School of Electronic Engineering
and Computer Science

Final Report

Programme of study:
Digital and Technology Solutions
(Software Engineer)

Project Title:
Machine Learning for Anomaly Detection in Financial Data

Supervisor:
Dr Anthony Constantinou

Student Name:
Martin Sedláček

Final Year
Undergraduate Project 2021/22

Queen Mary
University of London

Date: 4th May 2022

Abstract

Recent advancements in machine learning, especially in deep generative models, have enabled new unsupervised anomaly detection methods to achieve great results. However, there have been very few attempts at implementing these models to solve real-world problems. One area that has seen a surprisingly small adoption rate of productionised machine learning systems despite access to large amounts of data is finance. This project focuses on the detection of market anomalies, which is a generally unsolved problem with a high potential impact in this domain.

This work aims to concisely explain market anomaly detection formulated as a machine learning problem. Formally, this problem can be viewed as unsupervised anomaly detection in structured multivariate time-series within a constrained fully observable finite parameter environment. A low dimensional proprietary dataset of real-world data is created to reproduce a simple version of this environment for experimentation.

It is established in theory and further backed by empirical evidence from experiments, that a significant limitation of many models is working under the assumption that the training data is largely non-anomalous. In the context of real-world big data, this assumption is rarely true and hard to verify. Mechanisms to deal with outliers in the training data such as RSR layers have shown promising results on more volatile time-series presumed to contain larger portions of anomalous data points. Further experimental results also show that reconstruction-based models aiming to capture the entire underlying distribution of the data are consistently outperformed by simpler lightweight architecture that captures data on the distribution boundary. Notable limitations of current models are identified in the inability to learn group dependent behaviour, operating under false assumptions of full observability, and as such not accounting for uncertainty in the data.

The research has concluded that recent generative anomaly detection models significantly outperform traditional statistical and density-based methods in a smaller dataset representation of a complex real-world environment. However, attempting an application in a productionised system would require significant human expertise in modelling a constrained operational environment. As such, it is concluded that the current models are not yet fully capable of reliably solving the market anomaly detection problem in the broader environment of financial markets.

Keywords *– machine learning, unsupervised learning, anomaly detection, multivariate time-series, financial market anomalies*

contents

List of tables

List of figures

Nomenclature

Acronyms / Abbreviations

AD – Anomaly detection

AE- Auto-Encoder

CLT – Central Limit Theorem

CNN – Convolutional Neural Network

DL – Deep Learning

DNN – Deep Neural Network

EM – Excess-Mass

GAN - Generative Adversarial Network

HBOS - Histogram-Based Outlier Score

LOF – Local Outlier Factor

LSTM – Long-Short Term Memory

ML – Machine Learning

MLP – Multi-Layer Perceptron

MV – Mass-Volume

NN - Neural Network

OC-SVM – One Class Support Vector Machine

PCA – Principal Component Analysis

PR – Precision-Recall

RL – Reinforcement Learning

RNN – Recurrent Neural Network

RSR – Robust Subspace Recovery

RSRAE - Robust Subspace Recovery Auto-Encoder

SOTA – State-Of-The-Art

STD – Standard Deviation

w.r.t – with regards to

Chapter 1: **Introduction**

1.1 Motivation

Modern world is becoming increasingly more data-driven as companies attempt to leverage the vast amounts of data available to them. Notably, financial institutions have exposure to some of the largest pools of data with millions of transactions and trades occurring each day.

Application of anomaly detection (AD) in financial data is well researched in areas of fraud detection and anti-money laundering (Ahmed, Mahmood & Islam, 2016). This is reflected in the fact that most of the productionised Machine Learning (ML) systems in finance are for compliance (Bank of England, 2019). Beyond this, the application of AD in finance from a big data perspective is widely ignored (Ahmed, Choudhury & Uddin, 2017).

One potentially high impact area of AD and the focus of this project is detection of market anomalies. According to Zhang (2006), investors tend to underreact to events when there is a higher degree of information uncertainty. Therefore, timely and reliable detection of market anomalies can substantially improve the risk management strategies and help identify opportunities for potential profit.

In theory, ML appears well suited for such a task thanks to its versatility and ability to find patterns in highly complex data structures. It has repeatedly shown an ability to outperform all rule-based and statistical systems in environments with a high degree of uncertainty (Siddique et al., 2022; Nandanwar et al., 2020). However, for a variety of reasons, time-series analysis has largely remained dominated by statistical models such as ARIMA, Holt-Winters or ETS. These models were able to achieve similar performance as most ML models with significantly lower computational costs (Makridakis, Spiliotis & Assimakopoulos, 2018).

Recently, deep neural networks (DNNs), and notably deep generative models have shown significant improvements in time-series forecasting (Lim & Zohren, 2021). This allowed for advancements in the development of unsupervised generative-based AD models that started to outperform statistical models on tasks such as network intrusion detection and anomaly detection in cyber-physical systems.

While there is known work on applying ML to basic forms of AD in financial time-series (Akyildirim et al., 2022; iSmile Technologies, 2021; Ahmed, Choudhury & Uddin, 2017), the experiments are often aimed at illustrating a particular model's capabilities, rather than accurately formulating, and solving the market anomaly detection problem. To the best knowledge of the author, application of recent state-of-the-art (SOTA) models to the market anomaly detection problem is novel.

1.2 Problem Statement

While researching the topic, the following concrete problems were identified:

- General absence of academic research concerned with application of recent AD models to real-world data.
- Most experiments are conducted on benchmarking datasets, which can often satisfy unrealistic assumptions about the nature of the anomalies in the data (Munir et al., 2019; Li et al., 2019; Liang et al., 2021).
- The models were only evaluated with criteria that require the evaluation dataset to be labelled, which is rarely the case in big data.

As such, it is hard to draw insights about the possibility of applying SOTA AD techniques to the market anomaly detection problem in a productionised environment.

1.3 Aims and Objectives

This project aims to address gaps found in previous research work by conducting novel experimental research on applying the latest SOTA models to the market anomaly detection problem within the constraints of real-world financial data.

This aim translates to the following individual objectives:

- Model a dataset of relevant financial information with traits resembling real-world market streaming data (chapter 4).
- Determine a suitable performance evaluation criterion and method that does not require the evaluation dataset to be labelled (section 5.5).
- Implement and train SOTA AD methods (chapter 5)
- Provide a detailed comparative evaluation of model capabilities on the market anomaly detection problem (chapter 6).
- Evaluate how well the originally presented capabilities of the models extrapolate to this type of environment (chapter 6).
- Identify gaps, misassumptions, and limitations (chapters 4 and 7).
- Propose solutions and potential improvements to push forward the adoption of ML in production-grade systems (chapter 7).

Chapter 2: **Preliminaries**

2.1 Reader Profile

The purpose of this section is to establish what concepts the reader is assumed to be familiar with prior to reading this report. These are as follows

- Basic Statistics and Probability (conditional probability, Bayes rule, random variables, CLT, Law of Large Numbers, correlation, etc.)
- Basic Linear Algebra and Multivariate Calculus (linear transformations, vector/matrix multiplication, differentiation, integration, chain rule etc.)
- High-level understanding of financial markets (e.g., equities, volatility)

Additionally, knowledge of the following topics is helpful but not required

- Basic Set Theory and Real analysis, particularly Measure Theory
- Experience with training feedforward Neural Networks

Formal knowledge of AD, ML, or any other niche domain knowledge is not expected, and all relevant concepts are explained.

2.2 Anomalies

An anomaly can be classified as “a data point that is significantly different from the remaining data” (Aggarwal, 2016:p.1). *Anomalies* are also commonly referred to as *outliers* and the two terms can be treated as synonyms in the context of this report.

2.2.1 Market Anomaly

In financial markets, anomalies are characterised as “movements or events which cannot be explained by using efficient market hypothesis” (Latif et al., 2011:p.3).

They can be divided into three basic categories:

1. *Fundamental* – mispricing and other value anomalies.
2. *Technical* – originating from inefficiencies in technical analysis.
3. *Calendar or seasonal* – related to particular periods, such as the release of an earnings report.

By nature, technical anomalies are not directly reflected in the raw data, therefore the term *market anomaly* will only refer to fundamental and seasonal anomalies.

Note the objective of the AD methods in this project is to maximise the reliability of spotting that a certain data point is anomalous by either definition, rather than finding the root cause.

2.3 Machine Learning

This section is aimed at giving a brief and concise overview of fundamental concepts in ML necessary for navigating later sections of this report. Loosely, ML can be seen as a branch of algorithms that can continually improve by learning from data.

It is further classified into supervised, unsupervised, semi-supervised, and reinforcement paradigms based on the type of data and environment. There is a variety of algorithms and architectures that differ in how they facilitate the learning aspect.

2.3.1 Normal Distribution

Perhaps the most well-known and important distribution is the Normal (Gaussian) distribution, which is a continuous probability distribution following a bell curve shape.

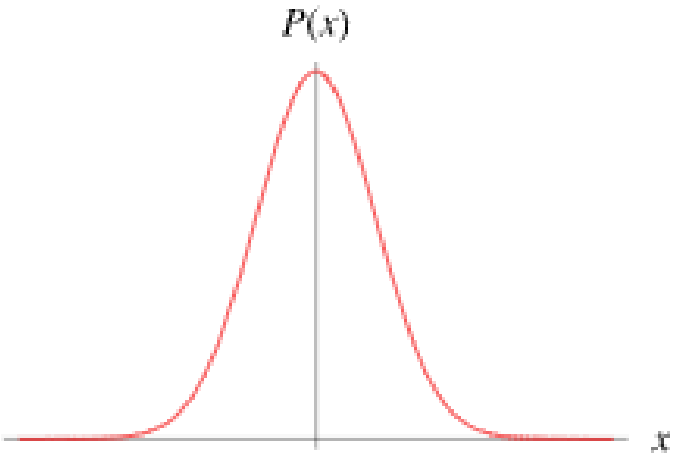


Figure 1. shape of the normal distribution (Wolfram MathWorld, n.d.)

This concept is central to many topics in ML and statistics, particularly linear models. It has been shown that normally distributed data is easier for most ML models to learn from thanks to its desirable statistical properties (Zhang et al., 2020:chap.18.6; Sirignano & Spiliopoulos, 2020; Do, 2019).

2.3.2 Learning Associations

ML models effectively aim to discover relations between variables in the dataset. In other words, they try finding the conditional probability $P(X \mid D)$, where X is some desired output and D is a set of attributes (features) of the input data, referred to as a feature vector.

2.3.3 Classification

Classification is a problem concerned with identifying a set of categories (*classes*) the input data belongs to, based on its feature vector (Fig. 2). In the context of this project, anomaly detection can be thought of as a binary classification problem. This implies two classes, normal and anomalous, commonly represented numerically as 0 and 1.

