Queen's University Belfast
School of Electronics, Electrical Engineering and Computer Science

Minh Nghia Nguyen

BEng (Hons) Final Year Project Report:

Data Analytics for Anomaly Detection in sensitive data access

Supervised by Dr. Vien Ngo

Moderated by Dr. Cheng Long

Partnered by SAP Belfast

April 2018

Abstract

Artificial intelligence and machine learning are the most attractive topics in technology in the recent years, with the goal of fulfilling the ambition of scientist generations in which computers and machines can learn and work like human beings. In this project, machine learning is brought into the security context of cloud services, which has also been emerging rapidly and becoming an important part as a backbone in many industrial areas. In particular, a machine learning system is developed applying state-of-the-art techniques to detect anomalous patterns in the cloud data access and provide rational interpretation of its decision making to the superusers. During the progress of the project, novel approaches and modifications to the existing algorithms are made, especially by combining deep learning with the popular anomaly detection method One-class support vector machine to enhance its capability to operate with large-scale data which is inevitable in cloud services. This also opens up the opportunity to apply the concepts in gradient-based explanation methods that are only feasible for deep learning models, and bring it into the anomaly detection to achieve the desired solution for the interpretability problem.

Project Specifications

DATA ANALYTICS FOR ANOMALY DETECTION IN SENSITIVE DATA ACCESS

- **Supervisor**: Dr. Vien Ngo
- **Moderator**: Dr. Cheng Long
- **Industrial Partner**: SAP Belfast
- **Topics**: Software, Data Analytics, Intelligent Systems
- **Descriptions**:

  As cloud services are now a reality among single users, small and medium-sized enterprises (SMEs) and large enterprises, there is still a certain degree of trust that customers have to put into cloud providers. This is especially true when sensitive data are involved. There is, therefore, a need to enable higher levels of transparency on how data are managed by the cloud providers. Transparency can be achieved by monitoring means, normally through the use of logs. Customers can define the policies regulating all operations over their data in terms of who, what, how and when, and logging can help to identify breaches in those policies.

  Although providing on the spot policy breaching detection is useful for users, a more advanced approach would be the ability to look at a series of events (as events we intend data usage as reported in logs) and correlates those events so that potential risky patterns can be identified and possible future outcome predicted.

  The project should investigate techniques of deep learning to create predictive models able to identify abnormal patterns so that the system can either warn about potential future policy breaches or eventually take ahead actions.

- **Expected Procedure**:

  1. Study of machine learning and deep learning techniques.
  2. Develop models and algorithms for anomaly detection.
  3. Investigate different machine learning frameworks and libraries such as TensorFlow, Keras, Theano, Caffe, Torch, etc.
  4. Implement Python models for training and testing.
  5. Evaluate and compare the performance of the models.

- **Learning Outcomes**:

  At the end of the project the student will be able to:

  1. Gain know-how of machine learning techniques.
  2. Demonstrate advanced knowledge of cutting edge machine learning frameworks and libraries.
  3. Develop real models for sensitive data access events correlations.

Acknowledgements

First and foremost, I want to thank every single member of my family for always giving me all the support and motivation to do the best that I can for almost 23 years of my life. Through all the ups and downs, they are all the reasons in the world that I am who I am today, and for the person that I will be in the future.

I would like to give my utmost thankfulness to my supervisor, Dr. Vien Ngo, who has been giving me guidance in every step of this project, and has as well become my closest companion throughout this pathway to the end of my university journey. His instruction has led me to overcome all of the obstacles that had been holding me up in this project and to achieve fruitful results today.

I am also grateful to SAP Belfast and all of my former colleagues there. My placement year in the most precious name in the industry was the best that I could have asked for and will always be one of the most important part in my career. The time I spent there really gave me the chance to find out what I am capable of, and the skills that I needed for not only fulfilling this project but also living up my dreams in the next chapters of my career. Special thanks to Dr. Carmelo Ragusa from SAP, who has been assisting me and giving me valuable advices during the entire time of this project.

My sincere gratitude also goes to all of the lecturers in Queen's University Belfast and INTO Queen's, who gave me the not only all the knowledge that I certainly will make use the most of in the future but also the inspiration that made me believe in myself and what I have been doing. I also want to give my appreciation to all friends who have been supporting me during my four-year time in Belfast.

Declaration of Originality

I declare that this report is my original work except where stated.

. . . .

. . . .

. . . .

Table of Contents

List of Abbreviations

AD	Anomaly detection
AE	Autoencoder
AE-1SVM	Autoencoder - One-class Support vector machine
API	Application programming interface
CPU	Central processing unit
GCP	Google Cloud Platform
GPU	Graphical processing unit
IP	Internet protocol
JSON	JavaScript Object Notation
LSTM	Long Short-term memory
ML	Machine learning
NN	Neural network
OCSVM	One-class Support vector machine
PR	Precision-Recall
RBF	Radial basis function
ReLU	Rectified linear unit
RFF	Random Fourier features
ROC	Receiver operating characteristic
SGD	Stochastic gradient descent
SVM	Support vector machine
TF	TensorFlow

List of Tables

List of Figures

1 Project overview

The core objective of the project is to develop a machine learning (ML) system to detect anomalous activities in enterprise cloud environments, where the data required for learning are the user access events provided through logging.

Figure 1.1 illustrates the system the project revolves around. A superuser predefines a set of policies, including rules for allowing or raising alerts for certain actions matching users, operations, resources, locations and time. Any time there is an access from the user to the cloud services, a log recording the access event will be generated and fed to the log processor. Then, the log processor manipulates the retrieved logs for further use, while creating alerts matching the non-allowed policies.

From the ML perspective, the knowledge of the policies is not available due to privacy restriction, therefore, the data used for learning will be the set of allowed actions, obtained by filtering out the alerted logs from the entire set of generated logs. The ML system detects the abnormal behaviors and gives feedback to the superuser, including the explanations on why it decided to mark the events as anomalies, so that the superuser might adjust the policies accordingly. The cloud services can be among public cloud services like Google Cloud Platform (GCP), Amazon Web Services, Microsoft Azure, or private cloud services of enterprises.

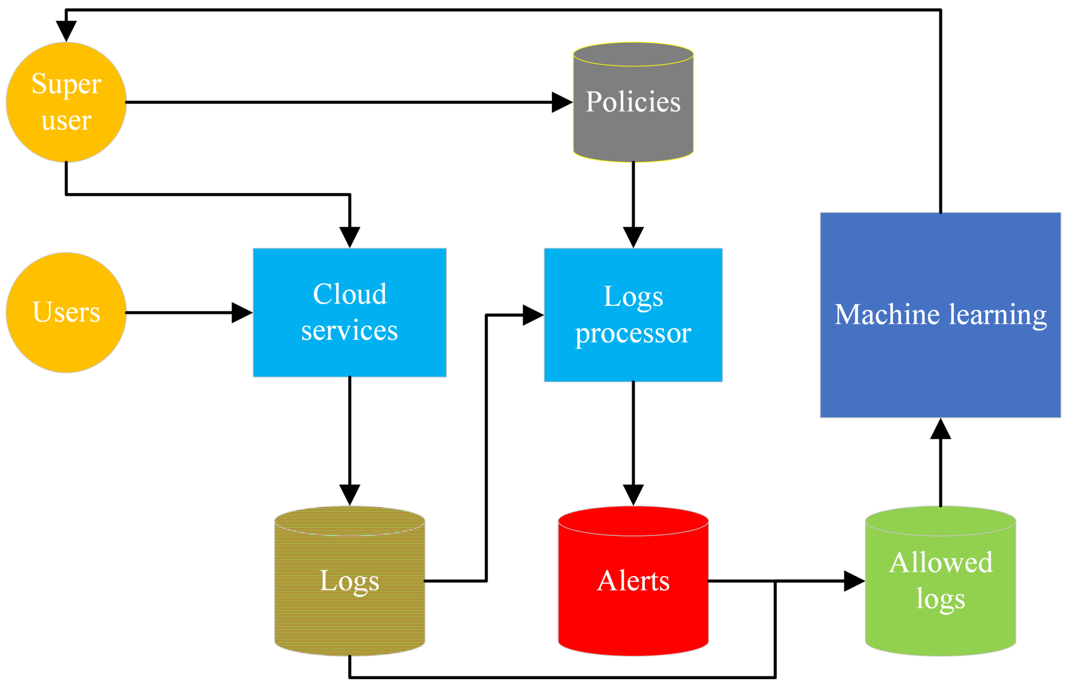


Figure 1.1: An overview of the system alongside with machine learning model. Circles, cylinders, and rectangles represent users, data, and systems, respectively.

Deep learning, which is increasingly used in the recent years in a wide range of areas, especially with the emergence of TensorFlow, has the capability of growing accordingly to the scale of the cloud services, as well as enhancing the integrity and learning speed. Therefore, in this project, it is desirable to investigate TensorFlow framework and apply deep learning techniques to address the anomaly detection (AD) problem.

Based on those facts, the requirements for the project can be summarised as follows:

- Develop ML solution to extract useful information from the structured data of cloud access and detect anomalous patterns in it.
- Develop methods and algorithms to interpret the prediction results from the ML models, in order to give superusers beneficial information to adapt the policies of the system accordingly.
- Apply AD algorithms that operate effectively with large-scale data of at least millions of samples. Utilising deep learning and its ability to exploit Graphical processing units (GPU) power to accelerate training time is preferable.

2 Background

This section presents the background theory for machine learning and related techniques and tools that are applied in the scope of this project to tackle the problems outlined in the project specifications. First, machine learning is introduced, followed by anomaly detection, which is one of its specific application field. Then, an overview of deep learning and neural networks is given. A detailed background of One-class Support Vector Machine (OCSVM), the baseline method that is utilised to detect anomalous patterns, is also described. Finally, gradient-based explanation methods are introduced as the solution to the interpretability requirement of the project.

2.1 Machine learning

ML is a field of Artificial Intelligence where the systems have the ability to learn from data and make predictions or decisions based on the patterns that they have discovered. A commonly used definition of learning is as following: "a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."[1]. Intuitively, a ML model will learn better with more data provided.

The simplest example of ML is a linear regression model where one has a set of value pairs (x, y), and wants the computers to predict the value of any new y given a value of x. In real life, this model is used to forecast weather or market/financial trends. Other applications of ML include classifying objects, clustering data into different groups, or even more complex ones like those involving robotics. Nevertheless, in general, ML can be categorised into three forms:

1. **Supervised learning**: In this form, the training data come with labeled outputs. The models attempt to learn the relationship between the inputs and the outputs to perform either a regression task or a classification task. In regression, the output contains one or more continuous variables that have to be predicted, while in classification, the objective is to assign each input to a finite number of defined classes.

2. **Unsupervised learning**: In this case, target labels are not available. Some subcategories of unsupervised learning are clustering, density estimation, or data visualisation [2]. Clustering discovers the similarities in data and defines a finite number of groups that best describe those characteristics. In density estimation, it determines

the stochastic distribution of the data. Visualisation involves projecting the data from a high dimension to a lower dimension that is easy to visualise like 2D or 3D.

3. **Reinforcement learning**: In this type of ML, there is no data set. Instead, the machines find the optimal actions given any situations to maximise a long-term reward. A reinforcement learning model contains an agent that at any state, perform actions to the environment, receive a reward from the environment and proceed to a new state. Reinforcement learning is widely used in robotics, control, and communications.

2.2 Anomaly detection

AD involves identifying the patterns in the data which are different from the expected behaviour, and samples that have significantly lower frequencies than others [3]. It is widely applied in intrusion detection, fraud detection, and also in medical applications where outlying patterns in medical images can indicate diseases. In the context of this project, AD is the most important aspect, as anomalies in the logs are associated with breaches and abnormal behaviors, and our aim is to detect those data points as accurate as possible.

