As we can see on Figure 11-13, unsupervised anomaly detection approaches Autoencoder allows us to identify unusual patterns and behaviors.

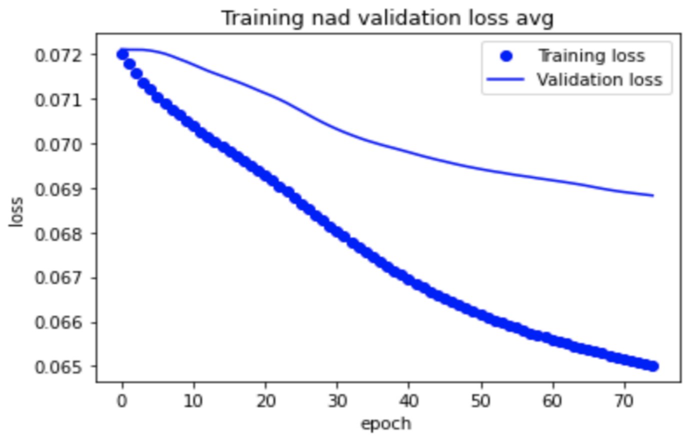


Figure 9. Autoencoder trained

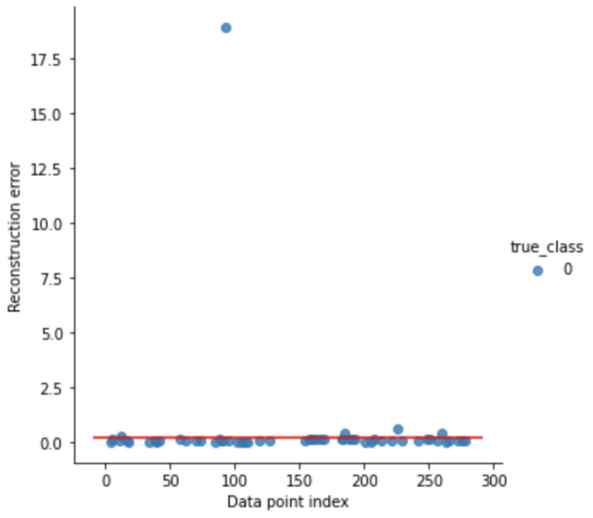


Figure 10. Model allows to detect anomalies

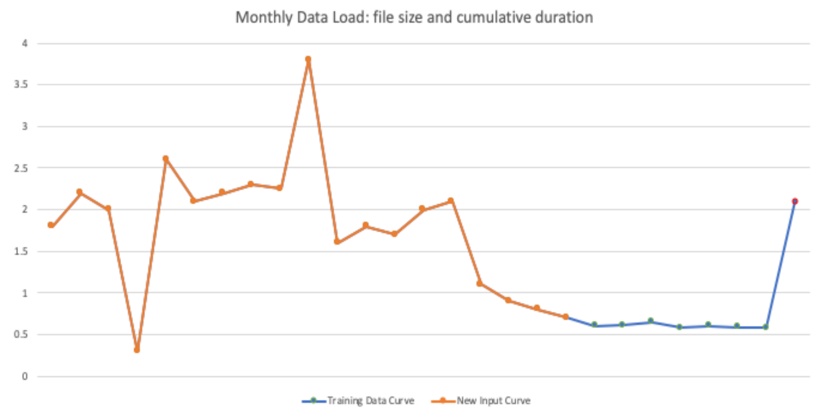


Figure 11. Identify unusual patterns and behaviors: dataset 1

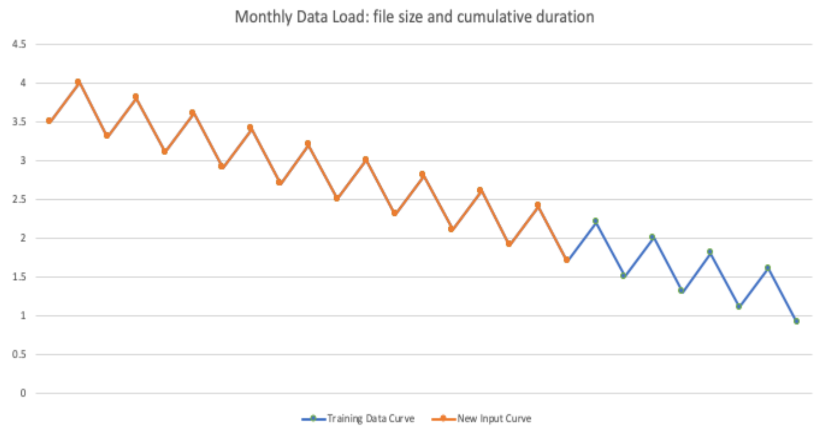


Figure 12. Identify unusual patterns and behaviors: dataset 2

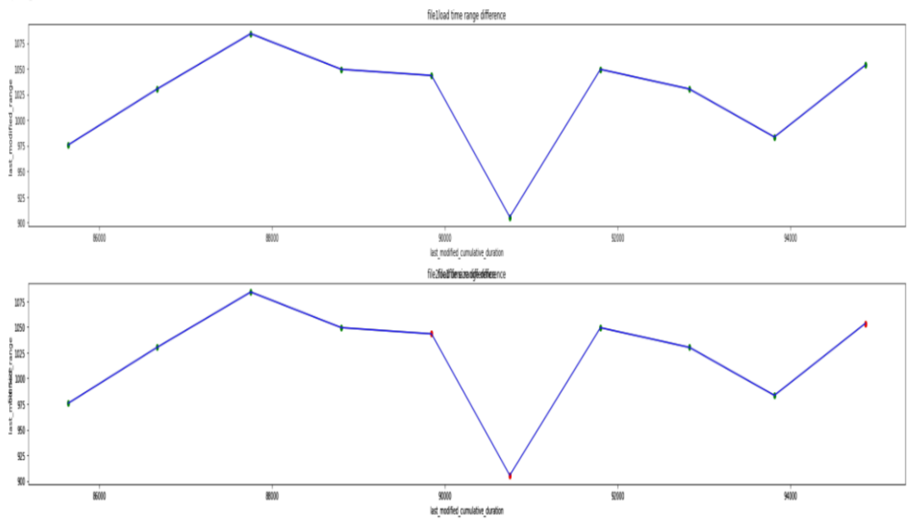


Figure 13. Identify unusual patterns and behaviors: dataset 3

Conclusion.

The main purpose of unsupervised deep learning suggested approach is ability to detect anomalies based on load patterns for data files in S3. By using metadata from Objects in S3 and performs transformations and formatting on top of the metadata so it can be consumed by

Autoencoder Artificial neural network. In the future our plan to apply the proposed technique for various applications. Also the plan to conduct a more in-depth theoretical analysis of the proposed technique.

Benefits of the offered model. Unsupervised learning: no need to know the labels of anomalies before learning. We are using an artificial neural network - Autoencoder algorithm. That allows us to identify unusual patterns or behaviors and label output with the results. Autoencoder algorithm able to take all factors that may cause anomalies into count at the same time and take multiple files together to run at the same time

References

[1] ИСО 8000-2 Качество данных. Часть 2. Словарь (ISO 8000-2, Data quality - Part 2: Vocabulary)

[2] ИСО/TC 8000-110 Качество данных. Часть 110. Основные данные. Обмен данными характеристик. Синтаксис, семантическое кодирование и соответствие спецификации данных (ISO 8000-110, Data quality - Part 110: Master data: Exchange of characteristic data: Syntax, semantic encoding, and conformance to data specification

[3] ИСО/TC 8000-120 Качество данных. Часть 120. Основные данные. Обмен данными характеристик. Происхождение (ISO/TS 8000-120:2009, Data quality - Part 120: Master data: Exchange of characteristic data: Provenance)