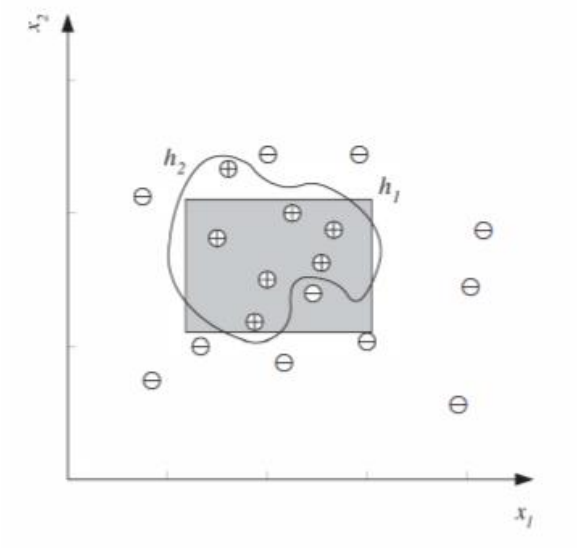


Figure 2. classification in 2D space (Alpaydin, 2014:p.31)

2.3.4 Regression

Generally, regression problems take a feature vector $\boldsymbol{x}$ as an input and produce some numerical value as an output. This is made possible by the algorithm fitting a function f to the data, where $f(\boldsymbol{x})$ best approximates the target numerical value $\boldsymbol{y}$.

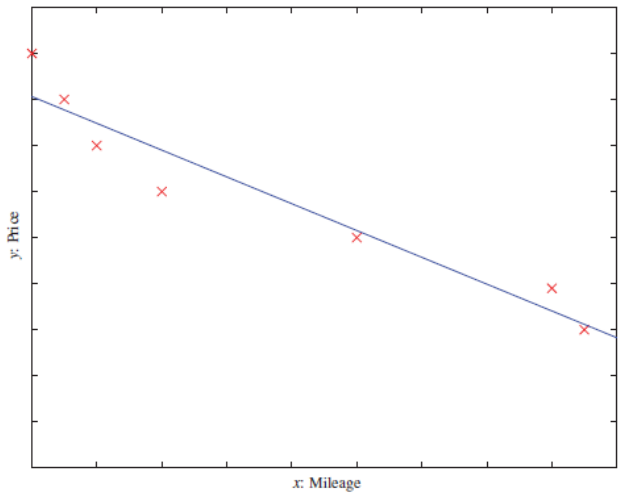


Figure 3. estimating car prices with linear regression (Alpaydin, 2014:p.10)

Linear regression (Fig. 3) is one of the simplest and most fundamental building blocks of a Neural Network (NN). Assuming input $\boldsymbol{x}$, linear regression can be expressed as the linear transformation $\boldsymbol{y} = \boldsymbol{x}A^T + b$, where A represents a learnable vector of weights of the same size as $\boldsymbol{x}$ and b is the bias – a learnable scalar value.

2.3.5 Norm, Loss, and Loss Function

The way a model composed of just one linear regression layer would facilitate learning the association between mileage and car price in Fig. 3 is by optimising A and b to maximise $P(y \mid \boldsymbol{x})$ for known pairs of $\boldsymbol{x}, \boldsymbol{y}$ values. Note that the correctness of a prediction $\boldsymbol{y}$ can be measured in different ways. These measurements are known as *norms* and can also be loosely thought of as basic *loss functions* for training a model. The measured scalar of a loss function value is referred to as the *loss*.

2.4 Gradient-Based Optimisation

The idea of ML models learning associations from the data and the role of learnable parameters was introduced in 2.3.2 and 2.3.4 respectively. This section is focused on giving a brief overview of how these parameters can be learned (*optimised*). The most widespread approach to learning is through gradient-based optimisation. For the sake of brevity, primarily intuitive explanations will be used to illustrate these concepts.

2.4.1 Gradient

The gradient can be defined as a vector of partial derivatives of a function output *w.r.t*. the function inputs and denoted by the *nabla* symbol (∇) (1).

$$\nabla f(x_1, .., x_n) = \left[\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}\right] \tag{1}$$

This can be intuitively interpreted as a quantification of the relationship between the change in input to the change in output. The resulting structure is a vector field whose value at any point is a vector representing the direction and rate of fastest increase as given by the computed partial derivatives for the given input values.

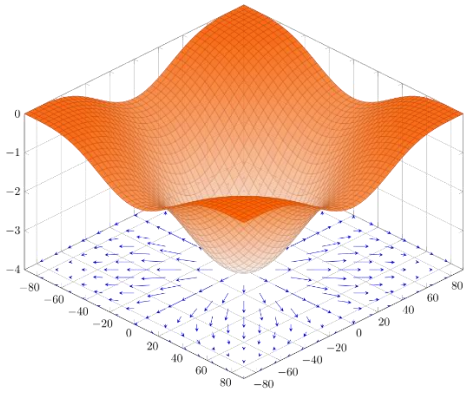


Figure 4. visualising the gradient of a bivariate function (Thoma, 2018)

Fig. 4 illustrates how the gradient of a bivariate function with a single output is visualised in 3D vector space. Higher-dimensional gradients are harder to depict. One might notice that obtaining the gradient would be particularly valuable for optimisation problems aiming to maximise the output of a function.

2.4.2 Backpropagation

Most ML architectures can be interpreted as a sequence of linear transformations on the input data. Therefore, a *gradient* can be computed. Backpropagation refers to the process of obtaining the gradient of such a fully differentiable ML model by propagating information 'backwards' through the model via chain rule to compute the influence of model inputs on the target output value (i.e., the *loss,* briefly introduced in 2.3.5).

2.4.3 Optimisation

The gradient is used in optimisation theory for finding the local maxima of a multivariate function. However, in the context of ML, the model is not trying to maximise the output but to minimise loss. Therefore, the gradient is computed on the loss *w.r.t*. the model inputs, rather than on the raw output. Once the gradient is obtained, there is a range of optimisation algorithms that can be used to find the local minima (i.e., minimise the model loss/error). Notable examples of gradient optimizers include Stochastic Gradient Descent (SGD), Adam, and AdaGrad (pytorch, n.d.).

2.5 Unsupervised Learning

While supervised learning requires the data to be labelled as belonging to a particular class, unsupervised learning does not need these labels. The learning process identifies a structure to the input space, where some patterns are naturally more common than others (Alpaydin, 2014). In statistics, this is known as *density estimation*.

2.5.1 Nearest-Neighbour based techniques

These techniques assume that "normal data instances occur in dense neighbourhoods, while anomalies occur far from their closest neighbors" (Chandola, Banerjee & Kumar, 2009:p.25). Based on this, they use the *distance* between data points to measure *similarity*.

2.5.2 Clustering techniques

One of the most widely used approaches to unsupervised learning is clustering. Same as nearest-neighbour techniques, data points are grouped into clusters based on some measurement of distance representing feature similarity.

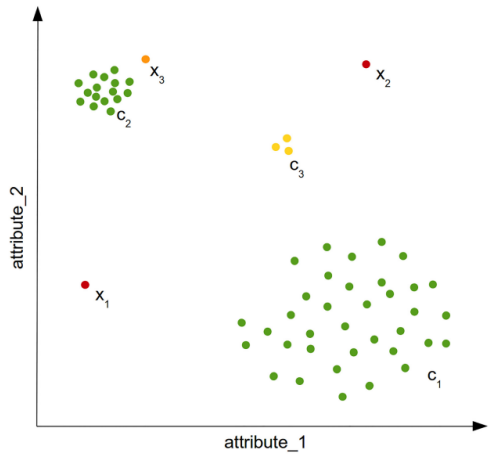


Figure 5. clustering in 2D space - clusters (C_1, C_2), global anomalies (X_1, X_2), local anomaly X_3, and micro-cluster C_3 (Goldstein & Uchida, 2016:p.5)

Fig. 5 illustrates how clustering identifies anomalies on arbitrary 2D data. Aggarwal's (2016) definition of an anomaly as “a data point that is significantly different from the remaining data” (p.1) is the same in notion as the anomalous points X_n. These points lie outside the clusters (i.e., the remaining data), displaying the overall fitness of clustering-based algorithms for general AD tasks.

2.5.3 Bayesian methods and uncertainty

The case of micro-clusters in Fig. 5, such as C_3, provides another layer of complexity to anomaly detection. Goldstein & Uchida (2016) argue that these data points should not be binarily labelled, but rather assigned some form of a score, representing the likelihood that a given data point is an anomaly, as it is not possible to say with certainty whether the C_3 data points are anomalies. This issue is often addressed by using various Bayesian methods, which generally aim to quantify uncertainty in some way. The Bayes rule is given by (2)

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2}$$

where A and B are events, $P(B)$ is a non-zero probability of event B, and $P(B|A)$ is the conditional probability of event B given event A.

Applying the Bayes rule directly in large datasets is costly. Therefore, most of the methods used in ML are *approximate Bayesian methods,* such as Laplace's approximation, and expectation propagation (Ghahramani, 2004).

2.6 Neural Networks

2.6.1 Deep Learning

Deep Learning (DL) refers to the use of multiple layers in a NN representing multiple layers of abstraction. DL is not a separate paradigm, and the concepts apply to supervised, unsupervised, and reinforcement learning alike. It is based on representation-learning**,** which is “a set of methods that allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification” (Lecun, Bengio & Hinton, 2015:p.1). Multiple levels of representation are obtained by composing non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, more abstract level. With a composition of enough transformations, very complex functions can be learned (Lecun, Bengio & Hinton, 2015).

2.6.2 Multi-layer perceptron (MLP)

MLPs are extensions of basic linear models such as linear regression. They introduce intermediary hidden layers between the input and output layers. In the simplest configuration, this is a single fully connected layer with an arbitrary hidden dimension

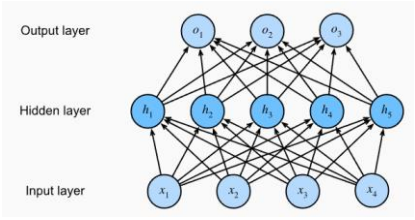


Figure 6. single hidden layer MLP (Zhang et al., 2020:fig.4.1.1)

The MLP in Fig. 6 can be thought of as the most basic and raw form of a NN. It is also commonly enhanced by applying non-linear element-wise activation functions that determine whether the unit should *fire* – i.e., forward propagate the information in a significant manner. This linear architecture is fully differentiable and therefore, can be easily trained with backpropagation and gradient-based optimisation.