Based on the characteristics of the training data, there are three AD categories [4] as follows:

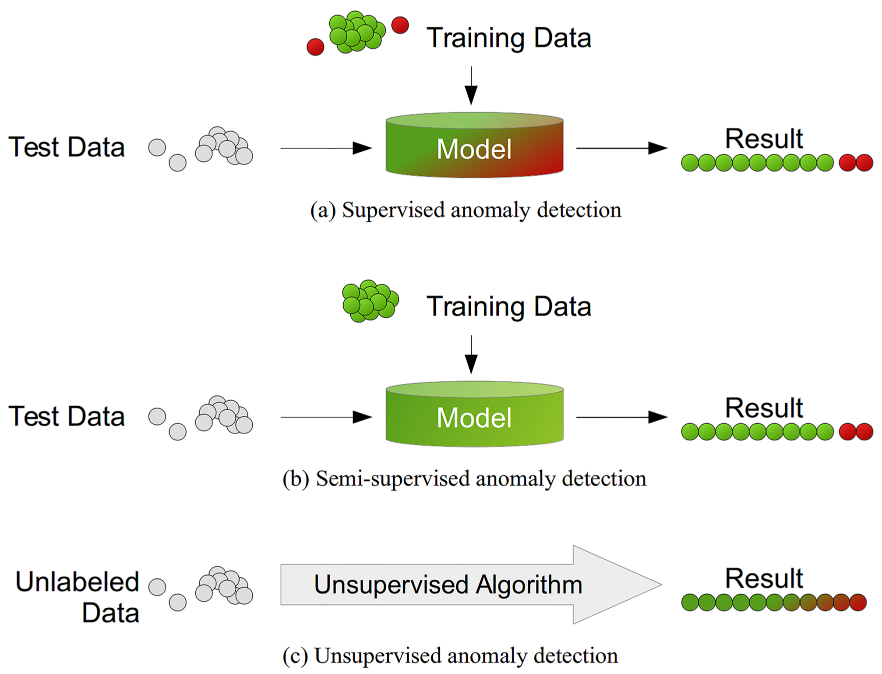


Figure 2.1: [4] The three categories of Anomaly detection. Green samples are nominal data, red samples are anomalies, and gray samples are unlabeled data.

1. **Supervised AD**: In this setup, the training and test data are fully labeled. Thus,

binary classification techniques can be used to classify each sample to either "normal" or "anomaly". Notwithstanding, this is not applicable in a lot of cases where providing expertise knowledge on the data is costly or even completely infeasible.

2. **Semi-supervised AD**: The training data in semi-supervised AD comprise only normal samples. From this, One-class classifier is trained to generalise the region of expected behavior in the feature space. Then, any sample that does not fall into this region is considered anomalous.

3. **Unsupervised AD**: There is no need for any labeling or knowledge of the samples in the data set. The model should be able to explore the abnormal patterns in the set. It then gives any data point a score accordingly to rate how likely it is an anomaly.

Subject to the properties of the extracted data, semi-supervised or unsupervised AD can be applied in this project.

To address AD tasks, several algorithms can be considered. Autoencoder, which is explained later in section 2.3.3, represents a class of AD techniques that attempts to compress the data and reconstruct it to its original form. One of the most recent deviation of Autoencoder tailored for AD is the Robust Deep Autoencoder [5], which decomposes the data into the normal part that is error-free (zero loss) and the outlier part that are hard to reconstruct. On the other hand, clustering-based methods can also be used, exploiting the fact that anomalies belong to less dense clusters or are abnormally far from its cluster's centroid, matching the definition of anomalies mentioned earlier in this section. The main drawback of this method and most of the autoencoder-based methods is that a threshold has to be determined by trial-and-error procedures. Isolation Forest [6] (an ensemble method similar to the popular Random forest in classification) and One-class Support vector machine [7], instead fit a boundary that covers all the nominal data inside and separate them with the anomalies, as illustrated in Figure 2.2.

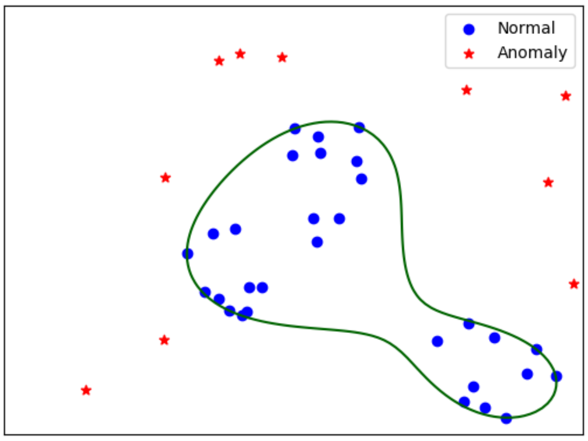


Figure 2.2: Example of a boundary that separates normal and anomaly classes.

2.3 Deep learning

2.3.1 Artificial neural networks

Artificial neural networks (NN) [2] are the root of deep learning. This architecture comprises of multiple neurons (or perceptrons) that can be activated whenever a certain condition is met, similar to those inside human brains. Nevertheless, in the artificial NN context, the neurons are normally grouped as layers, with the most basic model being a feed-forward NN as shown in Figure 2.3. In this simple network, there is an input layer of 3 neurons, two 4-neuron hidden layers in the middle, and a single-neuron output layer, where the neurons feed their outputs to all of the neurons in the successor layers and only in the forward direction. Deep learning is the term usually used to refer to deep NNs with a large number of layers.

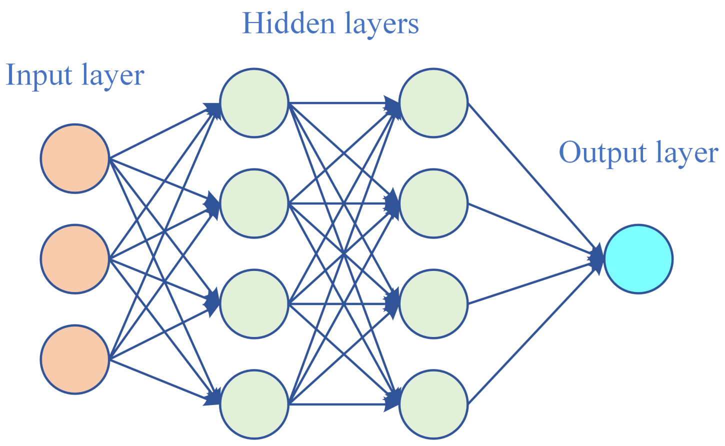


Figure 2.3: A basic feedforward neural network.

Assuming that one has a NN containing L layers with the number of neurons in each layer as $\{N_1, N_2, ..., N_L\}$, the value of the i^{th} neuron in the l^{th} layer is formulated as a function of all output values from the previous layer as

$$x_{i,l} = \sigma(\mathsf{w}_{i,l}^T x_{i-1} + b_{i,l}), \tag{2.1}$$

where $\mathsf{w}_{i,l}^T$ and $b_{i,l}$ are the weight and bias vectors of length N_{l-1}, x_{i-1} is the output vector of layer $l-1$, and σ is the activation function.

The activation function is introduced into NNs to enhance their ability to learn non-linear relationships. Table 2.1 lists the commonly used functions in neural networks.

Name	$\boldsymbol{\sigma(x_i)}$	**Range**
linear	x_i	$(-\infty, \infty)$
sigmoid	$\frac{1}{1+\exp(x_i)}$	$(0, 1)$
tanh	$\tanh(x_i)$	$(-1, 1)$
ReLU [8]	$\max(0, x_i)$	$[0, \infty)$
softmax	$\frac{\exp(x_i)}{\sum_j(\exp(x_j)}$	$(0, 1)$

Table 2.1: Activation functions.

The goal of NN is finding all weights and biases that minimise some cost functions, which might be the sum or average over the entire training set of the loss functions for all samples like the mean square error $\|\mathsf{u} - \mathsf{v}\|^2$, or the cross-entropy $-\sum_i \mathsf{u}_i \log(\mathsf{v}_i)$, where u is the trained output and v is the expected output.

The algorithm used to find the weights and biases minimising the loss in NN is called Stochastic gradient descent (SGD) [9], which is a first-order optimisation algorithm. Let θ denote the set of parameters for weights and biases, SGD first randomises initial values for θ_0, then iterates to update θ as

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t), \tag{2.2}$$

where η is the learning rate, and $L(\theta_t)$ is the cost function of a randomly selected subset of samples. The gradient $\nabla L(\theta_t)$ is calculated using the backpropagation method. To speed up the whole learning process, certain improvements have been made by means of adaptive learning rate algorithms, e.g. Adam [10], Adagrad [11], or ADADELTA [12], where each one has its own strengths and weaknesses, and choosing the algorithm best suiting the use case is often needed.

2.3.2 Embedding layer

Embedding is a neural network technique that is usually used to deal with high-dimensional sparse features, for instance, words in natural language processing. Assuming a dictionary (or a set of items) of size V, representing the items in the dictionary with only one feature column is meaningless. As an example, if index 1 is used for the word "machine" and 2 for "learning", it implies that "learning" is two times greater than "machine", which obviously lacks any justification. As such, one way to represent those features is to use one-hot encoding vectors of length V. However, as the dictionary grows, the dimensionality also increases linearly, and the feature space becomes excessively large. Due to the curse of dimensionality, the performance of any ML model with such high feature space will be affected severely. Therefore, there is a need for a method of representing those features more efficiently.

Embedding converts the items into dense vectors of real numbers, usually ranged from -1 to 1. This type of variables are advantageous in neural networks by virtue of the nature of the activation functions, thus the models will be trained better with embedded inputs. Moreover, embedding captures the essentials of the distributional hypothesis in linguistics [13], which states that the words that appear in similar context are likely similar in meanings. In NNs, the variables in the embedding layer will be updated with backpropagation in the training phase, and afterwards the distance between two closely related words or items in the embedded vector space will appear smaller.

In this project, embedding is useful as there can be a large set of users, resource types, or locations.

2.3.3 Autoencoder

Autoencoder is an unsupervised deep learning technique where the network tries to captures the characteristics of the data in a low dimensional space and reconstruct them back to their original form. The structure of an autoencoder, presented in Figure 2.4, contains the input layer, the output layer of the same dimension to the input layer, and one or more hidden layers that have lower number of neurons. The hidden layer in the middle, which has the least number of neurons, is called the bottleneck layer, and is often used as a dimensionality-reduced representation of the inputs. The structure from the input layer to the bottleneck layer is called an encoder, while the one from the bottleneck layer to the outputlayer is a decoder. The loss function of an autoencoder is the mean squared difference between the output layer and the input layer, which is also referred to as reconstruction loss.

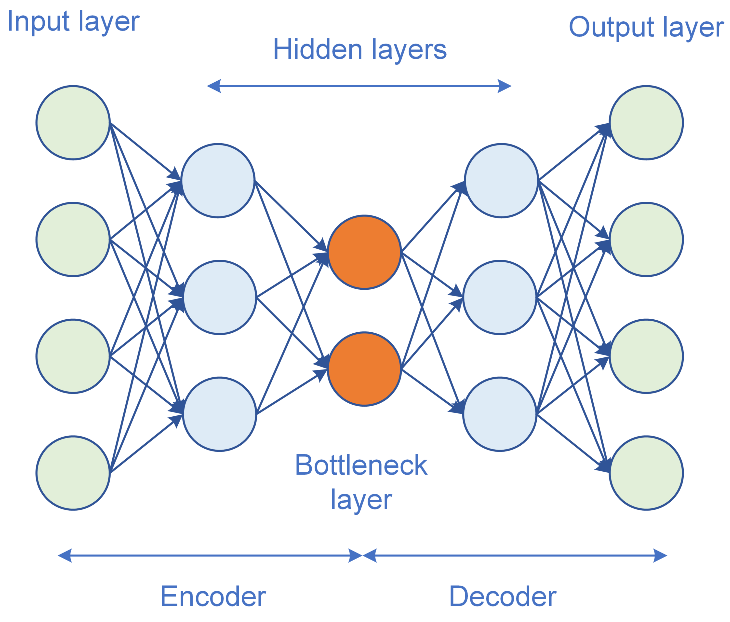


Figure 2.4: A simple autoencoder structure.

2.4 One-class Support vector machine

2.4.1 Overview

OCSVM [7] is a more conventional-style ML algorithm used for one-class classification problems in anomaly novelty detection. It developed from the idea of support vector machine, where the model tries to find the hyperplane that maximise the margin separating the data points. In OCSVM, the hyperplane is found to best separate the data points from the origin (Figure 2.5).

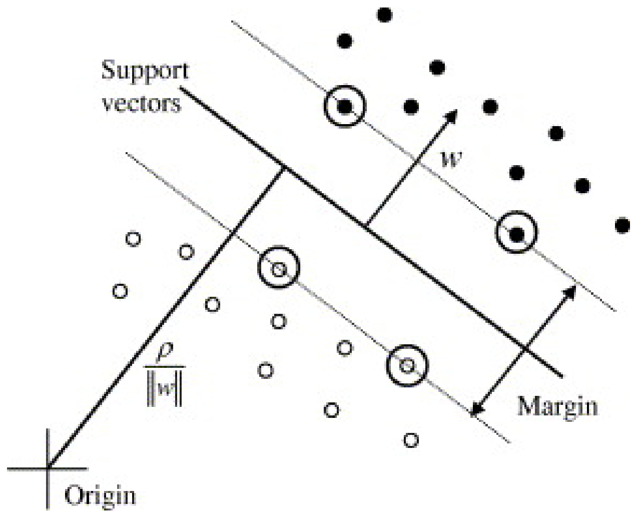


Figure 2.5: [14] Hyperplane of OCSVM.