[4] Data quality //[Online Resource] – Access Mode: https://en.wikipedia.org/wiki/Data_quality // Access Date: 14.02.2022

[5] Anomaly detection with Keras, TensorFlow, and Deep Learning //[Online Resource] – Access Mode: Anomaly detection with Keras, TensorFlow, and Deep Learning - PyImageSearch // Access Date: 14.02.2022

[6] Deep Learning for Anomaly Detection: A Review //[Online Resource] – Access Mode: https://www.researchgate.net/publication/342732975_Deep_Learning_for_Anomaly_Detection_A_Review // Access Date: 14.02.2022

[7] Giancarlo Zaccone, Md. Rezaul Karim, Ahmed Menshawy, Deep Learning with TensorFlow, 2017

[8] Jason Brownlee, Machine Learning Mastery With Python Understand Your Data, Create Accurate Models and Work Projects End-To-End, 2016.

[9] Chun Chat Tan, AUTOENCODER NEURAL NETWORKS: A Performance Study Based on Image Reconstruction, 2009.

ОБНАРУЖЕНИЕ АНОМАЛИЙ С ИСПОЛЬЗОВАНИЕМ АВТОЭНКОДЕРА ДЛЯ МОНИТОРИНГА КАЧЕСТВА ДАННЫХ В ОБЛАКЕ

К.С. ДИК
Аспирант БГУИР, Ютех Солюшнс, инженер по программному обеспечению и данным, проводит научные исследования по обнаружению аномалий с помощью искусственной нейронной сети автоэнкодера

И.И. ПИЛЕЦКИЙ
Кандидат физико-математических наук. доцент Белорусского государственного университета информатики и радиоэлектроники. В сфере IT более 50 лет. Участие в разработке нескольких десятков крупных проектов.

Белорусский государственный университет информатики и радиоэлектроники, Республика Беларусь
E-mail: kanstantind@gmail.com, ianmenski@gmail.com

Аннотация. Решению проблемы качества данных посвящено множество работ, разработан ряд стандартов, но проблема решается десятилетиями. Более того, данная проблема в настоящее время требует более сложного решения из-за обработки больших объемов неструктурированных данных в облаке. В данной работе представлен оригинальный проект Autoencoder, ориентированный на технологию анализа, обнаружения и прогнозирования некачественной передачи данных на основе машинного обучения и использования нейронных сетей.

Ключевые слова: обнаружение аномалий, автоэнкодер, искусственная нейронная сеть, MLP, глубокое обучение, AWS, S3, неконтролируемое обучение.