2.6.3 Convolutional Neural Network (CNN)

CNNs are a family of NNs predominantly used in computer vision. The networks take their name after a mathematical procedure called convolution that expresses how the shape of one function is modified by another. This principle is applied in at least one of the layers, called the convolutional layer. Convolutional layers also leverage principles of translational invariance and locality to reduce the size of the learnable parameter. CNN models also use pooling layers to reduce the dimension of the data by combining many outputs from one layer into a single point in the next one. This is commonly an average or maximum of small clusters (e.g., 2x2) (Zhang et al., 2020).

2.6.4 Recurrent Neural Network (RNN)

RNNs are a subclass of NNs best suited for sequential data such as time-series. They use state variables to remember information about the input, which allows them to be very precise in predicting the output. This added concept of memory allows the network to backpropagate through previous states of the network, making it possible to account for the impact of previous time steps on the current outcome during training.

2.6.5 Long-Short Term Memory (LSTM)

LSTM extends the memory of the basic RNN, allowing it to learn from important data far back into the sequence. The LSTM architecture also introduces memory cells with three gates (Sak, Senior & Beaufays, 2014). An *input gate* determines whether to let the next input in, an *output gate* computes output and feeds forward, and a *forget gate* adaptively resets the cell's memory. The LSTM is a crucial building block for SOTA DNNs used for multivariate time-series analysis introduced later in chapter 3.

2.6.6 Generative Adversarial Network (GAN)

GAN is composed of two models which are trained simultaneously

- **Generative model (G)** captures the data distribution
- **Discriminative model (D)** estimates the probability that a sample came from the training data rather than **G**.

"The training procedure for G is to maximize the probability of D making a mistake. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to 1/2 everywhere" (Goodfellow et al., 2014:p.1). When G and D are defined by MLPs or other fully differentiable architectures, the entire system can be trained with backpropagation (Goodfellow et al., 2014).

2.6.7 Auto-Encoder (AE)

AEs are a subclass of NNs well suited for classification tasks (2.3.3). The architecture is composed of an encoder and a decoder. The goal of the encoder is to compress the into a typically lower latent dimension and extract meaningful patterns. The role of the decoder is to attempt to reconstruct the original data from the encoder output in the latent dimension. Similarly, to GANs, there is a form of adversarial loss used for training.

2.7 Reinforcement Learning

Reinforcement learning (RL) models aim to produce a series of optimal actions, also referred to as a policy (Alpaydin, 2014), based on inputs from the environment they operate in. The goodness of a policy is assessed through a pre-defined reward function and propagated to the learning agent. As such, RL is best suited for more demanding interactive tasks, including self-driving vehicles, distributed sensor networks, and agile robotic systems (Recht, 2019).

2.8 Tools

2.8.1 Python

Python is a high-level, interpreted, object-oriented programming language (The python foundation, n.d.). It is widely used by the ML and data science community, mainly thanks to its vast range of supported libraries and tools. Due to its simple syntax, it allows for fast prototyping and development. Another major factor is the fact that most of the available ML source code is already written in Python. These aspects make python the best option currently available for easy implementation of ML models. Anaconda is used for virtual environment and dependency management.

2.8.2 Python Libraries

This project leveraged multiple libraries to support the various data manipulations needed to support the ML algorithms. This includes NumPy, Pandas, and SciKitLearn among others. Notably, PyTorch (or simply torch) is an optimised tensor library for ML with Python. Torch greatly simplifies the implementation of ML algorithms, primarily NNs, thanks to features such as automatic differentiation, predefined activation functions, and optimizers. It supports computing on a Graphics Processing Unit (GPU) with CUDA cores, which speeds up the training of complex models thanks to increased bandwidth and parallel computation (pytorch.org, n.d.).

2.8.3 Jupyter Notebooks

Jupyter is an open-source notebook for interactive computing. It supports programming in many languages other than Python, but this project mainly leverages it as an interface that allows for executing Python code, saving states, and plotting graphs and tables. This was determined to be the best op for prototyping and later showcasing the models, allowing for the presented results to be easily reproducible.

Chapter 3: **Background: Unsupervised Anomaly Detection**

This chapter discusses a variety of relevant unsupervised AD algorithms and provides a critical summary discussing their suitability for multivariate time-series data.

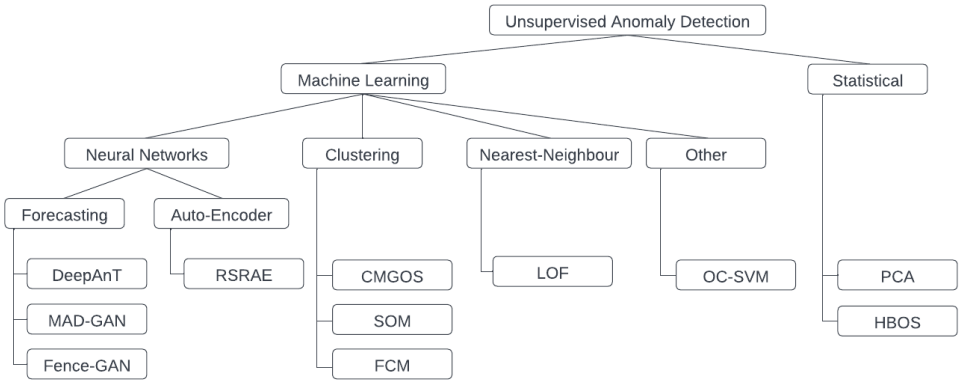


Figure 7. taxonomy of unsupervised anomaly detection models

Fig. 7 lists and categorises the relevant algorithms discussed further in this chapter. Clustering algorithms such as CMGOS, SOM, and FCM are briefly discussed for performance comparison in section 3.4 but not implemented for experimentation.

3.1 Unsupervised Models

3.1.1 Local Outlier Factor (LOF)

LOF is based on k-NN, which is a classification algorithm (see 2.3.3), where *k* nearest neighbours are used in determining the class of a data point (Cunningham & Delany, 2020). It is also a *local algorithm*, meaning that the score of a data point is determined by its isolation from the surrounding neighbours (Breunig et al., 2000). To compute the LOF score, firstly, *k* nearest neighbours are found for each element *x*. Next, using the nearest neighbour N_k and reachability distance d_k (usually Euclidian distance), the local reachability density (LRD) is estimated by (3)

$$LRD_k(x) = 1 \,/\, \frac{\sum_{o \in N_k} d_k(x, o)}{|N_k(x)|} \tag{3}$$

$$LOF(x) = \frac{\sum_{o \in N_k} \frac{LRD_k(o)}{LRD_k(x)}}{|N_k(x)|} \tag{4}$$

(Goldstein & Uchida, 2016:p.10)

The final LOF score (4) is essentially a ratio of local densities. For data points that are inside a cluster, it will be close to 1. For outliers, the score is naturally higher as their minimum distance to other data points in their local neighbourhood is much higher.

3.1.2 One-class SVM (OC-SVM)

OC-SVMs are often used for semi-supervised anomaly detection, but they are fundamentally unsupervised algorithms. They intend to detect novelty by effectively learning the area boundary of normal data by separating by learning a function that is positive for high-density areas. In the unsupervised anomaly detection scenario, each instance is scored by a normalized distance to the determined decision boundary. Some OC-SVMs have been modified such that they include robust techniques for explicitly dealing with outliers during training. The idea is that anomalies contribute less to the decision boundary than normal instances (Goldstein & Uchida, 2016).

3.2 Statistical Models

3.2.1 Principal Component Analysis (PCA)

PCA is primarily a dimensionality reduction technique. However, it can be used for anomaly detection using deviation from detected subspaces. The algorithm computes principal components as the eigenvectors of the covariance matrix. A projection of a feature vector on the retrieved eigenvectors along with a reconstruction loss are then used to assign an anomaly score (Goldstein & Uchida, 2016).

3.2.2 Robust PCA (rPCA)

The main difference from naive PCA is that rPCA computes the covariance matrix based on the Mahalanobis distance. Then, using the major components, global deviations from most of the data can be detected. Minor components can help indicate smaller local deviations (Goldstein & Uchida, 2016).

3.2.3 Histogram-Based Outlier Score (HBOS)

HBOS is a simple statistical AD algorithm that assumes independence of the features. It uses normalised sized histogram bins for density estimation. The HBOS score for each instance is then computed based on the size of the bins within distance *d* of an instance *p*, as given by (5)

$$HBOS(p) = \sum_{i=1}^{d} \log\left(\frac{1}{hist_i(p)}\right) \tag{5}$$

(Goldstein & Dengel, 2012:p.4)

As such, it has substantially lower computational complexity, taking only a few seconds to produce results on large datasets where complex ML models might take days (Ahmed, Choudhury & Uddin, 2017; Goldstein & Uchida, 2016).

3.3 Neural Networks

3.3.1 DeepAnT

Munir et al. (2019) propose a model consisting of a CNN based predictor module and a distance-based anomaly detector. This model is capable of learning periodic and seasonal anomalies while simultaneously requiring lower amounts of data to train than RNN based models. The predictor consists of a pair of convolutional layers followed by max-pooling layers and a single fully connected layer producing the prediction (Fig 8.).

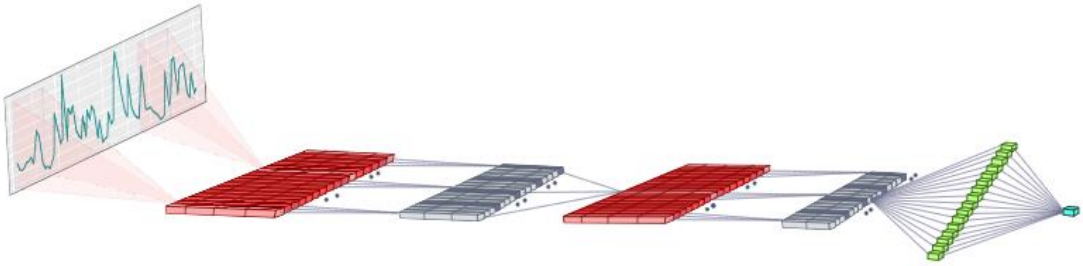


Figure 8. CNN time-series predictor layer (Munir et al., 2019:p.6)

Since the CNN is fully connected, it can be trained with a gradient-based optimizer such as SGD, which was used in the original paper (Munir et al., 2019).

Subsequently, the anomaly detector module computes the anomaly score as the Euclidian distance between the predicted and actual values. High scores naturally indicate large anomalies, while lower scores are harder to judge. A threshold needs to be defined to separate normal and anomalous scores.

3.3.2 MAD-GAN

MAD-GAN is a reconstruction-based algorithm with an LSTM-RNN GAN architecture designed to capture the distribution of a time-series. The architecture exploits both the generator and the discriminator for anomaly detection (Li et al., 2019, 2018).