In addition, support vector machine in general, and OCSVM in particular, have the ability to capture the non-linearity in a highly effective fashion by using the kernel trick. The kernel method maps the data points from the input feature space to a infinite-dimensional space where the data is linearly separable by a transformation ϕ, as shown in Figure 2.6. The kernel function that is utilised in most cases for SVM are the linear kernel $K(x, x') = x^T x'$ and the radial basis function (RBF) kernel $K(x, x') = \exp(-\frac{\|x-x'\|^2}{2\sigma^2})$.

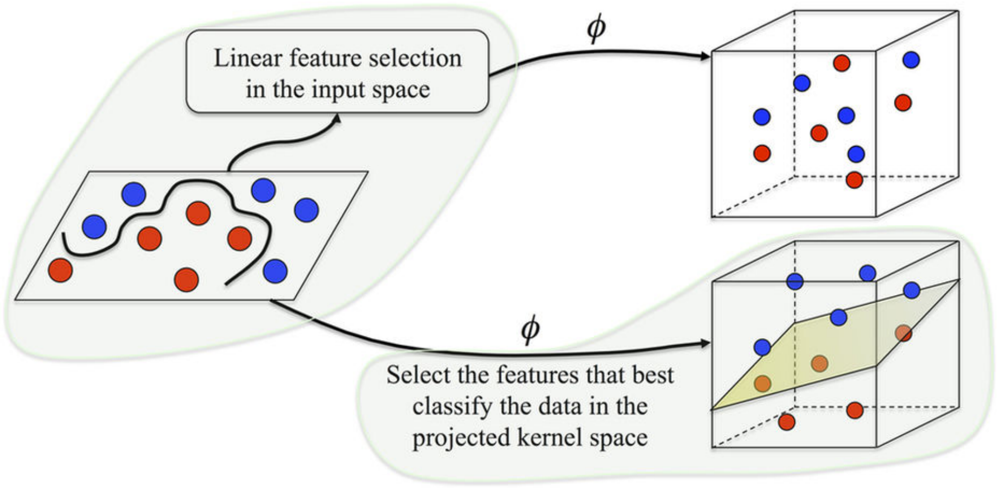

Linear feature selection in the input space
ϕ
ϕ
Select the features that best classify the data in the projected kernel space

Figure 2.6: [15] Illustration of the kernel method used in SVMs.

2.4.2 Training

The objective function of OCSVM is as follows

$$\min_{\mathsf{w},\xi,\rho} \frac{1}{2}\|\mathsf{w}\|^2 - \rho + \frac{1}{\nu n}\sum_{i=1}^{n} \xi_i, \tag{2.3}$$
$$\text{subject to } \mathsf{w}^T\phi(x_i) \geq \rho - \xi_i, \xi_i \geq 0.$$

where ξ_i is a slack variable, ρ is the offset parameter determining the distance from the origin to the hyperplane, and ν is the regularisation parameter. Theoretically, ν is the upper bound of the fraction of anomalies in the data, and is often required to be tuned carefully. Also, by replacing ξ_i with the hinge loss, the unconstrained objective function is formulated as

$$\min_{\mathsf{w},\rho} \frac{1}{2}\|\mathsf{w}\|^2 - \rho + \frac{1}{\nu n}\sum_{i=1}^{n} \max(0, \rho - \mathsf{w}^T\phi(x_i)). \tag{2.4}$$

or

$$\min_{\mathsf{w},\rho} \frac{1}{2}\|\mathsf{w}\|^2 - \rho + \frac{1}{2\nu n}\sum_{i=1}^{n} \big(\rho - \mathsf{w}^T\phi(x_i) + |\rho - \mathsf{w}^T\phi(x_i)|\big). \tag{2.5}$$

Let $g(x) = \mathsf{w}.\phi(x_i) - \rho$, the decision function of One-class SVM is

$$f(x) = sgn(g(x)) = \begin{cases} 1 & \text{if } g(x) \geq 0 \\ -1 & \text{if } g(x) < 0 \end{cases}. \tag{2.6}$$

In scenarios where an anomaly score is needed to assess the chance that a sample is anomalous, it can be given as [16]

$$f_{score}(x) = \frac{g_{max} - g(x)}{g_{max}}, \tag{2.7}$$

where $g_{max} = \underset{i=1,2,..,n}{max} |g(x_i)|$.

The optimisation problem of SVM in (2.4) is usually solved in dual space with the use of Lagrangian multipliers to reduce the complexity while increasing the feasibility. LIBSVM [17] is the most popular library that provides efficient optimisation algorithms to train SVM, and has been widely used in the research community. However, solving SVM in the dual space can be susceptible to the data size, since a kernel function has to be calculated and stored for every pair of data points.

2.4.3 Kernel approximation with Random Fourier features

To address the mentioned problem of SVMs, approximation algorithms have been introduced and widely applied, with the most two dominant being Nyströem [18] and Random Fourier features (RFF) [19]. Between the two, Random Fourier mapping method has lower complexity and does not require pre-training, and is the one being chosen in this project. The method is based on the Fourier transform of the kernel function, given by a Gaussian distribution:

$$p(\omega) = \mathcal{N}(0, \sigma^{-2}\mathbb{I}), \tag{2.8}$$

where $\mathbb{I}$ is the identity matrix and σ is a tunable hyperparameter representing the standard deviation of the Gaussian process.

From the distribution p, D independent and identically distributed (IID) weights ω_1, ω_2, ..., ω_D are drawn. In the original paper [19], two mappings are introduced, which are:

- The combined *cosine* and *sine* mapping as

$$z_\omega(x) = \Big[cos(\omega^T x) \quad sin(\omega^T x)\Big]^T, \tag{2.9}$$

  which leads to the complete mapping being defined as follows

$$z(x) = \sqrt{\frac{1}{D}} \Big[cos(\omega_1^T x) \quad ... \quad cos(\omega_D^T x) \quad sin(\omega_1^T x) \quad ... \quad sin(\omega_D^T x)\Big]^T. \tag{2.10}$$

- The offset *cosine* mapping as

$$z_\omega(x) = \sqrt{2}cos(\omega^T x + b), \tag{2.11}$$

where the offset parameter $b \sim U(0, 2\pi)$. Consequently, the complete mapping in this case is

$$z(x) = \sqrt{\frac{2}{D}} \left[cos(\omega_1^T x + b) \quad ... \quad cos(\omega_D^T x + b) \right]^T. \tag{2.12}$$

It has been proven in [20] that the former mapping outperforms the latter one in approximating RBF kernels due to the fact that no phase shift is introduced as a result of the offset variable b. Therefore, it is chosen to be implemented, even though the offset *cosine* version is widely implemented by popular machine learning tools.

Applying the kernel approximation mappings to (2.4), the unconstrained OCSVM objective function with hinge loss becomes

$$\min_{\mathsf{w},\rho} \frac{1}{2} \|\mathsf{w}\|^2 - \rho + \frac{1}{\nu n} \sum_{i=1}^{n} \max(0, \rho - \mathsf{w}^T z(x_i)), \tag{2.13}$$

which is equivalent to a linear OCSVM in the approximated kernel space $\mathcal{R}^D$, and thus the optimization problem is more trivial, despite the dimensionality of $\mathcal{R}^D$ being higher than that of $\mathcal{R}^d$.

2.5 Gradient-based black-box explanation methods

One important objective of the project is deriving a method to explain the decision making of the AD model. This is normally challenging due to the complicated structure of neural networks, as well as the infinite-dimensionality of kernels. To tackle this issue, gradients can be used to score the contribution of each input features to the output function, in this case, the margin of the OCSVM.

In the recent years, many research studies [21, 22] have applied this approach to explain the classification decision and sensitivity of input features in deep neural networks and especially convolutional neural networks. Intuitively, an input dimension $\mathbf{x}_i$ has larger contribution to a latent node $\mathbf{y}$ if the gradient of $\mathbf{y}$ with respect to $\mathbf{x}_i$ is higher, and vice versa. This same concept can be applied to kernel-approximated support vector machines to score the importance of each input feature to the margin that separates the decision hyperplane.

3 Software reviews

This section describes the main software packages that are used in this project. The main programming language is Python [23], a high-level language being adopted widely for both scientists and software engineers thanks to its natural-language style and interpretability.

3.1 TensorFlow

TensorFlow (TF) [24] is an open-source library developed by Google in 2016, with the mission of elevating ML, as well as deep learning in particular, on large-scale and distributed systems. TF defines high-level computational graphs consisting of nodes called tensors. The operations in TF are not executed at compile time like ordinary mathematical functions, but instead, the whole graph is transferred to the processing units where the calculations take place. The library also provides optimisation tools for SGD, as well as the adaptive learning rate algorithm mentioned in section 2.3.1. To accelerate the training processes in deep learning, TF also has the ability to utilise GPUs, which are more capable of doing matrices manipulation.

TF provides a visualisation tool in TensorBoard, which runs as an interactive website. Figure 3.1 shows the TensorBoard's feature where the graph definition of the model is visualised, containing all operators and tensors. Several other features are also available, such as histograms or component analysis projections of the embedding layers, which give even more analytical power to the data scientists and engineers. The package also includes tools in the form of TF Serving to serve the ML models via web servers by exposing application programming interfaces (APIs) that implement Google's Protobuf data format [1]. This opens the potential for convenient integration of the ML services with the cloud systems

3.2 Numpy

Numpy [25] is a package that contains high-performance scientific computing. Compared to the native mathematical functions of Python, Numpy provides more powerful tools in the form of matrices and extra functions that manipulate those. It is also used as a tool to process and store the data in a more efficient way.

[1]https://github.com/google/protobuf

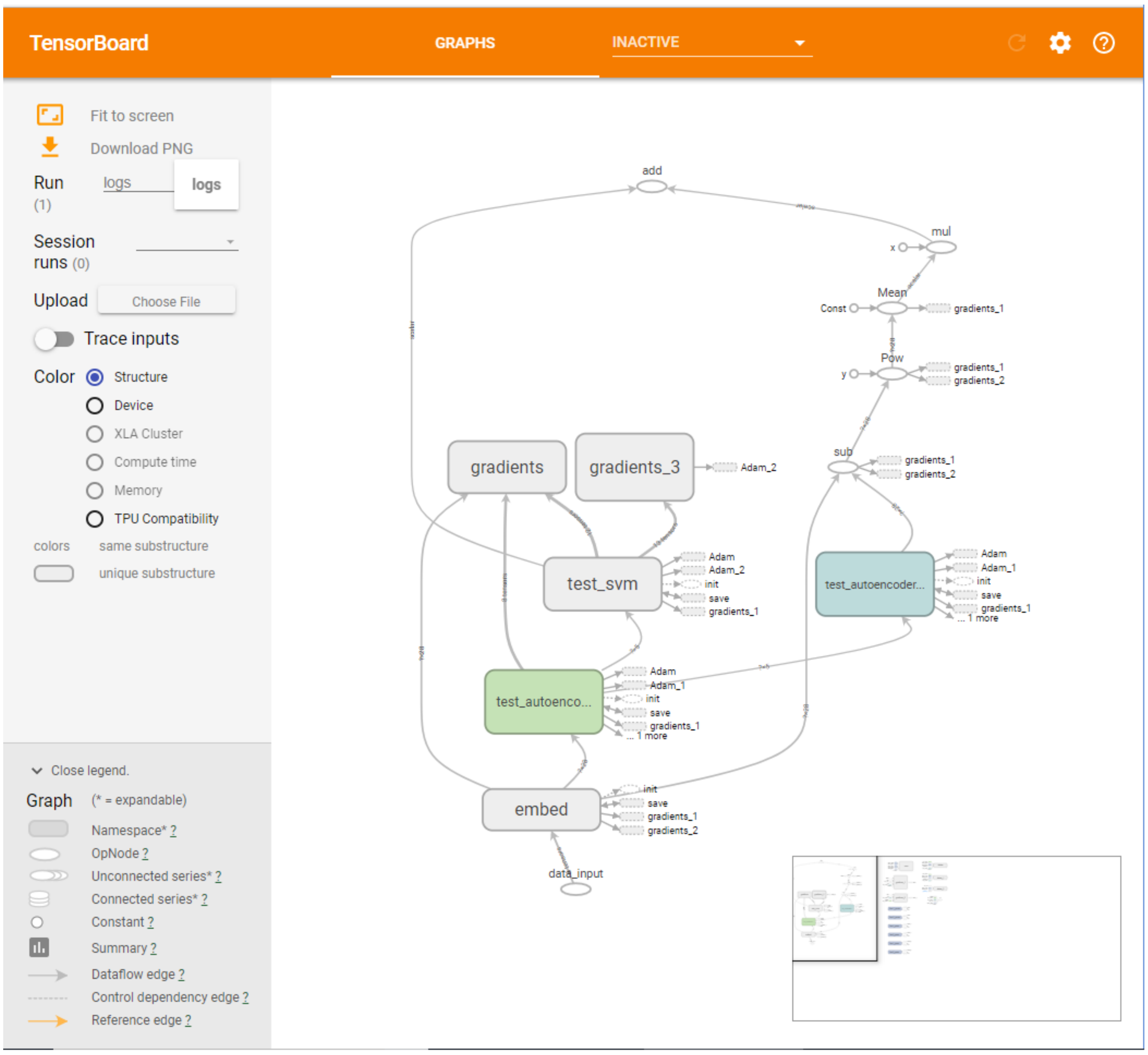


Figure 3.1: TensorBoard interface, including the graph visualisation of the model implemented for this project.

3.3 Scikit-learn

Scikit-learn [26] comprises of most of the conventional non-deep-learning ML algorithms, from regression to classification, clustering, component analysis, .etc. It also includes multiple useful tools for ML experiments. It is used mainly in the project to implement the baseline methods, perform data preprocessing and evaluation metrics for all experiments as described later in Section 5.

4 Proposed solutions

In this section, the solutions for the anomaly detecting and interpreting problems are presented and justified. First, procedures to extract and process the training data from the raw data obtained from cloud services are explained. After that, the ML model is described in details, followed by the mathematical derivations to calculate the gradients that are used to explain the decision making of the model.

4.1 Feature engineering

4.1.1 Data extraction

The data retrieved from the system come from either a PostgreSQL database or a JSON file. The structure of each sample in the data consists of the following fields:

- *User*: Indicates the human or robot user that makes the access. The set of users can scale to the size of the cloud-service project.
- *Operation*: Indicates the action, including ACCESS, MODIFY, CREATE, MOVE, DELETE, etc., which make a finite set.
- *Resource*: Indicates the subject of the action. The data includes a resource type that is one of those listed in GCP documentations [2].
- *Resource location*: The location of the resource is comprised of a zone, also listed in GCP documentations [3], and an IP address.
- *Request location*: The location of the user contains an IP address and the country that they make the request from.
- *Time*: Indicates the exact time that the action occurs.

Python scripts to extract the data from the database to Python's data types have been written. The manipulation of the data is described in the next subsection.

[2]https://cloud.google.com/deployment-manager/docs/configuration/supported-resource-types
[3]https://cloud.google.com/compute/docs/regions-zones/

4.1.2 Feature engineering

The user, operation, and resource are indexed by integer values is used. For resource feature, only the resource type is taken into account, as there can be an infinite number of resources in the data. As the amounts of users and resources are large, the embedding technique, as mentioned in 2.3.2, is used to represent that information in the feature space. In contrast, operations are in a finite set, therefore a three or four-dimensional binary encoding is chosen instead. Both encoding types are realised using the embedding lookup function of TF. The difference is that for binary encoding, constant lookup array is provided, whereas to implement embedding layer, trainable variables are initialised and passed into the function. One-hot encoding can also be used in place of binary encoding.

The next entity to deal with is time. Since the time retrieved from the data are regular Python datetime values, it is a linear function, and thus might be misleading. For example, 23:00PM and 0:00AM is close to each other, but in a linear scale, they are significantly far away. Therefore, it is converted into a circle in two-dimensional space to account for the cyclic characteristic (Figure 4.1). The conversion is done as follows

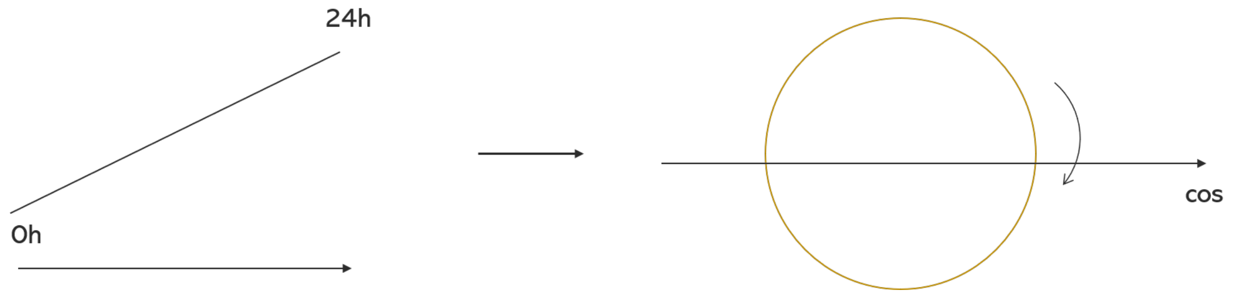


Figure 4.1: Illustration of time conversion.

$$\begin{cases} x_1^{day} = \cos(\frac{2\pi t}{86400}) \\ x_2^{day} = \sin(\frac{2\pi t}{86400}) \end{cases}, \tag{4.1}$$

with t being the linear time value. Moreover, the weekday cycle might be of interest, since it can be speculated that more anomalies take place in some specific days of the week, e.g. non-working days. A similar conversion can be performed as

$$\begin{cases} x_1^{week} = \cos(\frac{2\pi t}{86400\times 7}) \\ x_2^{week} = \sin(\frac{2\pi t}{86400\times 7}) \end{cases}. \tag{4.2}$$

The last entity to consider is location. A quick look into the data suggests that IP address field comes with both IPv4 and IPv6 values, and it would be challenging to capture the structural attributes, especially in our case where it is of global scale. Since each IP address appears to be independent of others, the same embedding approach can be utilised.

Besides, for request country and resource zone fields, one can try to capture the geographical position by looking up the coordinates of the location names using tools like the Geopy package in Python [4], and convert them into a 3-dimensional spherical representation as follows

$$\begin{cases} x_1^{coor} = \cos(a) \times \cos(b) \\ x_2^{coor} = \cos(a) \times \sin(b) \\ x_3^{coor} = \sin(a) \end{cases}, \tag{4.3}$$

where a is the latitude, and b is the longitude.

In addition, another way to represent the resource location can be a nested tree, since according to the GCP documentations, they are organised in a hierarchical manner. For instance, a zone named "europe-west-1a" indicates that there are 4 levels of division in resource zones.

4.1.3 Summary

Table 4.1 summarise the fields, sizes and extraction methods of the dataset that is used during the project. The processed dimensions of the operation, resource, resource zone, request country, request IP, and time fields according to the scope of the available dataset are 6, 2, 7, 3, 8, and 2, respectively. The total dimensionality is 28, and can scale proportionally with the number of resource types and IP addresses, or if the locations are extracted using another method.

[4] https://github.com/geopy/geopy/

Field	**Description**	**Size**	**Method**
User	Human or Robot accounts	Scale with cloud project	Drop
Operation	Action performed by user	Small & finite set: ACCESS, MODIFY, CREATE, MOVE, DELETE, .etc	One-hot encoding
Resource	Subject of the action. Defined by GCP	Large & finite set of types	Embedding layer
Resource zone	The zone of the resource (e.g. Europe-west-1a)	Large & finite number	Convert to 3D geographical coordinates
Request country	The country of the action performer	Large & finite number	Convert to 3D geographical coordinates
Request IP	The IP address of the action performer	Large number & scale with project scope	Embedding layer
Time	Time of the action	Continuous linear	Convert to sine and cosine

Table 4.1: Summary of the data used in the project and their extraction methods.

4.2 Machine learning model

In deep learning, the autoencoder structure can be used for anomaly detection in a semi-supervised manner. The model is trained with normal data, and a threshold on the reconstruction error is set such that if the error for any sample exceeds the threshold, it is detected as an anomaly. Nonetheless, choosing the value for this threshold is problematic, especially when there is heterogeneity in the data [27]. One-class SVM, as mentioned, is the most commonly used conventional technique in anomaly detection, while having its own limitations when it comes to big data size and high dimensionality. Inspired by those reasons, in this project, a novel deep learning model is proposed to combine the scalability of deep learning and the robustness of SVM in detecting anomalous actions.

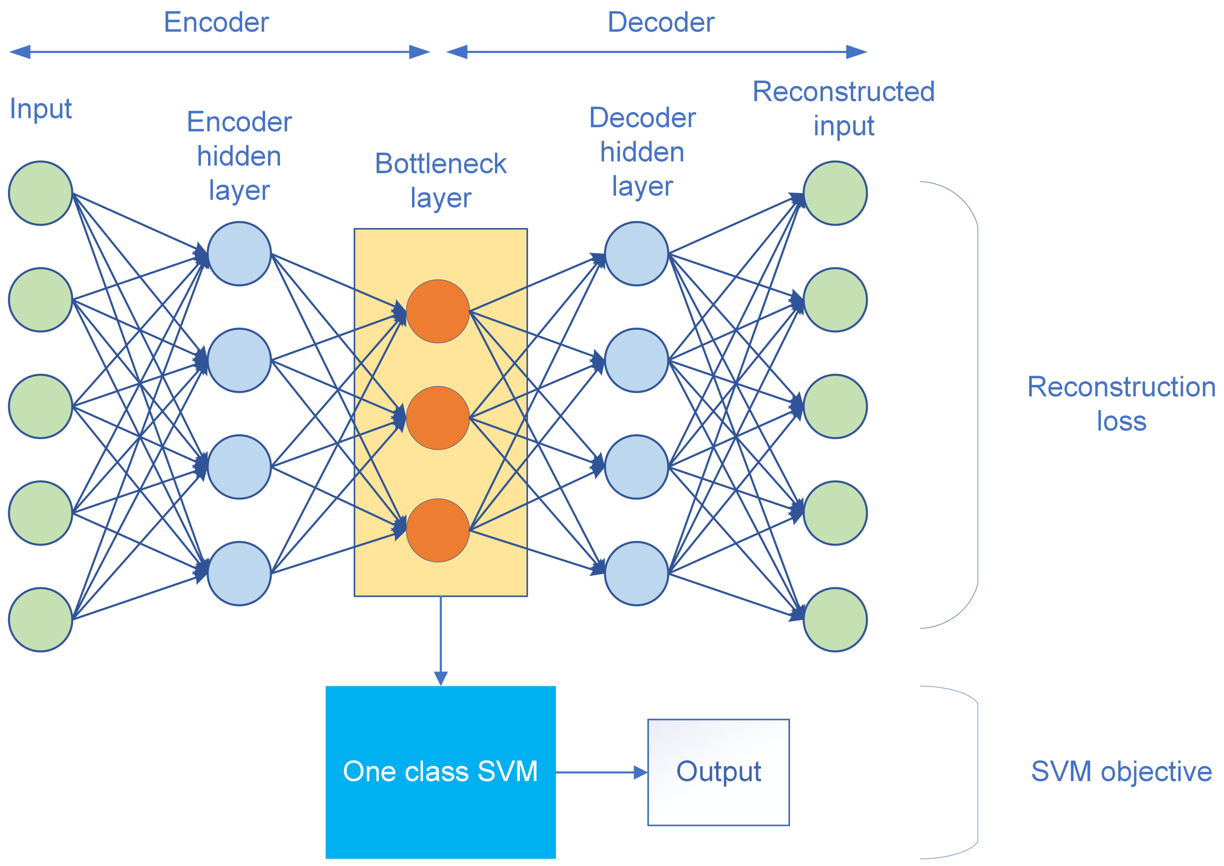


Figure 4.2: The combined model structure of Autoencoder and One-class SVM.

The model is comprised of a deep autoencoder and a one-class SVM (AE-1SVM) whose weights w and offset ρ are determined by training with SGD simultaneously with the autoencoder. The autoencoder acts mainly as feature representation, and the bottleneck layer is fed directly to the SVM. The loss function in this model is the weighted sum of the reconstruction loss and the SVM objective function, as formulated below

$$Q(\theta, \mathsf{w}, \rho) = \alpha L(\mathbf{x}, \mathbf{x}') + \frac{1}{2}\|\mathsf{w}\|^2 - \rho + \frac{1}{\nu n}\sum_{i=1}^{n}\max(0, \rho - \mathsf{w}^T z(x_i)) \tag{4.4}$$

The components and parameters in (4.4) are

- $L(\mathbf{x}, \mathbf{x}')$ is the reconstruction loss of the autoencoder, which is normally chosen to be the L2-norm loss $L(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_2^2$.
- Since SGD is utilised, the variable n, which is formerly the number of training samples, becomes the batch size since the hinge loss is calculated using the data points in the batch.
- z is the Random Fourier mappings as defined in (2.10). Due to the random features being data-independent, the standard deviation σ of the Gaussian distribution has to be fine-tuned correlatively with the parameter ν.
- α is a hyperparameter controlling the trade-off between feature compression and SVM margin optimization. In most cases, $\alpha = 1$ is sufficient to achieve appropriate results.

The output of the model is based on the decision function of the one-class SVM. Because of the unsupervised nature of one-class SVM and the autoencoder being utilised only to compress the feature space, this model can also be used in unsupervised mode.