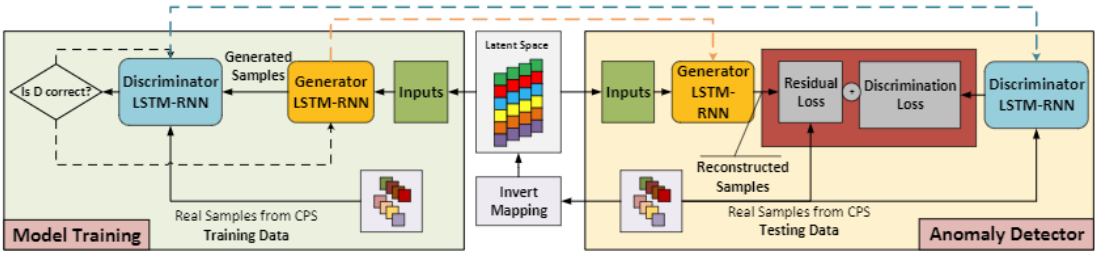


Figure 9. MAD-GAN architecture (Li et al., 2018:fig.1)

Fig. 9 on the left is a GAN framework in which the generator and discriminator are obtained with iterative adversarial training based on the LSTM architecture. The generator produces the probability distribution for the generated samples. Next, the discriminator is trained to distinguish the generated samples from real ones by minimizing negative cross-entropy loss.

Fig. 9 on the right is the exploitation of both the GAN's generator and discriminator for anomaly detection. The generator is used for computing the residual loss between reconstructed samples and real ones, while the discriminator is used to compute the discrimination loss. These two are combined to produce the final anomaly score.

This way, the discriminator is well trained to distinguish an anomalous sample from a normal one, since its purpose is to determine whether the given sample is realistic. The samples from the generator can be interpreted as if it was drawn from the real data's normal distribution since the generator's objective is to make the generated samples as similar to the real data as possible. Therefore, the difference between the generated and real sample can also be used to identify anomalies in the real data as seen in the combined residual and discrimination loss used for anomaly scoring.

3.3.3 Fence-GAN

The main strength of the Fence-GAN lies in the modified loss function for the generator. Ngo et al., (2019) propose a loss function (6) given by a weighted sum of the encirclement loss (7) and dispersion loss (8).

$$\mathcal{L}_{generator}^{FGAN}(G_\theta, D_\phi, \mathcal{Z}) = EL + \beta * DL \tag{6}$$

$$EL(G_\theta, D_\phi, \mathcal{Z}) = \frac{1}{N}\sum_{i=1}^{N}\left[\log\left(\left|\alpha - D_\phi(G_\theta(\mathcal{Z}_i))\right|\right)\right] \tag{7}$$

$$DL(G_\theta, \mathcal{Z}) = \frac{1}{\frac{1}{n}\sum_{i=1}^{N}(\|G_\theta(\mathcal{Z}_i) - \mu\|_2)} \tag{8}$$

(Ngo et al., 2019:pp.3–4)

where G_θ is the trained generator, D_ϕ is the trained discriminator, $\mathcal{Z}$ is sampled from latent space, $\alpha \in (0,1)$ is a tuneable hyperparameter used to generate points on the boundary of the distribution, and β is the weight of dispersion loss.

This way, the objective of the generator is changed to capture the boundary of the data distribution, rather than regenerate points from the entire original distribution (Ngo et al., 2019). The discriminator is trained using the average of weighted sums of Binary Cross-Entropy (BCE) losses from the real and generated samples as given by (9)

$$\mathcal{L}_{discriminator}^{FGAN}(G_\theta, D_\phi, X, \mathcal{Z}) = \frac{1}{N}\sum_{i=1}^{N}\left[-\log\left(D_\phi(x_i)\right) - \gamma\log\left(1 - D_\phi(G_\theta(z_i))\right)\right] \tag{9}$$

(Ngo et al., 2019:p.4)

These modifications allowed the Fence-GAN to achieve outstanding results in initial tests and beat known SOTA models at the time. Only the discriminator is needed for anomaly detection as it was trained to distinguish original (non-anomalous) and artificially generated samples. In the case of time-series, this model can be trained with relatively simplistic MLP-based GAN architecture for both generator and discriminator.

3.3.4 Robust Subspace Recovery Auto-Encoder (RSRAE)

This model introduces an RSR layer within AE (see 2.6.7). According to Lai, Zou & Lerman (2019), it leverages "a special regularizer, that enforces an outliers-robust linear structure in the embedding obtained by the encoder" (p.3).

The RSR layer can be thought of as a layer in the NN, which applies outlier-robust dimensionality reduction. An input $X \in \mathbb{R}^D$ is mapped into a lower dimensional latent space $Z \in \mathbb{R}^d$ via a linear transformation embedded in the RSR layer. This operation is a simple matrix-vector multiplication of X with a learnable parameter $\boldsymbol{A} \in \mathbb{R}^{d \times D}$ which is optimised via minimising a loss function (10), given as a sum of (11) and (12)

$$L_{RSRAE}(\mathcal{E}, \boldsymbol{A}, \mathcal{D}) = L_{AE}^{1}(\mathcal{E}, \boldsymbol{A}, \mathcal{D}) + L_{RSR}^{1}(\boldsymbol{A}) \tag{10}$$

(Lai, Zou & Lerman, 2019:p.3)

Assuming input data $\{x^{(t)}\}_{t=1}^{N}$ the loss functions for AE (11) and the RSR layer (12) are given as

$$L_{AE}^{p}(\mathcal{E}, \boldsymbol{A}, \mathcal{D}) = \sum_{t=1}^{N} \left\| x^{(t)} - \hat{x}^{(t)} \right\|_2^p \; for \; p > 0 \tag{11}$$

$$L_{RSR}^{q}(\boldsymbol{A}) = \lambda_1 \sum_{t=1}^{N} \left\| \boldsymbol{z}^{(t)} - \boldsymbol{A}^T \boldsymbol{A} \boldsymbol{z}^{(t)} \right\|_2^q + \lambda_2 \| \boldsymbol{A}\boldsymbol{A}^T - \boldsymbol{I}_d \|_F^2 \tag{12}$$

(Lai, Zou & Lerman, 2019:pp.3–4)

where $\boldsymbol{z}^{(t)}$ is the output of encoder $\mathcal{E}$ given as $\boldsymbol{z}^{(t)} = \mathcal{E}(x^{(t)})$, $\mathcal{D}$ is the decoder mapping $\hat{x}^{(t)} = \mathcal{D}(\boldsymbol{z}^{(t)})$, $\boldsymbol{I}_d$ denotes an identity matrix, λ_1, $\lambda_2 > 0$ are tuneable hyperparameters, and q is the power of the deviation (preferably $q = 1$).

This allows the RSRAE to operate well in unsupervised settings where there is uncertainty about the degree of anomalous behaviour in the training data.

The RSR layer itself is not restricted to AE, but in the context of this project, only the RSRAE implementation is used. Lai, Zou & Lerman (2019) have established initial working experiments with GAN architectures as well.

3.4 Critical Analysis and Summary

Campos et al. (2016) conclude that OC-SVM and SOM have consistently performed worse than LOF. However, LOF was shown to be highly prone to false positives in datasets with global anomalies. HBOS, rPCA, and CMGOS all performed better than LOF on average in the respective order (Goldstein & Uchida, 2016). Additionally, OC-SVM, SOM, and CMGOS are all very computationally expensive compared to LOF or the purely statistical HBOS.

Li et al. (2021) propose an extended FCM clustering algorithm which uses a sliding window approach with FCM enhanced by an extended Euclidian distance function and PSO reconstruction. This method achieved strong results on multiple datasets,

outperforming all enhanced PCA based methods and statistical models. Like other clustering-based algorithms, the main drawback of this method is its large computational cost. It also significantly depends on the cluster centres revealing a structure in the data. If the structure within the multivariate data cannot be revealed, the performance of anomaly scoring will deteriorate significantly.

Majority of the unsupervised DNN methods rely on some form of accurate forecasting. This allows them to incorporate seasonal and periodic anomalies into consideration, which was previously not possible with clustering and statistical models. The approaches to forecasting vary significantly by architecture. For example, Munir et al. (2019) and Li et al. (2021) leveraged CNNs, while other models rely on RNNs for time-series prediction. Many recent models incorporate a form of adversarial training loss (Inoue et al., 2017; Li et al., 2019; Liang et al., 2021).

Inoue et al. (2017) have used an LSTM based DNN architecture and a probabilistic outlier detector for anomaly detection in data collected from cyber-physical secure water treatment system (SWaT), outperforming an OC-SVM. Working with similar multivariate time-series data from a SWaT, Li et al. (2019) propose the MAD-GAN architecture. MAD-GAN uses an LSTM based model for both the generator and discriminator. This method has shown promising results, severely outperforming PCA-based approaches (Li et al., 2019).

Fence-GAN has achieved great results on multiple datasets with a lightweight MLP-based GAN architecture by modifying the objective of the loss function. This is a different approach to the previous forecasting-based anomaly detection models that focused on generating samples directly close to the underlying distribution of the data.

RSRAE attempts to address a significant limitation of MAD-GAN, Fence-GAN and DeepAnT, which all expect the training data to be predominantly normal. The introduction of an RSR layer to an AE model allows for unsupervised outlier removal from the training data. The model is still fundamentally based on making accurate normal predictions but is much more flexible in application under uncertain conditions. It should be noted that OC-SVM can be trained on data with anomalous samples but suffers from other significant limitations mentioned earlier in section 3.6.

Deep learning architectures are currently highly computationally expensive by design and this cost is only going to grow in the future (Thompson et al., 2020; Hu et al., 2021). Therefore, the DNNs must be able to outperform linear models by a significant enough margin to justify added complexity.

3.5 Related Work

Ahmed, Choudhury & Uddin (2017) apply LOF, rPCA, and CMGOS to a dataset of financial transactions. They conclude that clustering algorithms are more suitable than semi-supervised SVMs and statistical approaches, however, the work excludes NNs. Akyildirim et al. (2022) apply a novel signature method to a dataset of cryptocurrency transactions, concluding that it is capable of similar performances as supervised methods. Comparison with common baselines is not provided. Lastly, there is a variety of non-academic articles on the application of NNs to financial data, with sequence prediction being the most popular example (iSmile Technologies, 2021). However, these are often simple tests on univariate data that serve more as a showcase of the capabilities of sequential models such as LSTM. Application of SOTA anomaly detection algorithms to real-world datasets is generally scarce and predominantly done in the networks and cyber-physical systems domain (Inoue et al., 2017; Li et al., 2019; Siddique et al., 2022).