4.3 Derivation of the gradients of the proposed model

Considering an input x of a RFF kernel-approximated OCSVM with dimension R^d. In the AE-1SVM model, x is the bottleneck representation of the latent space in the deep autoencoder. The expression of the margin $g(x)$ with respect to the input x is as follows

$$\begin{aligned}
g(x) &= \sum_{j=1}^{D}\mathsf{w}_j z_{\omega_j}(x) - \rho \\
&= \sqrt{\frac{1}{D}}\sum_{j=1}^{D}\left[\mathsf{w}_j cos(\omega_j^T x) + \mathsf{w}_{D+j} sin(\omega_j^T x)\right] - \rho \\
&= \sqrt{\frac{1}{D}}\sum_{j=1}^{D}\left[\mathsf{w}_j cos(\sum_{k=1}^{d}\omega_{jk}x_k) + \mathsf{w}_{D+j} sin(\sum_{k=1}^{d}\omega_{jk}x_k)\right] - \rho.
\end{aligned} \tag{4.5}$$

As a result, the gradient of the margin function on each input dimension $k = 1, 2, ..., d$ can be calculated as

$$\frac{\partial g}{\partial x_k} = \sqrt{\frac{1}{D}} \sum_{j=1}^{D} \omega_{jk} \big[-\mathsf{w}_j sin(\sum_{k=1}^{d} \omega_{jk} x_k) + \mathsf{w}_{j+D} cos(\sum_{k=1}^{d} \omega_{jk} x_k) \big]. \tag{4.6}$$

Next, the gradient of the latent space nodes with respect to the deep autoencoder's input layer can be derived. In general, considering a neural network with M input neurons $x_m, m = 1, 2, ..., M$, and the first hidden layer having N neurons $u_n, n = 1, 2, ..., N$, as depicted in Figure 4.3. The gradient of u_n with respect to x_m can be derived as

$$G(x_m, u_n) = \frac{\partial u_n}{\partial x_m} = w_{mn} \sigma'(x_m w_{mn} + b_{mn}) \sigma(x_m w_{mn} + b_{mn}), \tag{4.7}$$

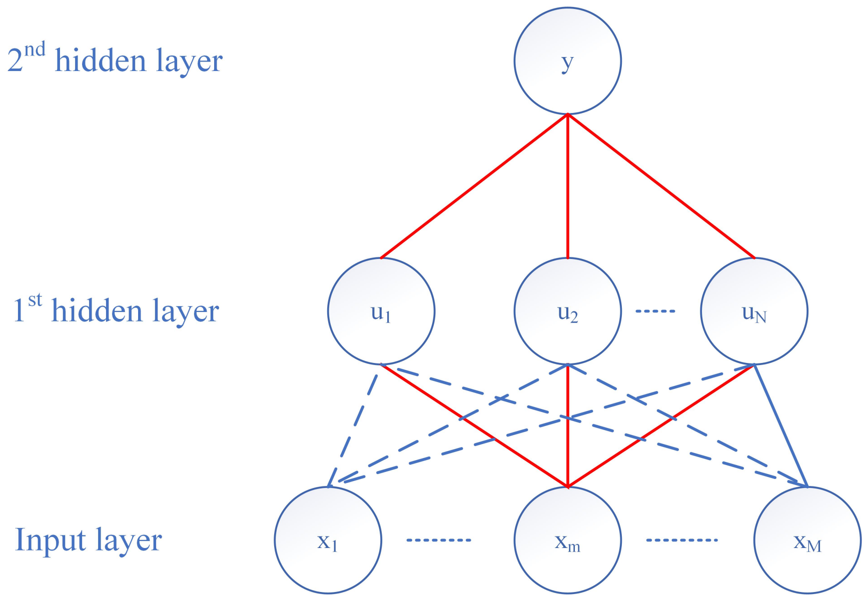


Figure 4.3: Connections between input layer and hidden layers of a neural network.

where $\sigma(x_m w_{mn} + b_{mn}) = u_n$, $\sigma(.)$ is the activation function, w_{mn} and b_{mn} are the weight and bias connecting x_m and u_n. The derivative of σ is different for each activation function. For instance, with a sigmoid activation σ, the gradient $G(x_m, u_n)$ is computed as $w_{mn} u_n (1 - u_n)$, while $G(x_m, u_n)$ is $w_{mn}(1 - u_n^2)$ for *tanh* activation function.

To calculate the gradient of neuron y_l in the second hidden layer with respect to x_m, chain rule and sum rule can simply be applied as follows

$$G(x_m, y_l) = \frac{\partial y_l}{\partial x_m} = \sum_{n=1}^{N} \frac{\partial y_l}{\partial u_n} \frac{\partial u_n}{\partial x_m} = \sum_{n=1}^{N} G(u_n, y_l) G(x_m, u_n). \tag{4.8}$$

The gradient $G(u_n, y_l)$ can be obtained in a similar manner to (4.7). By maintaining the values of G at each hidden layer, the gradient of any hidden or output layer with respect to the input layer can be calculated. Finally, combining this and (4.6), the end-to-end contribution scores of all input features on the OCSVM margin can be acquired.

TF also provides a function named *tf.gradients* that has the ability to obtain the gradients between any two tensors, which have already been calculated as a step in backpropagation process. Therefore, the function can be utilised to simplify the implementation of the method to reduce computational redundancy.

4.4 Rules of interpretation

Using the gradients retrieved by the model, the rules to interpret the decision can be made according to the following propositions:

- *For an anomalous sample, the dimension which has higher gradient has higher contribution to the decision making of the ML model. In other words, the sample is further from the centroid of the boundary in that particular dimension.*
- *For each mentioned dimension, if the gradient is positive, the value of the feature in that dimension is smaller than the the lower limit of the boundary. In contrast, if the gradient holds a negative value, the feature exceeds the level of the normal class.*

From a mathematical point of view, the gradient of a function with respect to its variable indicates the rate of change of that function as the variable changes. In this case, the function in question is the margin $g(x) = \mathsf{w}.\phi(x_i) - \rho$, where a lower margin value means a more anomalous sample. As such, if the rate of change of a dimension is high, one can interpret that "adjusting the value this feature can result in a more significant change to the margin".

Furthermore, the sign of the gradient also holds useful information. If the gradient is greater than 0, it means increasing the variable will increase the value of the function, suggesting that on the $(-\infty, \infty)$ axis, the value of the variable is on the left side of the maximum that represents the centroid of the boundary, and vice versa.

5 Implementation and Experiments

The implementation details and experiment procedures are showcased in the scope of this section. As mentioned, all code is written in Python, with the deep learning model implemented using TF. Multiple experiments are carried out to validate the proposed models and algorithms. After that, results on the provided dataset of cloud access are described. Additionally, a developmental graphical interface is created as the tool to experiment and visualise the working model.

5.1 Hardware and software preparation

For developmental purposes, a server has been provided by the industrial partner SAP. The server specifications are as follows:

- CPU: Intel® Xeon® X5650 24 Dual Core @ 2.67GHz
- RAM: 150Gb
- GPU: NVIDIA Tesla K20Xm
- Operating system: Ubuntu 16.04.3

Also, the necessary software has been installed, including NVIDIA drivers, Docker for automated deployment and testing, Python 2.7.12 and 3.5.2 with tensorflow-gpu package and its dependencies. A similar setup is prepared in a personal laptop to deal with small experiments whose results must be monitored, collected and analysed continuously.

5.2 Anomaly detecting model validation

Since the real dataset from cloud access loggings does not include any ground truth, i.e. it is not known if the samples are anomalous or not, to verify the correctness and effectiveness of the proposed model, experiments have to be performed on popular ML datasets that are used in scientific researches, with the results being compared with that of baseline methods.

5.2.1 Datasets

The following datasets listed below are used for comparison. Most of the datasets are originally used for classification. However, they are modified for the anomaly detecting task by either selecting some classes to compose normal class and other classes with different characteristics as the anomaly class or adding noise to some normal samples to make them outliers.

- **Gaussian**: This dataset is used to showcase the performance of the methods on high-dimensional and large data. The normal samples are drawn from a normal distribution with zero mean and standard deviation $\sigma = 2$, while $\sigma = 10$ for the anomalous instances. Theoretically, since the two groups have different distributional attributes, the AD model should be able to separate them.

- **USPS**: This dataset is formed from the U.S Postal Service handwritten digits dataset [28]. Among the entire dataset, 2000 samples from digit 1 are extracted as normal data, and 100 samples from digit 7 are used as anomalous data, as the appearance of the two digits are similar. Each sample is a 16×16 image of grayscale pixels that range from 0 to 255.

- **Musk**: The Musk dataset originally from the UCI repository [29]. This subset is obtained from ODDS [30], and contain normal samples combined from three non-musk classes: j146, j147, and 252, while anomalies are from the musk classes 213 and 211. The purpose of using this dataset is to test the capability of the models on a high-dimensional situation.

- **Shuttle**: The Shuttle dataset [29] is also obtained from ODDS. The normal group is comprised of samples from classes 2, 3, 5, 6, 7, while the outlier group is made of class 1.

- **KDDCup99**: KDDCup99 dataset [29] is one of the most popular dataset for AD researches. Originally, it has approximately 5 millions samples, 80% of which are anomalies. Therefore, the 10-percent dataset is used at first to give a better comparison with the conventional methods which do not work well with large-scale data. Moreover, since anomalies are usually rare, only 10,000 samples are selected for the experiments. The categorical features in this dataset is converted to one-hot encoding, resulting in 118 dimensions in total.

5.2.2 Baseline methods

The two boundary-based AD methods are used as the baselines to compare the proposed algorithm with:

- **Conventional OCSVM**: The popular implementation in Scikit-learn, which applies the LIBSVM solver to optimize the objective function, is used. The RBF kernel is taken into consideration in the experiments.

Dataset	Dimensions	Instances	Anomalies
Gaussian	512	95000	5000
USPS	256	2000	100
Musk	166	3062	97
Shuttle	9	49097	3511
KDDCup99	118	97278	10000

Table 5.1: Summary of the datasets used in the experiments.

- **Hybrid AE - OCSVM**: This is a modification to the conventional OCSVM method. A deep autoencoder is pre-trained with hidden layers similar to that in the proposed model (in terms of layer sizes, learning rate, and number of training epochs) to reduce dimensionality, and the output is fed into OCSVM.
- **Isolation Forest**: This ensemble method is based on the idea that the anomalies in the data have less frequencies and are different from the normal points.

5.2.3 Metrics

For AD, the performance metrics are usually different from the classification tasks. Accuracy is inadequate due to the rarity of anomalies. For example, if the anomaly ratio is 1% and the AD model always predict negatively, i.e. no anomaly, the accuracy will be 99%, which is significantly high. First, considering the confusion matrix as in Table 5.2 below:

	True class: Anomaly	**True class: Normal**
Predicted: Anomaly	True positive (TP)	False positive (FP)
Predicted: Normal	False negative (FN)	True negative (TN)

Table 5.2: Structure of a confusion matrix.

The following measurements can be defined from the fields of the confusion matrix:

- Precision: Measures the positive predictive value obtained by $\frac{\text{TP}}{\text{TP} + \text{TN}}$.
- Recall (sensitivity): Measures the rate of detecting anomaly calculated as $\frac{\text{TP}}{\text{TP} + \text{FN}}$.
- Fall-out: Measures the false alarm rate formulated as $\frac{\text{FN}}{\text{FP} + \text{TN}}$.

Based on the mentioned measurements, the following two metrics used for AD [31] are formed:

- Area under Receiver operating characteristic (ROC) curve: Plots recall against fall-out rate at different thresholds.
- Area under Precision-Recall (PR) curve: Plots recall against precision at different thresholds.

The ROC curve emphasises only the positive predictive values. The PR curve, on the other hand, focuses on the positive (anomaly) class instead. Obviously, a good model has to achieve a high ROC curve to better classify both normal and anomaly classes. On top of that, for models with similar area-under-ROC-curve values, the PR curve can indicate if it has fit the boundary too strictly and returns too many false alarm, since having a model that predict anomalies excessively is also not desirable.

5.2.4 Results

Results are obtained from experiments in a semi-supervised setup to simulate the scenarios of the available cloud data. For each testing dataset, the data is split in training set and testing set a 1:1 ratio, where the normal samples of the training set is used for training, and the entire testing dataset is used to measure the performance. All inputs are normalised to have the range $[-1, 1]$, using scikit-learn's MinMaxScaler tool [5].

Table 5.3 presents the results in the two aforementioned metrics. The values are written in the format [MEAN $\pm$ STANDARD ERROR] where the means and standard errors are measured after 20 runs for each experiment. In all cases, the combined autoencoder and OCSVM structure outperforms other methods regarding both metrics. It is obvious that conventional OCSVM suffers from large datasets, especially for the Gaussian dataset, which has both high dimensionality and large data size. Additionally, it can also be observed that training the autoencoder simultaneously with the OCSVM forces the autoencoder to learn better representation, as the performance of the end-to-end model exceeds the hybrid model within the same amount of iterations.

Figure 5.1 shows the training and testing time between conventional OCSVM and the end-to-end deep autoencoding OCSVM model. It can be seen that for small dataset, OCSVM outperforms the deep-learning-based model in terms of time. Nonetheless, for larger datasets

[5] http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

Area under ROC curve					
Method	**Gaussian**	**KDDCup**	**Musk**	**Shuttle**	**USPS**
OCSVM	0.5545 ± 0.0003	0.9158 ± 0.0006	0.9612 ± 0.0010	0.9688 ± 0.0003	0.9895 ± 0.0014
Hybrid-AE -OCSVM	0.9944 ± 0.0011	0.9601 ± 0.0003	0.9712 ± 0.0110	0.9125 ± 0.0023	0.9536 ± 0.0017
Isolation Forest	0.9994 ± 0.0008	0.9459 ± 0.0006	0.9843 ± 0.0256	0.9429 ± 0.0004	0.9806 ± 0.0032
AE-1SVM	0.9999 ± 0.0003	0.9705 ± 0.0013	0.9875 ± 0.0033	0.9781 ± 0.0013	0.9844 ± 0.0033
Area under Precision-Recall curve					
Method	**Gaussian**	**KDDCup**	**Musk**	**Shuttle**	**USPS**
OCSVM	0.3218 ± 0.0002	0.3349 ± 0.0015	0.1271 ± 0.0012	0.4673 ± 0.0013	0.5102 ± 0.0017
Hybrid-AE -OCSVM	0.9964 ± 0.0001	0.3876 ± 0.0012	0.1471 ± 0.0010	0.4254 ± 0.0042	0.4722 ± 0.0024
Isolation Forest	0.9994 ± 0.0008	0.4148 ± 0.0006	0.7523 ± 0.0101	0.5674 ± 0.0012	0.6250 ± 0.0054
AE-1SVM	0.9999 ± 0.0001	0.5105 ± 0.0213	0.8748 ± 0.0114	0.8688 ± 0.0202	0.8024 ± 0.0103

Table 5.3: Accuracy performance of baseline methods and the proposed method on five testing datasets.

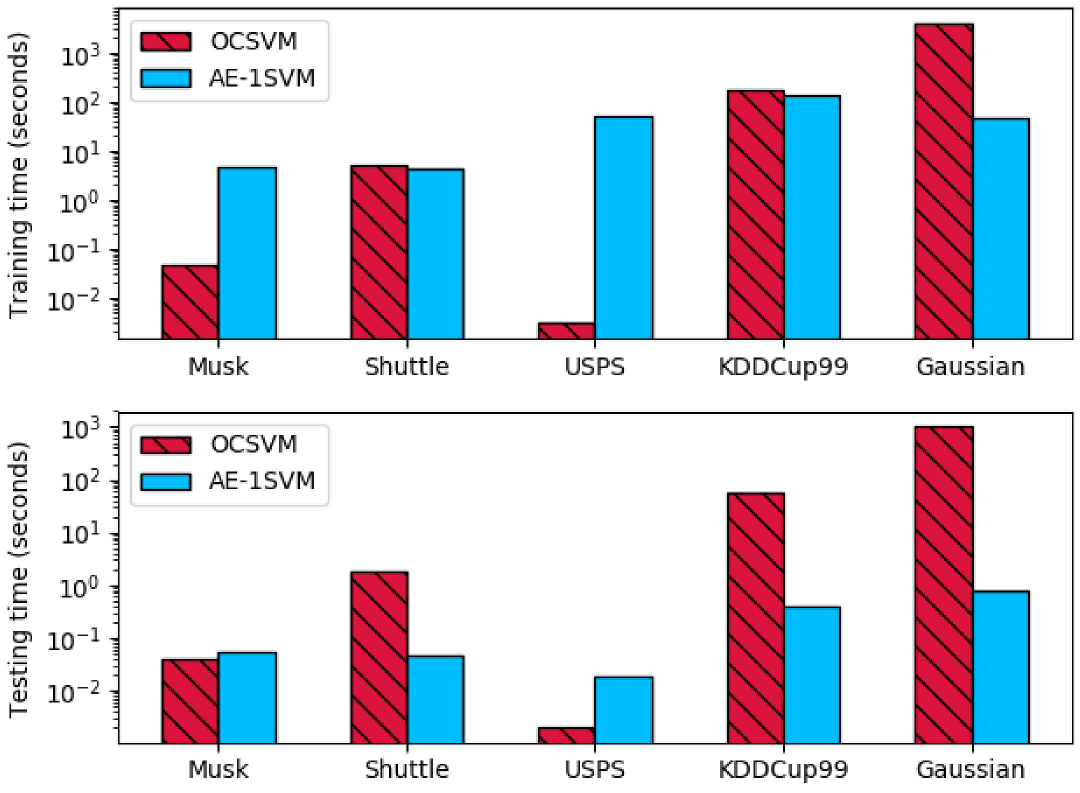


Figure 5.1: Time comparison between the proposed model and conventional OCSVM.

like KDDCup99 or generated Gaussian, OCSVM takes excessive amount of time to train, as well as to calculate the margin determining the output. Besides, regarding KDDCup99 and Gaussian datasets, which both have the size of nearly 100 thousand samples, the operating time of OCSVM is significantly higher for the Gaussian dataset, which have greater dimensionality. The training and testing time of OCSVM for KDDCup99 is around 180 and 60 seconds, respectively. Meanwhile, it takes approximately 4000 seconds to train, and 1000 seconds to run decision functions for the Gaussian dataset.

To further showcase the ability of our proposed model in big-data situations, the sizes of those datasets are boosted, and promising outcomes are achieved. The model is successfully trained on the full KDDCup99 normal class (972,781 samples) in about 200 seconds, with AUROC of 0.9703 ± 0.0010. For the Gaussian data, the size is expanded to one million training samples, which accomplishes perfect AUROC in 100 seconds. The dimensionality is increased to 1024 while another layer in the deep autoencoder is added, resulting in a similar result in 400 seconds. This has justified the prospect of the deep autoencoding OCSVM model in AD for large-scale applications, when it is almost infeasible to operate conventional OCSVM with such enormous datasets in a reasonable time frame.

5.3 Verification of gradient-based explanation method

In this section, two experiments are conducted. The first experiment is carried out on a simple example dataset to illustrate the empirical analysis of gradient-based methods. The second experiment investigate the application of the method onto the USPS dataset used in the previous section.

5.3.1 Setup

For the first experiment, considering a spherical dataset of 2000 uniformly drawn samples where the center lies at $(1.0, 1.0, 1.0)$ and the radius is $1/\sqrt{2}$. Equation (4.6) is applied to measure the contribution of each input dimension to the decision making of the OCSVM. Six test points are $P_1(1.0, 1.0, 1.0)$, $P_2(1.5, 1.5, 1.5)$, $P_3(2.0, 1.0, 1.0)$, $P_4(2.0, 0.0, 1.0)$, $P_5(2.0, 0.0, 2.0)$, and $P_6(3.0, 2.0, -2.0)$.

In the second experiment, the goal is to interpret the predictive results from the images and find the difference between anomalous digits (digits '7') to the normal digits (digits '1') in the USPS dataset according to the semantics of the gradient maps.

5.3.2 Results

The results of the first experiment are illustrated in Figure 5.2. It can be seen that the gradients of all dimensions for P_1 are negligible since the point coincides with the center of the decision hyperplane. P_3 is distant from the center only in dimension x_1, therefore $\partial g/\partial x_1$ is significantly higher. The same observations can be made for the other points, where a dimension with further distance to the nominal class has a larger gradient, and thus having more contribution to the decision making. It can also be noticed that when the values are higher than the centroid, the gradients are negative, and vice versa.

Figure 5.3, on the other hand, shows the gradient maps for six digits '7' in the USPS dataset. The second row and third row depict the positive and positive maps, respectively. Interesting patterns can be seen from the results, as the positive gradient maps are more intense in the middle part, where the brighter pixels in the digits '1' normally stand. In contrast, the negative maps are darker on the parts matching the brighter pixels on the original image, and around the center part.

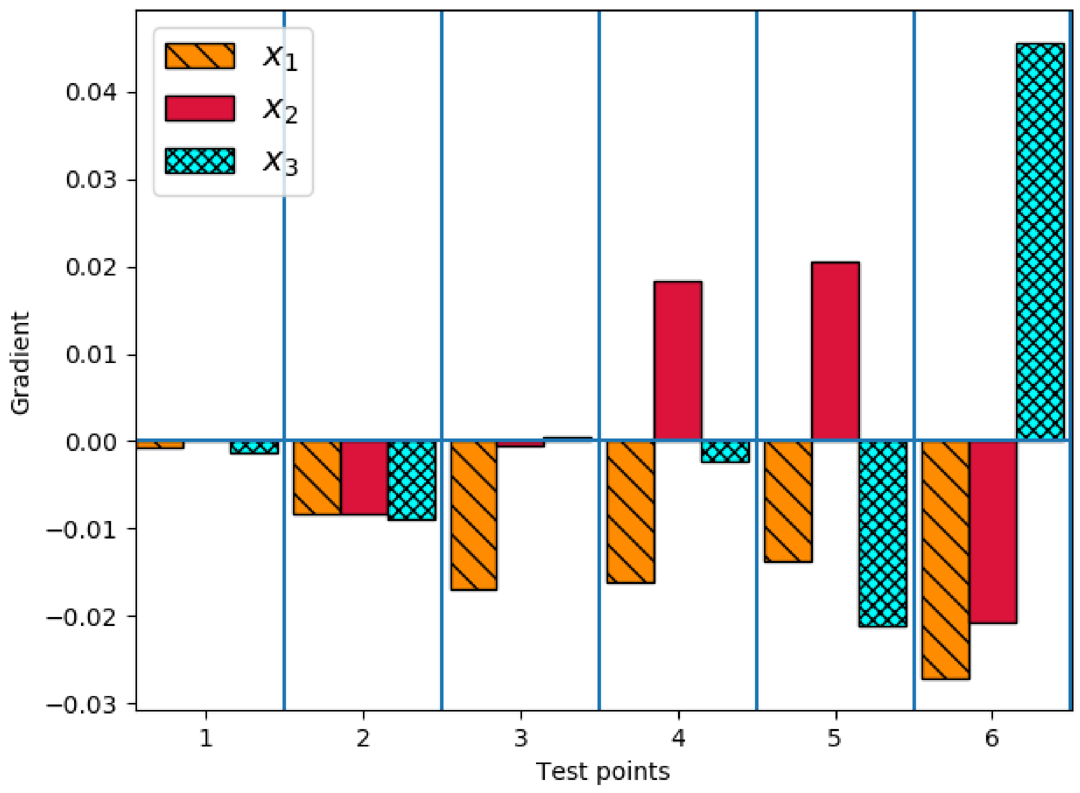


Figure 5.2: Comparison of gradients for 6 test points from the spherical experiment.

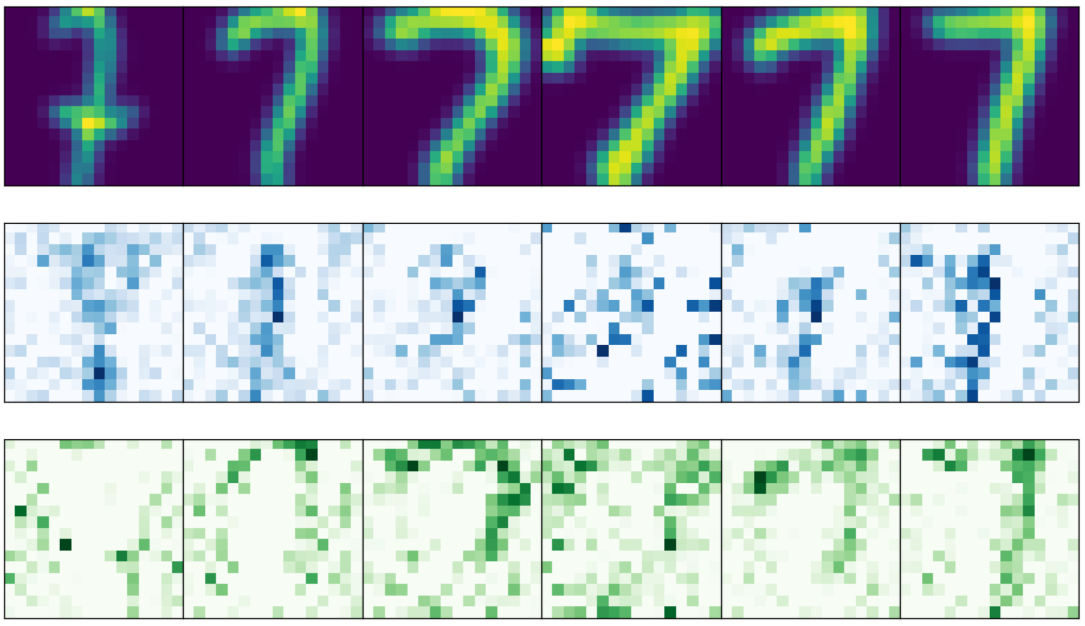


Figure 5.3: Original images and corresponding gradient maps of several digits '7'.