Chapter 4: **Formulating Market Anomaly Detection as a Machine Learning Problem**

4.1 Defining the Environment

The financial market is a complex environment with many moving parts. Any remotely significant event taking place around the world contributes to changing the state of the environment to some extent (Harper, 2021).

Therefore, the number of parameters is effectively infinite. Intuitively, no human or machine can hold complete information about such state - meaning the model has only partial observability of the environment. Further borrowing from RL terminology, this environment be defined as an infinite partially observable model (Doshi-Velez, 2009; Hassan Mahmud, 2010) and cannot be defined *a priori*.

While traditional ML models do assume the environment model cannot be defined, they derive inferences under the assumption of full observability in a finite parameter state space. Intuitively, everything the model was ever exposed to is contained within a finite-sized training dataset to which the model has full access. One option to deal with this constraint is using an RL agent designed for operation in this type of environment. However, this is currently not a viable option as discussed later in section 7.1.3.

Another way is simply constraining the parameter space to a finite number of observable metrics. This, of course, brings some issues. Firstly, significant human domain expertise is needed and the now hidden forces altering the state of the environment remain in place. This can intuitively be seen as performing naive dimensionality reduction on the state parameters. A human trader is effectively doing the same - i.e., choosing a finite number of market signals that best indicate the state of the market *w.r.t* an asset. Additionally, this environment also introduces a level of uncertainty about the model predictions. This is implicit in an unsupervised setting (Mathieu & Lecun, 2017).

Unless a model is fitted with mechanisms to address the uncertainty and complexity of the environment, it will naturally falsely assume complete independence of the observable state from the hidden state. Only one example of using such mechanisms in unsupervised AD is known to the author (Yong & Brintrup, 2022), which was released mid-way through this project and excluded from experimentation. Note that none of the recent SOTA models seen in chapter 3 apply mechanisms for quantification of uncertainty. Loosely speaking, such models should be considered naive to an extent. This is important to realise if one is to consider relying on their predictions in production.

4.2 Choice of Metrics

Picking which of the state parameters to track is challenging and there are entire businesses built on providing such data in large quantities (Bloomberg, n.d.). This project has created an aggregated dataset that captures several of the relevant market signals for equities picked by human experts and sourced them from freely accessible datasets on Kaggle (David, 2018; Marjanovic, 2017; Aaron7sun, 2019).

Date	Ticker	Open	High	Low	Close	Volume	Sentiment	SPX_Close
2008-09-02	artna.us	13.7470	13.7470	13.6370	13.6370	1435	0	1277.58
2008-09-02	whr.us	70.8870	73.8420	70.8780	72.2900	2030206	0	1277.58
2008-09-02	forr.us	28.7730	29.5110	28.5390	29.2500	170160	0	1277.58

Table 1. aggregated dataset sample

Table 1. provides a sample of the dataset. For a detailed overview and description of each feature, see appendix E. This data is an unlabelled, structured, multivariate time-series with seven features (date and ticker are excluded). Additional metrics are harder to obtain from free sources. Note that in a production-grade system, a significantly higher number of features would be available.

4.3 Exploratory Data Analysis

For ML systems to produce better results, it is desirable to first analyse the data they are trained on (Sahoo et al., 2019). Exploratory data analysis (EDA) refers to this process of initial investigations on the data that tests hypotheses and checks any initial assumptions about the data. Note that this is conducted on the aggregated dataset of all time-series, not a specifically selected one.

4.3.1 Feature Correlation

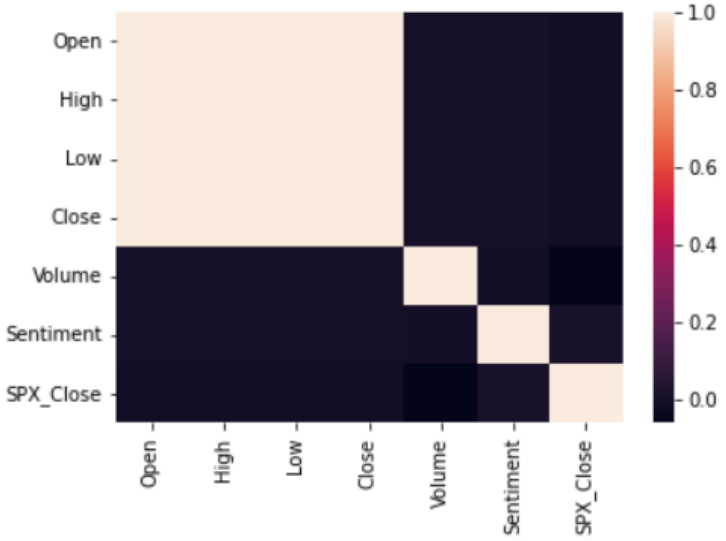


Figure 10. correlation heatmap (raw data)

Fig. 10, proves the intuitive assumption that the Open, Close, High, and Low features are highly correlated. Primarily, these are just fluctuations from the real underlying price of the asset. As such, they will move in a similar direction together. According to known assumptions about the market, it would be expected that the movements of the SPX index are highly correlated to the market sentiment. Since the dataset contains raw rather than relative movements, it falsely appears that there is no correlation.

4.3.2 Feature Distributions

	Open	High	Low	Close	Volume	Sentiment	SPX_Close
Mean	9418.5	9714.76	9009.04	9320.84	1372301	0.54	1569.76
STD	1116755	1151771	1048268	1093199	8027966	0.4988	406.48
MIN	0.00	0.004	0.00	0.0037	0.00	0.00	676.53
25%	9.28	9.43	9.12	9.28	21981	0.00	1219.66
50%	18.87	19.11	18.60	18.87	125574	1.00	1556.22
75%	34.75	35.17	34.31	34.75	620070	1.00	1981.57
MAX	790414800	790524600	749351400	755170600	2423735000	1.00	2130.82

Table 2. statistical properties of underlying feature distributions

Table 2 proves an assumption that the raw stock prices are widely spread. For example, Google stock is valued at around 2,600USD, while Exxon Mobil is valued at around 82USD. The STD values for distributions in Table 2. are very high and likely skewed by significantly large statistical outliers such as the ones seen in the MAX row.

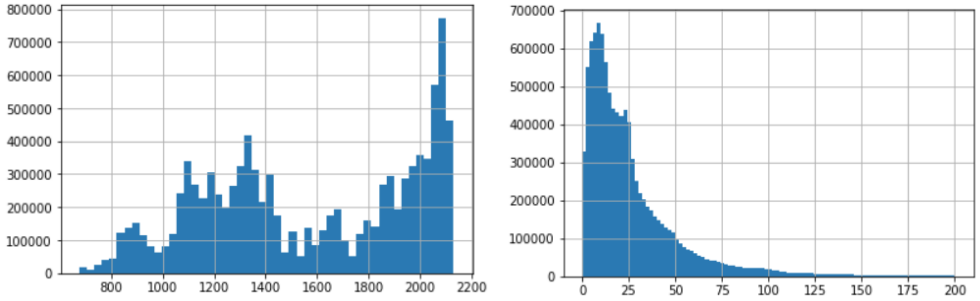


Figure 11. feature distributions for SPX_Close (left) and Close (right)

Fig. 11 shows the trend of the SPX index generally rising in value for the majority of the recorded period, spending most of the time in the value range of the low 2000s. It also shows a distribution for the Close feature partly resembling the shape of a Gaussian. However, in Table 6, it is seen that this distribution has a $\mu = 9320.84$ and $\sigma = 1093199$. Likely heavily skewed by large outliers such as the MAX of 755170600 seen in Table 6. The values also cannot be negative as they represent the raw price of an asset in USD.

All the figures and tables shown can be reproduced by running the *EDA.ipynb* notebook provided in the source code. Refer to Appendix C.

4.4 Data Pre-Processing

4.4.1 Feature Extraction

The essence of feature extraction is to derive features that best describe the nature of the dataset for a particular task. The concept is closely related to and can serve as dimensionality reduction where needed. For most of the features in this dataset, it is desirable to track intra-day changes in value instead of raw values. As illustrated in an example in 4.3.2, the raw values are far apart, and the SPX value is given in points instead of USD. Obtaining the relative change puts the values on a standardised scale in range (-100, 100), enabling accurate comparison. According to the CLT, this should also yield approximately normally distributed data. This does to the Sentiment feature which is retained in raw format to preserve meaningful state information.

Running the same notebook used for the initial EDA (see 3.2) on the dataset yields the following results (Fig. 12 and Fig. 13)

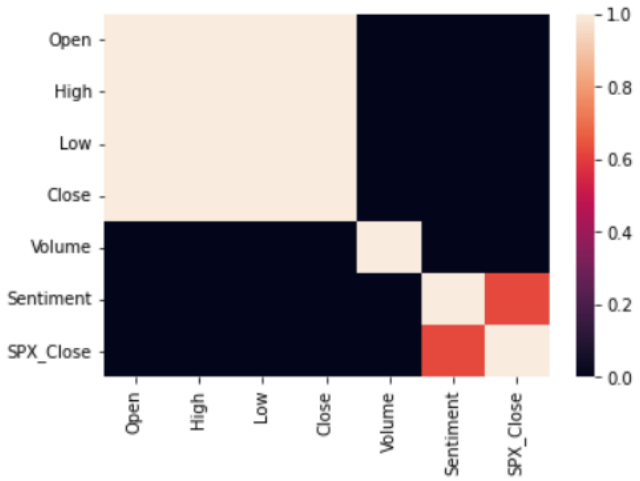


Figure 12. correlation heatmap (pre-processed data)

Fig. 12 shows a large spike in the correlation between SPX_Close and the sentiment indicator. This satisfies the hypothesis that both these values are representative of the overall market sentiment. Surprisingly, there is no direct correlation between the price movements of a stock and the SPX_Close. Since the SPX is a composite of many of the individual stocks, a higher correlation was expected.

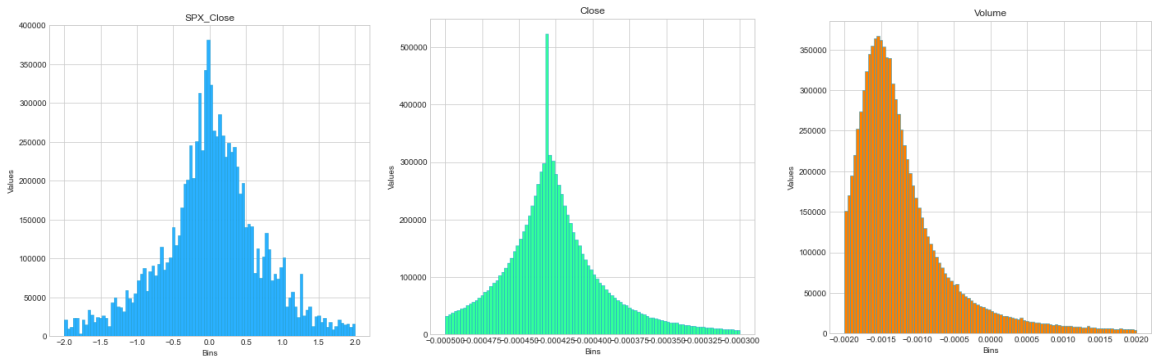


Figure 13. new feature distributions (Left to right – SPX_Close, Close, Volume)

Fig. 13 shows the newly derived feature distribution. These now roughly follow the shape of a Gaussian distribution with $\mu = 0$. However, given the significant outliers such as a 348366% increase, the STD remains highly skewed.