5.4 Observations

The following are the most perceptible points taken from the experiments with TensorFlow which can be helpful when applying on the cloud dataset:

- GPU is interestingly slower than CPU with small batch size (e.g. < 128). This is due to the large overhead when the information is transferred between the CPU and GPU, as only parts of the computations are delegated to the GPU.
- For the Autoencoder, tanh, sigmoid, and ReLU are the candidates for the activation function. All of the three functions are tested, where sigmoid gives the best performance in terms of accuracy and interpretability (better gradient maps).
- When using adaptive learning rate algorithm, it is recommended to use a high initial rate, i.e. 1.0 for Adagrad or 0.01 for Adam. However, after multiple observations, having a high initial rate sometimes make the loss fluctuate, and a lower rate performs better. As a result, it is still important to select a proper initial rate for those adaptive algorithms. Additionally, choosing the algorithm that converges the best in different data sets is also a necessity at times.

5.5 Application on cloud dataset

The proposed model is applied on the cloud dataset with the same implementation to that in the previous experiments. The data is preprocessed as mentioned in 4.1. Then, the main objectives are tuning the hyperparameters to optimise the performance of the model and visualising the results obtained from it.

As the labels of the samples are not available, it is non-trivial to access the performance of the model. However, since the results returned to the superusers also include the interpretation acquired from the gradient-based method and the ML model does not take denial actions, it is more acceptable to increase the regularisation of the model and make it more sensitive to anomalies at the cost of higher false alarm rate.

5.5.1 Model configuration

In Table 5.4 below, the details of tuning strategy and their corresponding values used on the cloud dataset are listed. Among all parameters, the OCSVM parameter ν is the most flexible one, as it indicates how strict the model will judge the access events. Choosing this parameter to find the most reasonable value is up to the superuser. Nevertheless, it is still desirable to select ν so that the model covers a majority of the training data ($90 \rightarrow 95$ %).

Figure 5.4 plots the 5-fold cross-validation training and validation score of the cloud dataset against the parameter ν. It is shown that the model does not experience over-fitting, and the best values for ν is between 0.001 and 0.3, as it is the lower bound for 90 % coverage.

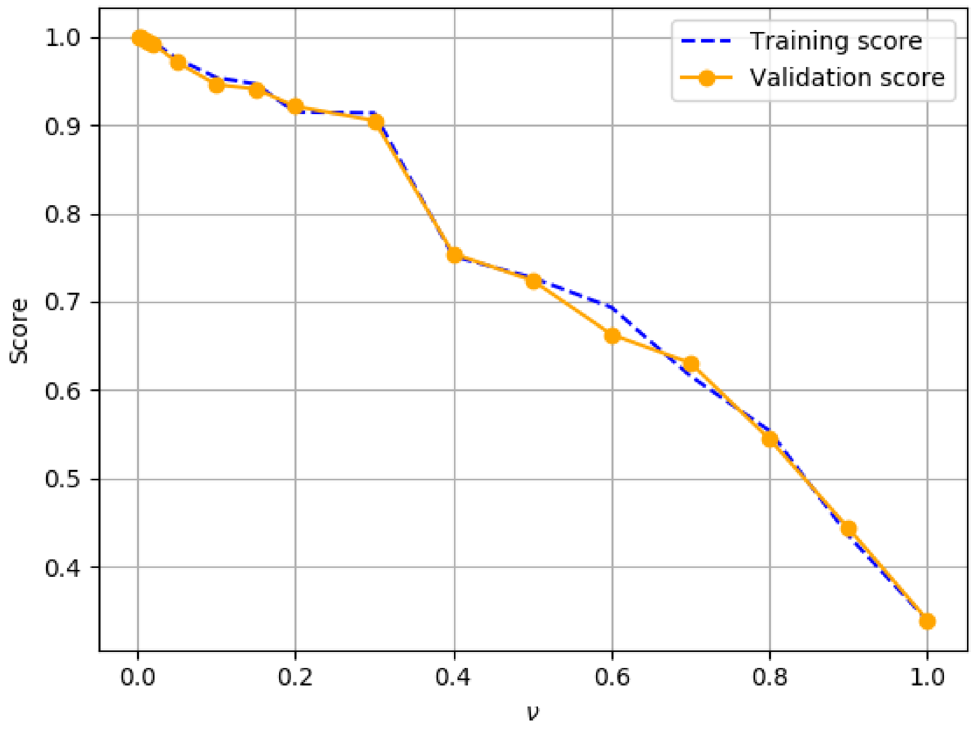


Figure 5.4: The training and validation true negative score of the model on the provided dataset against the parameter ν.

5.5.2 Graphical Experimental Interface

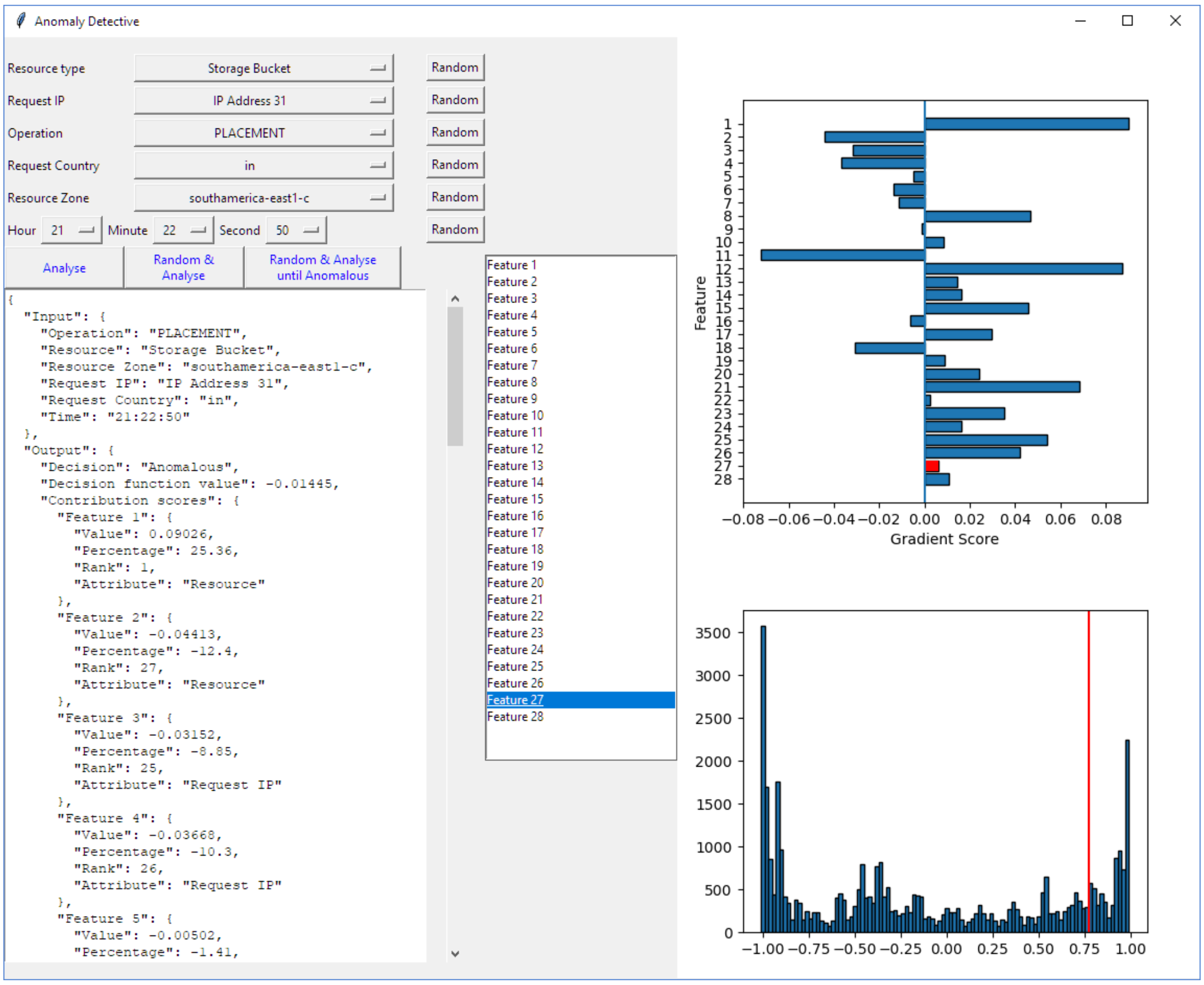
A graphical interface that supports the experiments and visualisation for the trained model is also developed using the package TkInter [6]. After being trained, the model has its structure and weights saved using TF Saver feature. It is then loaded and being consumed directly by the graphical interface without having to expose any API or server.

The interface provides the ability to select or randomise all fields of a cloud access event (top-left corner of Figure 5.5). The "Analyse" buttons will process the information of the visible fields and execute a command to TF to predict the decision value of the corresponding access event, as well as the contribution scores of each feature given by the gradients. The results are returned in a JSON format, as in the bottom-left corner of the Figure. The graph in the top-right corner plots the contribution scores of the features altogether for comparison,

[6] https://wiki.python.org/moin/TkInter

Parameter	**Tuning strategy**	**Experimented values**
Autoencoder layers	The following layer size is normally 2 to 5 times smaller than that of the previous layer. The number of layer is proportional to the original dimensions, but compress too deeply might reduce detecting performance.	(28, 15, 5)
Weights initialisation	Following Xavier's initialisation method [32]. Boundary $L = 4 \times \sqrt{\frac{6}{\text{size-in} \times \text{size-out}}}$ where size-in and size-out are sizes of the following and previous layers.	Uniformly drawn $U \sim (-L, L)$
Batch size (n)	Choose according to the size of the data and whether GPU or CPU is used.	128 for CPU and 1024 for GPU
RFF standard deviation (σ)	Choose the best values from the experiments on the testing datasets that maximise the performances.	3.0
Number of RFF features	Choose minimum value that gives a reasonable performance. From the experiments, a size of 100 to 1000 is sufficient for encoded size <200.	100
Trade-off parameter (α)	Monitor the decrease of reconstruction loss and SVM loss and choose trade-off parameter that balances the two.	10^4
SVM parameter (ν)	Run through a range of values and select the value based on a certain threshold	$0.0 \rightarrow 1.0$
Learning rate algorithm and initial rate	Choose between Adam [10] and AdaDelta [12], where Adam gives better overall performance but AdaDelta is more stable. Initial learning rate can be higher for larger-scaled data.	Adam (0.001)
Number of epochs	Apply early-stopping and terminate training when the change in loss function reaches a certain tolerance value.	Tolerance 10^{-7}

Table 5.4: Tuning strategy for all configurable hyperparameters in the model.



Resource type: Storage Bucket
Request IP: IP Address 31
Operation: PLACEMENT
Request Country: in
Resource Zone: southamerica-east1-c
Hour: 21 Minute: 22 Second: 50

```
{
  "Input": {
    "Operation": "PLACEMENT",
    "Resource": "Storage Bucket",
    "Resource Zone": "southamerica-east1-c",
    "Request IP": "IP Address 31",
    "Request Country": "in",
    "Time": "21:22:50"
  },
  "Output": {
    "Decision": "Anomalous",
    "Decision function value": -0.01445,
    "Contribution scores": {
      "Feature 1": {
        "Value": 0.09026,
        "Percentage": 25.36,
        "Rank": 1,
        "Attribute": "Resource"
      },
      "Feature 2": {
        "Value": -0.04413,
        "Percentage": -12.4,
        "Rank": 27,
        "Attribute": "Resource"
      },
      "Feature 3": {
        "Value": -0.03152,
        "Percentage": -8.85,
        "Rank": 25,
        "Attribute": "Request IP"
      },
      "Feature 4": {
        "Value": -0.03668,
        "Percentage": -10.3,
        "Rank": 26,
        "Attribute": "Request IP"
      },
      "Feature 5": {
        "Value": -0.00502,
        "Percentage": -1.41,
```

Figure 5.5: Screenshot of the graphical interface.

with the feature selected by the list box in the middle being highlighted, while the one in the bottom-right corner is the histogram of the selected feature based on the training data. The highlighted horizontal line in the histogram indicates the current visible value. Using those information, and by experimenting modifying certain fields, the users can find out the anomalous pattern and the suitable way to adapt the policies to cover those possible breaches.

6 Progress Review and Discussion

	Time	**Task**
First half	27/09/17 - 08/10/17	Study the fundamentals of Machine learning.
	09/10/17 - 22/10/17	Develop theoretical models and algorithms.
	23/10/17 - 29/10/17	Investigate frameworks and libraries.
	30/10/17 - 17/12/17	Initial implementation and experiments.
	18/12/17 - 21/01/18	Start write-ups.
Second half	12/01/18 - 04/02/18	Research on explanation methods.
	12/01/18 - 24/02/18	Perform more experiments and validations.
	25/02/18 - 04/03/18	Draw most conclusions of the project.
	29/02/18 - 20/03/18	Prepare for Oral examination and Project showcase.
	05/03/18 - 18/03/18	Final report draft.
	21/03/18	Attend Project showcase.
	19/03/18 - 15/04/18	Final report revision and correction.
	16/04/18	Final report submission.

Table 6.1: Scheduled plan for the project, being revised after the first half period.

Table 6.1 is the scheduled plan for the entire project after being revised in the interim report. Overall, all tasks have been fulfilled. The followings are the summary of the progress during the first half of the project (before 2018):

- Basic knowledge of ML, deep learning, and Python libraries is gained.
- The cloud dataset is analysed, and possible methods to process the data are proposed. Scripts to extract the data from the database were also completed.
- Clarified the requirements from the industrial perspective.
- Proposed the novel approach of Autoencoder with AE-1SVM to resolve the defined problem, and implemented the model in TF.
- Conducted initial experiments for: 1) Proving a few potential advantages of the approach; 2) Visualise the data in an 2D encoded feature space; 3) Getting familiar with the framework.

In the second half of the project, apart from presentations preparation (oral examination, project showcase, and final report), there were two main objectives:

- Researching on methods to explain the results from the ML model.
- Performing and reporting all experiments to validate the correctness and effectiveness of the model.

Both those goals can be considered complete. Algorithms based on state-of-the-art gradient methods have been discussed in Sections 4.3 and 4.4, while validation through experiments has been provided in Section 5.3. On the other hand, thorough experiments with detailed setup and results are presented in Section 5.2. The outcomes of all experiments suggest useful information and successfully verify the proposed models and algorithms. Additionally, a graphical interface has also been implemented, which is an extra achievement for the project.

Reflecting on the specifications of the project, all requirements and expected outcomes have been met. Moreover, to the best of the student's knowledge, the approach applied in the project is novel and has not been proposed before in research literature. On top of that, the solutions in the project aim to tackle the problems in big data and scalable ML, which are realistic and critical for a wide range of applications in the industrial and/or real-world circumstances, not limited to the the cloud service scope of this project.

7 Extensions and Future works

Potential extensions and future works to this project are as follows:

- Applying Recurrent neural networks, especially Long Short-term memory (LSTM) cells to account for time-series patterns. For instance, if a normal pattern contains action series $\mathsf{A} \rightarrow \mathsf{B} \rightarrow \mathsf{C} \rightarrow \mathsf{D}$, and the model observe $\mathsf{A} \rightarrow \mathsf{B} \rightarrow \mathsf{E} \rightarrow \mathsf{F}$ which the LSTM cell indicates to have low probability, it should detect this as an anomalous pattern. The processing of data and model structure of this approach can be significantly different from the approach presented in this report and will require research effort. A more detailed introduction of this technique can be found in Christopher Olah's blog [7].

- Using neural networks as approximators to measure the contribution of input features. This is driven by the nomogram approach in [33], where the margin function $g(x)$ is decomposed into $g(x) = \sum_i g_i(x_i)$ and the contribution is represented by plotting the margin as a function of each input. Since the proposed model consists of an autoencoder structure, the same approach cannot be applied directly. However, as neural networks are universal approximators [34], if sufficient data are available, multiple deep neural networks can be created and trained to capture the linearly decomposable relationship between the margin and the input dimensions.

  A small experiment is also conducted during the time of this project to investigate this idea, as shown in Figure 7.1. The graph on the left plots a 2D squared dataset and its OCSVM boundary. The two other graphs plots the two linearly-decomposed components of the margin against two input features. Sensible semantics can be observed, as the values of those functions are higher when the point is closer to the centre. This approach still to some extend lacks justification and might require a lot more data to work, but has the advantage that it approximates the direct relationship between the input and the margin, which the gradient-based methods cannot capture.

[7]http://colah.github.io/posts/2015-08-Understanding-LSTMs/

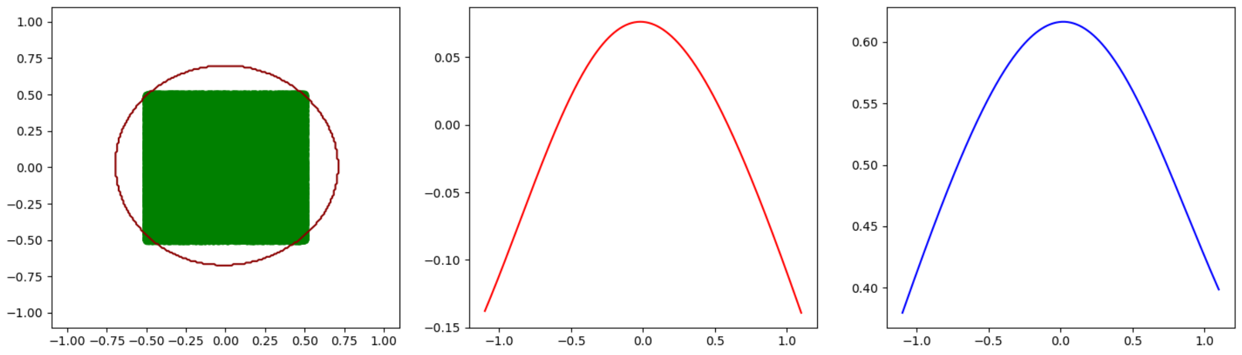


Figure 7.1: Example experiment of the linearly-decomposable approximators approach.

- In terms of implementation, some additions can also be made to the ML system. Firstly, online training that continuously adapt the model to new data can be considered. This has not been implemented in the scope of this project due to the difficulty in collection of new and continuous data. Secondly, when the ML system is decided to be put into the real cloud platform systems, the models should be served through APIs using TF Serving, as described in Section 3.1.

8 Conclusion

All things considered, the objectives and requirements of the project has been accomplished. A machine learning solution for detecting anomalous patterns and breaches in large-scale, big data cloud access as well as interpreting the decision making has been proposed, developed, and successfully validated. The outcomes of the project has contributed to both academic research, in the form of a novel approach to enhance the existing techniques in OCSVM, and industrial context presented by the implementation and application on the cloud dataset, alongside with extensive descriptions on procedures to integrate the machine learning system. This final report has outlined comprehensively all important literature with related background theory, the proposed solutions with detailed mathematical derivations, as well as experiment setups, results, and analyses. A review on the progress reflecting on the plan and a brief survey on the potential extensions and future works based on this project have also been given in the closing sections.

References

[1] Mitchell, T.M.: *Machine Learning*, (McGraw-Hill, Inc., 1997).

[2] Bishop, C.M.: *Pattern Recognition and Machine Learning (Information Science and Statistics)*, (Springer-Verlag New York, Inc., 2006).

[3] Grubbs, F.E.: 'Procedures for detecting outlying observations in samples'. *Technometrics*, **11** (1), 1969, pp. 1–21.

[4] Goldstein, M., Uchida, S.: 'A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data'. *PLOS ONE*, **11** (4), 2016, pp. 1–31.

[5] Zhou, C., Paffenroth, R.C.: 'Anomaly Detection with Robust Deep Autoencoders'. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (New York, NY, USA: ACM, 2017), KDD '17, pp. 665–674.

[6] Liu, F.T., Ting, K.M., Zhou, Z.H.: 'Isolation Forest'. *2008 Eighth IEEE International Conference on Data Mining* (2008), pp. 413–422.

[7] Schölkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J., Platt, J.: 'Support Vector Method for Novelty Detection'. *Proceedings of the 12th International Conference on Neural Information Processing Systems*, (Cambridge, MA, USA: MIT Press, 1999), NIPS'99, pp. 582–588.

[8] Nair, V., Hinton, G.E.: 'Rectified Linear Units Improve Restricted Boltzmann Machines'. *Proceedings of the 27th International Conference on International Conference on Machine Learning*, (Omnipress, 2010), ICML'10, pp. 807–814.

[9] Bottou, L.: *Large-Scale Machine Learning with Stochastic Gradient Descent*, (Heidelberg: Physica-Verlag HD, 2010). pp. 177–186.

[10] Kingma, D.P., Ba, J.: 'Adam: A Method for Stochastic Optimization'. *ArXiv e-prints*, 2014.

[11] Duchi, J., Hazan, E., Singer, Y.: 'Adaptive Subgradient Methods for Online Learning and Stochastic Optimization'. *J. Mach. Learn. Res.*, **12**, 2011, pp. 2121–2159.

[12] Zeiler, M.D.: 'ADADELTA: An Adaptive Learning Rate Method'. *ArXiv e-prints*, 2012.

[13] Harris, Z.S.: 'Distributional Structure'. *WORD*, **10** (2-3), 1954, pp. 146–162.

[14] Seo, K.K.: 'An application of one-class support vector machines in content-based image retrieval'. *Expert Systems with Applications*, **33** (2), 2007, pp. 491 – 498.

[15] Adeli, E., Wu, G., Saghafi, B., An, L., Shi, F., Shen, D.: 'Kernel-based Joint Feature Selection and Max-Margin Classification for Early Diagnosis of Parkinson's Disease'. *Scientific Reports*, **7**, 2017, p. 41069.

[16] Amer, M., Goldstein, M., Abdennadher, S.: 'Enhancing One-class Support Vector Machines for Unsupervised Anomaly Detection'. *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, (ACM, 2013), ODD '13, pp. 8–15.

[17] Chang, C.C., Lin, C.J.: 'LIBSVM: A Library for Support Vector Machines'. *ACM Trans. Intell. Syst. Technol.*, **2** (3), 2011, pp. 27:1–27:27.

[18] Williams, C.K.I., Seeger, M.: 'Using the Nyström Method to Speed Up Kernel Machines'. *Proceedings of the 13th International Conference on Neural Information Processing Systems*, (Cambridge, MA, USA: MIT Press, 2000), NIPS'00, pp. 661–667.

[19] Rahimi, A., Recht, B.: 'Random Features for Large-Scale Kernel Machines'. *Advances in Neural Information Processing Systems 20*, (Curran Associates, Inc., 2008). pp. 1177–1184.

[20] Sutherland, D.J., Schneider, J.G.: 'On the Error of Random Fourier Features'. *CoRR*, 2015.

[21] Simonyan, K., Vedaldi, A., Zisserman, A.: 'Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps'. *ArXiv e-prints*, **abs/1506.02785**, 2013.

[22] Sundararajan, M., Taly, A., Yan, Q.: 'Axiomatic Attribution for Deep Networks'. *CoRR*, **abs/1703.01365**, 2017.

[23] Oliphant, T.E.: 'Python for Scientific Computing'. *Computing in Science and Engg.*, **9** (3), 2007, pp. 10–20.

[24] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B.,

Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: 'TensorFlow: Large-scale machine learning on heterogeneous systems'. https://www.tensorflow.org/, 2015. Software available from tensorflow.org.

[25] Oliphant, T.: *Guide to NumPy*, (Trelgol Publishing, 2006).

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

[27] An, J., Cho, S.: 'Variational Autoencoder based Anomaly Detection using Reconstruction Probability' (2015).

[28] Hull, J.J.: 'A database for handwritten text recognition research'. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **16** (5), 1994, pp. 550–554.

[29] Dheeru, D., Karra Taniskidou, E.: 'UCI machine learning repository'. http://archive.ics.uci.edu/ml, 2017.

[30] Rayana, S.: 'ODDS library'. http://odds.cs.stonybrook.edu, 2016.

[31] Davis, J., Goadrich, M.: 'The Relationship Between Precision-Recall and ROC Curves'. *Proceedings of the 23rd International Conference on Machine Learning*, (New York, NY, USA: ACM, 2006), ICML '06, pp. 233–240.

[32] Glorot, X., Bengio, Y.: 'Understanding the difficulty of training deep feedforward neural networks'. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, (Chia Laguna Resort, Sardinia, Italy: PMLR, 2010), volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256.

[33] Cho, B.H., Yu, H., Lee, J., Chee, Y.J., Kim, I.Y., Kim, S.I.: 'Nonlinear Support Vector Machine Visualization for Risk Factor Analysis Using Nomograms and Localized Radial Basis Function Kernels'. *IEEE Transactions on Information Technology in Biomedicine*, **12** (2), 2008, pp. 247–256.

[34] Hornik, K., Stinchcombe, M., White, H.: 'Multilayer feedforward networks are universal approximators'. *Neural Networks*, **2** (5), 1989, pp. 359 – 366.