4.4.2 Normalisation

A solution to the problem with skewed values is the process of data normalisation, which retains the original distribution shapes but adjusts their statistical properties.

The most common type of normalisation is Z-Normalisation. This sets $\mu = 0$ and unit variance ($\sigma = 1$ for this data). It is done element-wise per feature by applying (13)

$$\hat{X} = \frac{X - \mu}{\sigma} \tag{13}$$

where X is the original element value, $\hat{X}$ is the new normalised value (Z-score), μ is the mean of its' feature distribution, and σ is the STD of the feature distribution.

	Open	High	Low	Close	Volume	SPX_Close
Mean	1.07e-15 ~=0.00	1.57e-15 ~=0.00	-7.32e-16 ~=0.00	3.25e-16 ~=0.00	2.30e-16 ~=0.00	6.35e-13 ~=0.00
STD	1	1	1	1	1	1

Table 3. statistical properties of normalised feature distributions

Table 3 shows that the resulting data frame now holds approximately normally distributed feature data. Note that the Sentiment feature was not normalised to retain meaningful information.

4.4.3 Trend and Seasonality

The extraction of relative percentage change yielding normally distributed data is presumed to have removed the trend of the series as seen in Fig. 14

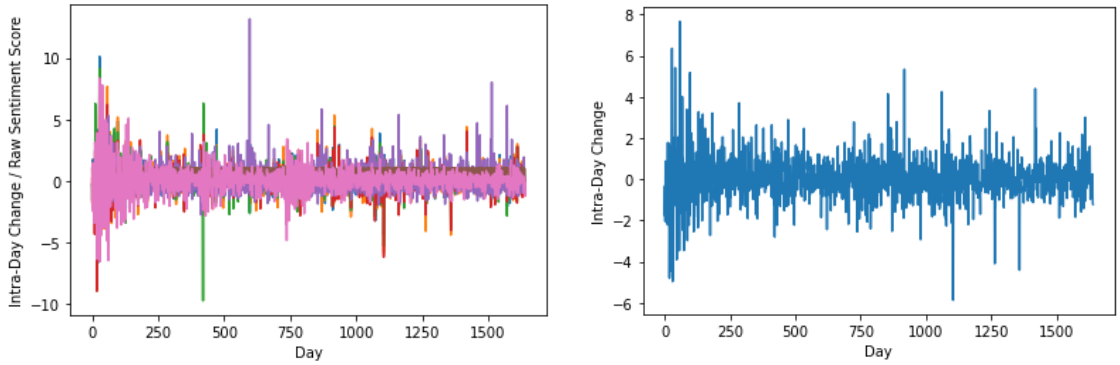


Figure 14. AAPL data visualisation (Left – full; Right – opening price change)

Therefore, techniques such as smoothing or Fourier transformations do not have to be applied. Lastly, recall that seasonal anomalies are a sub-category of market anomalies (see 2.2). Any signs of seasonality detected in normally distributed data should likely be considered as a sign of a market anomaly and flagged accordingly.

Results shown in section 4.4 can be reproduced by running the same *EDA.ipynb* notebook mentioned in 4.3 with the *raw* flag set to False in the cell that loads the data.

4.5 Selected Time-Series

This section discusses the datasets used for experimentation, assumptions about the data, and the rationale behind the choices. Multiple time-series are selected from the financial dataset mentioned earlier in section 4.2. Each stock and ETF in the market is different and as such, they will be prone to different types of anomalous behaviour. It is therefore important to select time-series that belong to different groups to properly test the overall suitability of each model. This dataset is unlabelled and given the market behaviour, no assumptions can safely be made about the number of anomalies.

4.5.1 Apple stock (AAPL)

Technology stocks such as *AAPL* commonly have higher volatility and appear more sensitive to the general news sentiment of the market (Brook & Cho, 2022). As such, the dataset is expected to contain multiple cases of market anomalies caused by short-term mispricing. In the past year alone, there were multiple cases of sharp declines in price for technology stocks. Similar behaviour is presumed to be reflected across instances in the whole time-series.

4.5.2 General Motors stock (GM)

Well established companies that have been in the market for a long time such as *GM* will generally be more stable in terms of price fluctuations. This is primarily because there is no flood of growth driven investments into the company, which would cause excessive sensitivity to news sentiment.

4.5.3 American Express stock (AXP)

The *AXP* time-series was picked to draw samples from the financial sector. This sector is presumed to be heavily affected by regulations and its performance correlates with how healthy the market is overall. This particular stock has not seen much growth.

4.6 External Benchmarking Dataset

An external dataset is also chosen to serve as a known, labelled proving ground that allows for initial testing of the models and the training procedure to first eliminate any errors before proceeding to testing on the main financial dataset. It also allows for illustrating differences between model performance in a different environment compared to the financial dataset.

4.6.1 kdd99

The *kdd99* is widely used as one of the baseline datasets for anomaly detection tasks and in this project serves to verify the models are implemented correctly. It is also used to assess how well the models extrapolate their performance to a different setting when compared to the financial market data. The variation used in this project is split into non-anomalous training data and mixed test data. There are 34 variables, however, dimensionality reduction to an arbitrary number between 0 and 34 is supported using naive PCA (Li et al., 2019). This choice is model dependent. All data is labelled.

Chapter 5: **Experimental Setup**

The primary objective of the experiment is to evaluate the performance of generative DNNs introduced in section 3.5 on the market anomaly detection problem. Common density-based AD models LOF and OC-SVM are selected as baselines. HBOS is selected as the baseline for statistical models (Goldstein & Uchida, 2016; Goldstein & Dengel, 2012). The results of DNNs will also be benchmarked against these.

This chapter briefly discusses the implementation of the training and evaluation procedure used in this experiment. According to Zhang et al. (2020), the outline of a model training pipeline can be seen as the following sequence of steps

Algorithm 1: General NN Training Pipeline

1. Load dataset;
2. Define model and hyperparameters;
3. Define loss and optimizer;
4. Train model;
5. Evaluate model;

5.1 Loading the Dataset

All loading of the source datasets, type conversions, and pre-processing is handled by methods provided in the data_utils file using primarily pandas and NumPy libraries. Dimensionality reduction via naive PCA and support for loading data as a sequence of sliding windows for sequence forecasting are provided. Data is split into batches and provided to the training pipeline as a torch DataLoader object. For more detail on the rationale behind the choice of data transformations refer to chapter 4.

5.2 Model Implementation

For the NN architectures, guidance on the choice of layers and hyperparameters was primarily taken from the original papers (Li, 2019; Ngo et al., 2019; Lai, Zou & Lerman, 2019; Munir et al., 2019), however, adjustments had to be made to better fit some datasets. For example, RSRAE is trained with an entirely different set of layers and parametrization. Dropout was tested and introduced to the original architectures where appropriate. Empirical hyperparameter tuning was also performed. See Appendix F for full specification of parametrisation and layer choices. Other significant changes were made on supporting GPU computation (see 2.8.2) and integrating the models with the custom training pipeline. HBOS, LOF, and OC-SVM are imported from the PyOD library (pyod, n.d.) and trained with default parametrisation.

The implementations are defined in the *models* package and the models can be simply imported and initialized in their respective Jupyter notebook files with a chosen set of hyperparameters and passed to a pipeline defined in the *training* package. Any source code used is referenced accordingly inside the files and in Appendix C.

Note that the explanation of model implementations provided here is very brief on purpose, as the main insights of this research are drawn from the results, rather than any advancements in terms of NN architecture design.

5.3 Loss and Optimizer

The choice of loss and optimizer (see 2.3.5 and 2.5.3) is unique for each model. While some optimizers are shown to generally work better than others, the computational complexity and nature of the model also need to be taken into consideration. All models used in this project have fully differentiable architectures and can be trained with gradient-based optimizers and automatic gradient computation provided by torch. Weight decay is introduced to some models. Adaptive learning rate via torch scheduler was also implemented. More detail is provided in Appendix F.

5.4 Model Training

Each model is provided with a separate custom training pipeline that contains all necessary methods for training available in the *training* package of the project source code. The pipelines simply need to be imported to the notebook file and the model be passed in for training. This choice was made to improve the cleanliness of the notebooks and ease the user experience during result replication.

5.4.1 Forward Chaining Cross-Validation

Gulenko et al. (2016) provide empirical evidence that training with cross-validation increases the average performance on the validation dataset by a significant margin. It also helps increase the confidence that the model performance is consistent across different subsets of the original data.

For time-series datasets it is not desirable to use cross-validation that shuffles the order of elements, such as k-fold. Instead, forward chaining cross-validation can be used, which retains the order of elements (Holmstrom, Liu & Vo, 2016; Bergmeir & Benítez, 2012). Forward chaining cross-validation splits the dataset into an arbitrary number of folds of equal size and starts by training the model on the first folds and testing on the next. Continue this by expanding the training set with the next fold and train on the one after that. This is repeated until the last fold is used. For example, a dataset with 5 folds would be trained as follows

Algorithm 2: 5-fold forward chaining cross-validation split

1. Training: [1]; Testing: [2];
2. Training: [1,2]; Testing: [3];
3. Training: [1,2,3]; Testing: [4];
4. Training: [1,2,3,4]; Testing: [5];

An average of the testing results is then computed, representing final evaluation score. This is implemented in code using methods from the ScikitLearn library.

5.5 Model Evaluation

5.5.1 Established evaluation methods

Vakili, Ghamsari & Rezaei, (2020) provide a summary of model evaluation metrics commonly used in ML research papers. The most prominent metrics were accuracy, F1 score (14), Precision/Recall (14), and Receiver Operating Characteristic (ROC) curves.

$$precision = \frac{TP}{TP + FP}; recall = \frac{TP}{TP + FN};\ F1 = 2 * \frac{precision * recall}{precision + recall} \tag{14}$$

where TP are true positives, FP are false positives, and FN are false negatives. Note that these metrics are only used for initial evaluation on the kdd99 dataset. All these metrics rely on knowing the correct model output (i.e., the label) and therefore cannot be used for the proprietary dataset. For unsupervised validation without labels, it is needed to best approximate the performance of the model based on just the input data.

5.5.2 Excess-Mass (EM) and Mass-Volume (MV)

When comparing unsupervised anomaly detection algorithms, Goix, (2016b) proposes using Excess-Mass (EM) and Mass-Volume (MV) curves for ordering anomaly scoring functions based on their proximity to the ideal scoring function. In the majority of cases, the EM/MV measurement identifies the same best algorithms as PR and ROC curves (Warzynski, Falas & Schauer, 2021).

With the set of all scoring functions *S* such that s: $X \rightarrow \mathbb{R}_+$ is integrable *w.r.t* Lebesgue measure and $s \in S$. A high-level overview of EM/MV curves is as follows:

(15) gives the MV curve of *s* as the plot of the mapping

$$\alpha \in (0,1) \rightarrow MV_s(\alpha) = inf\,\{Leb(s \geq u)\}\, s.t.\, \mathbb{P}(s(\boldsymbol{X}) \geq u) \geq \alpha;\, u \geq 0 \qquad (15)$$

(Goix, 2016b:p.2)

(16) gives the EM curve of *s* as the plot of the mapping

$$t > 0 \rightarrow EM_s(t) = \sup\{\mathbb{P}(s(\boldsymbol{X}) \geq u) - tLeb(s \geq u)\}\,;u \geq 0 \qquad (16)$$

(Goix, 2016b:p.2)

where *inf* is the infimum, *sup* is the supremum, *Leb* denotes Lebesque integration, $\mathbb{P}$ is a power set, $\alpha \in (0,1)$ and $u \geq 0$ are set parameters of the measurement (Goix, 2016b; Goix, Sabourin & Clémençon, 2015).

(15) can intuitively be thought of as a measurement of proximity between *s* and the perfect scoring function, as given by the *Leb* measurement of the area under the curve.

(16) can be thought of as a reframed objective from (15); i.e., instead of minimising the *Leb* measure, EM is aimed at maximising the difference between the *Leb* measure and retrieved powerset. This has been shown to provide desirable numerical properties over MV (Goix, 2016b; Goix, Sabourin & Clémençon, 2015).

These criteria induce a partial ordering on the sets of all scoring functions *S*, such that *s* is preferred to s' if $MV_s(\alpha) \leq MV_{s'}(\alpha)$ for all $\alpha \in (0,1)$. Similarly, *s* is preferred if $EM_s(\mathrm{t}) \geq EM_{s'}(\mathrm{t})$ for all $\mathrm{t} > 0$.

In practice, the distribution of the data is assumed to be unknown and the EM/MV values must be estimated with a Monte-Carlo simulation. The EM/MV measurements are therefore the closest empirical estimations of their true values.

Original code for EM/MV computation is provided by Goix (2016a). More concise implementation is given by O'leary (2021), however, changes had to be made to support torch models and GPU computation. This eliminated the need to transfer data between GPU and CPU for evaluation, significantly speeding up the Monte-Carlo step for costly reconstruction-based models such as MAD-GAN.

Chapter 6: **Experimental Results**

This chapter presents the results obtained from experimentation and discusses the findings. The main purpose of the experimentation was to provide a comparative evaluation and verify/refute theoretical assumptions and hypotheses about the models. Firstly, the models are evaluated on the *kdd99* benchmarking dataset. Next, they are evaluated on a variety of time-series selected from the *financial dataset* discussed in chapter 4. Implicit ordering of the models is retrieved using the EM/MV criterion introduced in 5.5.2. The full experimental result can be found in Appendix D.

Recall from 2.1.2 that the objective of the models is to classify data as anomalous or non-anomalous. While some models are also capable of sequence forecasting, the experiment was done on one-step-ahead forecasting/detection only, as the goal is to detect singular anomalous data points, rather than sequences containing an anomaly.

6.1 Performance Evaluation

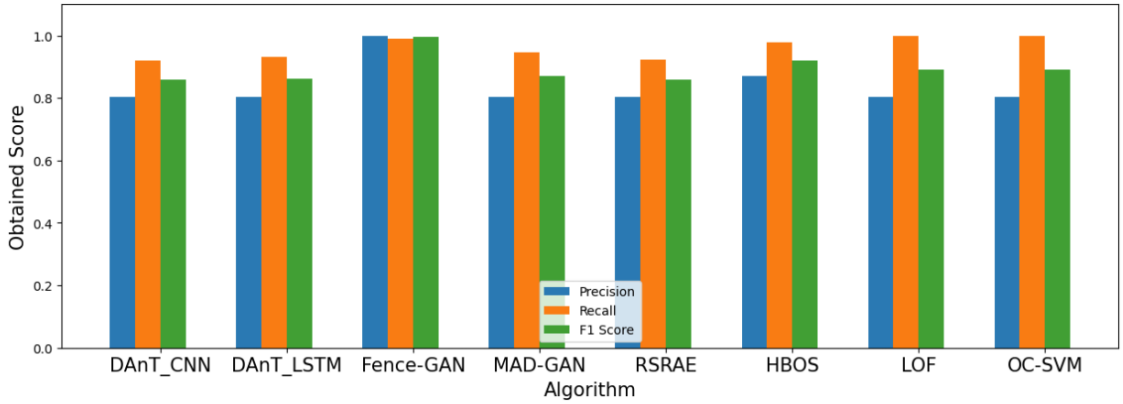


Figure 15. Precision, Recall, and F1 scores on kdd99

According to the PR and F1 scores in Fig. 15, nearly all models are capable of roughly equal performance on the kdd99 dataset with precision > *0.8* and Recall > *0.9*. Note that it was possible to achieve different trade-offs between Precision and Recall, however, results are selected according to the best F1 score (14), which accounts for both PR measurements. Fence-GAN achieved near-perfect scoring at recall = *0.99* and precision = *1.0*, outperforming previously presented results on different versions of the kdd99 dataset. It should be noted that the baseline models HBOS, LOF, and OC-SVM were expected to perform well on this type of data, which was proven and reflected in the results. MAD-GAN and both DeepAnTs produce scores roughly in line with the results presented in prior research papers. The RSRAE results were expected to be higher based on prior experiments on kdd99 by Lai, Zou & Lerman (2019), albeit on a different variation of the dataset. This might be indicative of a flawed choice of layers or hyperparameters. The source of the exact kdd99 dataset used in (Lai, Zou & Lerman, 2019) was not found and therefore a comparison could not be made.

For EM/MV scores, note that attention should be paid primarily to the retrieved ordering, rather than the difference in value between scores. As mentioned briefly in 5.5.2, EM/MV criteria retrieve a partial ordering on the set of *all* scoring functions *w.r.t* the dataset (Goix, 2016b). Therefore, the entire algorithm itself is evaluated, rather than only its trained scoring function.

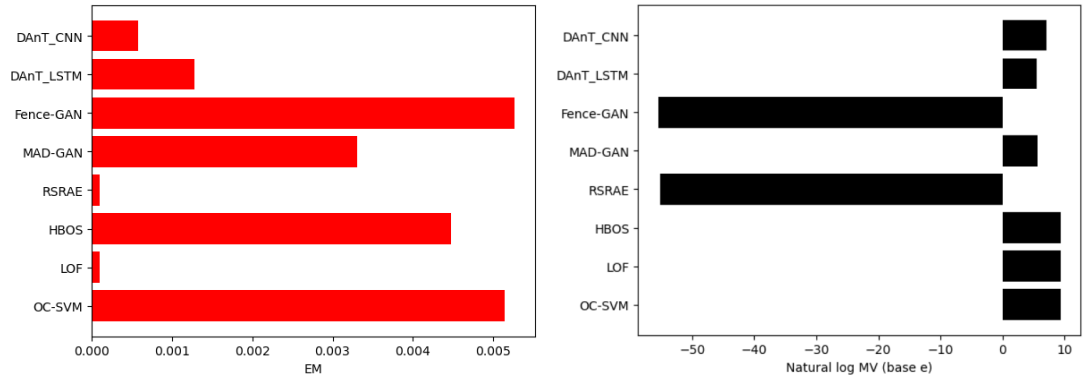


Figure 16. EM (red) and MV (black) scores on kdd99 dataset

In Fig. 16, it can be observed how EM scores on kdd99 retrieve nearly the same ordering as F1 scores with OC-SVM now slightly outperforming HBOS. RSRAE and LOF appear to be significantly outperformed by other algorithms, however, the ordering remains accurate. The MV criterion ordering seems to favour RSRAE, which scored nearly as low as Fence-GAN (recall from 5.5.2 that for MV, lower scores indicate better results). The baseline algorithms are penalised heavily compared to DNNs. The large discrepancy between RSRAE rankings in EM and MV is unusual and not seen in any of the financial time series or replicated by any of the other algorithms.

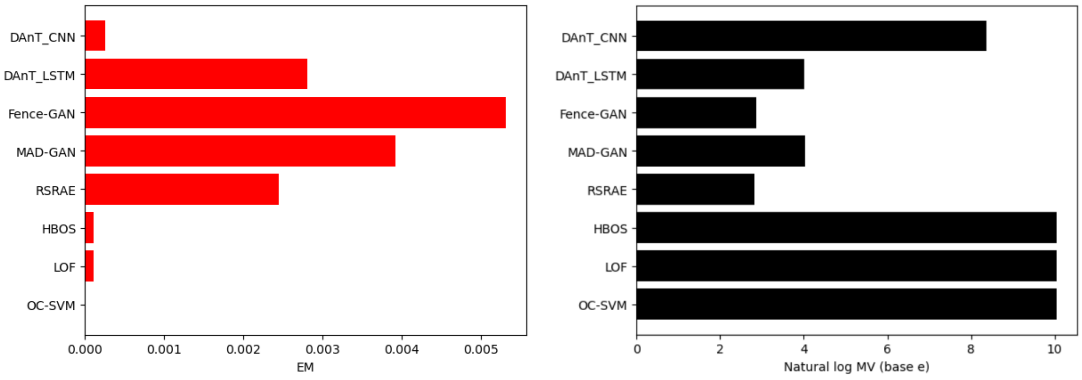


Figure 17. EM (red) and MV (black) scores on AAPL dataset

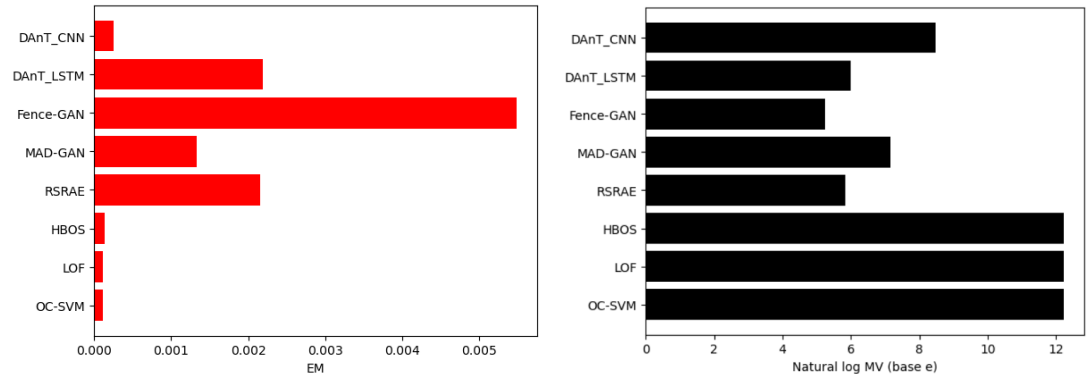


Figure 18. EM (red) and MV (black) scores on GM dataset

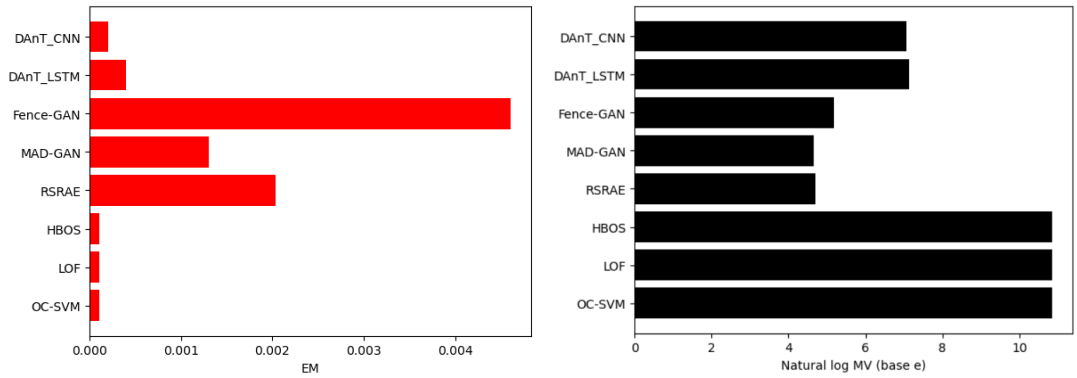


Figure 19. EM (red) and MV (black) scores on AXP dataset

From Fig. 17 – 19, it can be observed that the order changes significantly on time-series from the financial dataset compared to kdd99. HBOS, LOF, and OC-SVM appear to significantly deteriorate in performance according to the EM criterion. This was expected as HBOS assumes independence of the features, which is not satisfied. Similarly, LOF is a density-based approach, implying the limitations discussed in 3.4. The standard OC-SVM is designed for training on predominantly normal data and as such, this result was expected on the financial dataset. It is suggested by Goldstein & Uchida (2016) that more robust versions of OC-SVM have been developed, which should be capable of training on data with more outliers, however, these were not tested as this algorithm only serves as a commonly established baseline.

Fence-GAN retains the best performance according to both EM and MV, closely followed by RSRAE. These results are in line with those seen on kdd99 and further support the theory and prior experiments presented in chapter 3. This is indicative of the superior capabilities of the RSR layer and Fence-GAN objective as compared to directly reconstructing points on the full underlying distribution attempted by MAD-GAN.

The LSTM based DeepAnT outperforms the CNN version on all datasets, in line with the initial ordering established by F1 and PR scores on kdd99. However, it is surprising that the LSTM based version has repeatedly beaten the CNN, as it achieved strong results on the *http* data subset from kdd99 in experiments by Munir et al. (2019).

LOF, OCSVM, and HBOS all obtain the same MV score on all tested time-series in the financial dataset. It is speculated that this might be caused by an issue in the PyOD library implementation of these algorithms, as the MV evaluation works for all other models. Furthermore, this issue is not present when evaluating the algorithms on kdd99 with the same method, however, the values are very close to each other, therefore, a rounding error is suspected. Based on this observation, the MV measurements for LOF, OC-SVM, and HBOS presented in Fig. 17-19 are not regarded as reliable and further investigation is needed to determine the root cause of this issue.

6.1.1 Impact of asset volatility on model performance

Historically the AAPL stock is approximately 10-20% more volatile than the market benchmark. GM is in the 10-15% range on average, and AXP is approximately 5-10% higher (PortfolioLab, n.d.) as reflected on the intra-day price changes seen in Fig. 20.

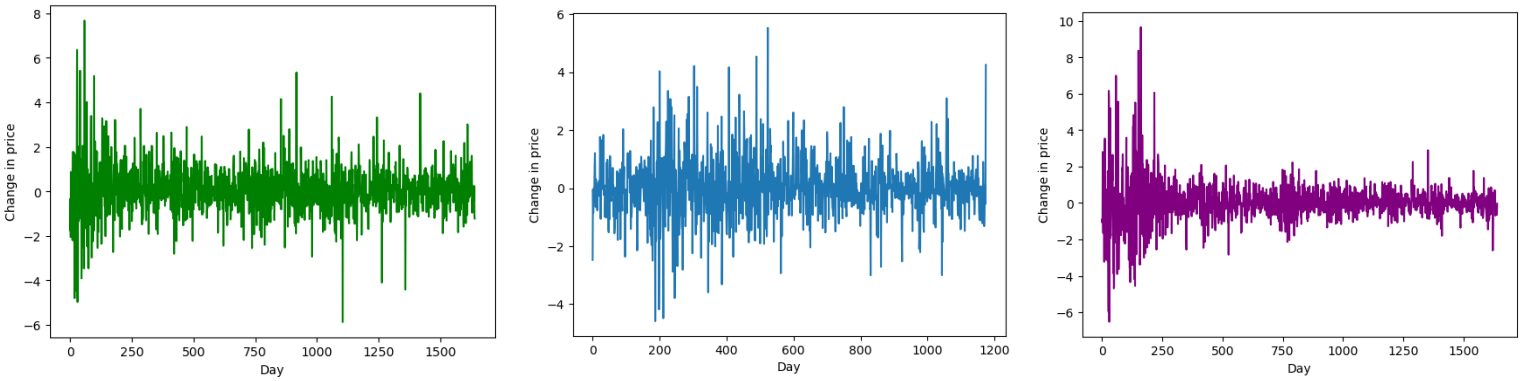


Figure 20. intra-day closing price change (left to right – AAPL, GM, AXP)

It is assumed that a higher degree of volatility naturally implies presence of more pricing anomalies (see 2.2.1). Notably, Fig 17. shows that MAD-GAN is performing surprisingly well for a reconstruction-based model, as opposed to the CNN and LSTM DeepAnTs. Observations from the remaining datasets and criteria suggest that reconstruction-based and forecasting models perform worse overall than Fence-GAN and RSRAE. This is not surprising as they rely on the unsatisfied assumption that training data is non-anomalous. The RSRAE ranked lower than expected according to EM on AAPL, which goes against initial assumptions about the capabilities of the RSR layer. OC-SVM has recorded its worst performance overall on the volatile AAPL dataset. Fence-GAN handles all three datasets with stable performance.

6.2 Runtime Evaluation

Model	Training Time (s / epoch)	Detection Time (s / batch)
DeepAnT_CNN	12.189	0.001181
DeepAnT_LSTM	15.795	0.004829
Fence-GAN	1.8845	0.000088
MAD-GAN	241.47	0.016829
RSRAE	2.5154	0.002004
HBOS	**0.0169**	**0.000031**
LOF	0.1130	0.020637
OC-SVM	146.48	1.069103

Table 4. runtime on kdd99

Table 4 shows training and detection time measurements for all models on the kdd99 dataset. Training time is measured in seconds per epoch, while detection time is measured in seconds per batch of unit size 256 with 6 features.

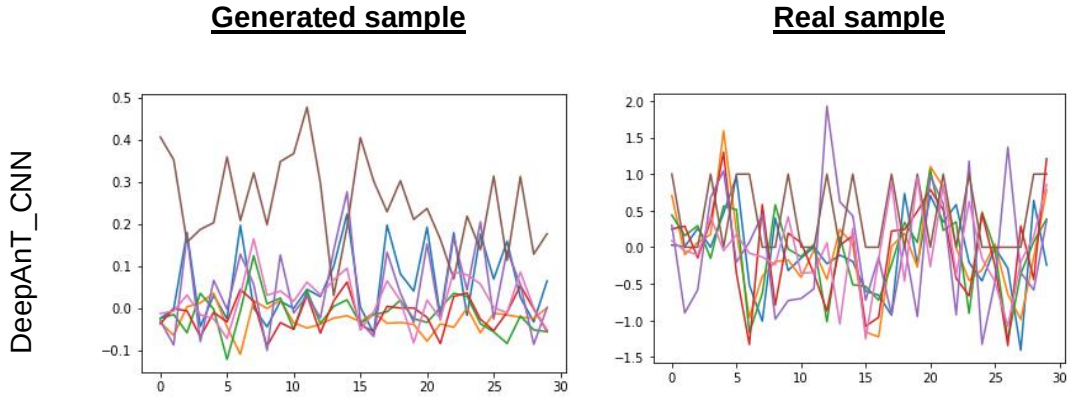
It is shown that the statistical HBOS has the lowest computation cost. This result was expected, as stated in section 1.1 and theoretically explained in sections 3.2.3 and 3.4. Fence-GAN is surprisingly close in terms of detection time. This can be explained by the lightweight MLP architecture and using only the discriminator for anomaly scoring. When the outstanding performance on both the financial dataset and kdd99 are considered as well, the Fence-GAN appears to be the most promising algorithm.

Both LOF and OC-SVM have higher detection times than any NN based model. This is not surprising for OC-SVM, but LOF was expected to beat models with complex scoring functions such as MAD-GAN, which needs to perform a small Monte-Carlo simulation to compute its reconstruction loss. In general, it is seen that the more complex forecasting LSTM-based architectures are slower than the lightweight MLP architectures of Fence-GAN and RSRAE, or the CNN based DeepAnT.

Lastly, it should be noted that the detection time on LOF, HBOS, and OC-SVM might be slightly inflated due to their underlying implementation. The models were imported directly from PyOD and are implemented to operate on NumPy arrays restricted to CPU computation. This introduces a significant overhead when compared to torch implementations that support GPU computation and greatly optimises linear operations such as matrix multiplications.

6.3 Generative Model Samples

To better illustrate how the different objectives and training procedures adopted by the generative models seen in chapter 3 extract meaningful information from the data, samples of their generated sequences are shown in Fig. 21.



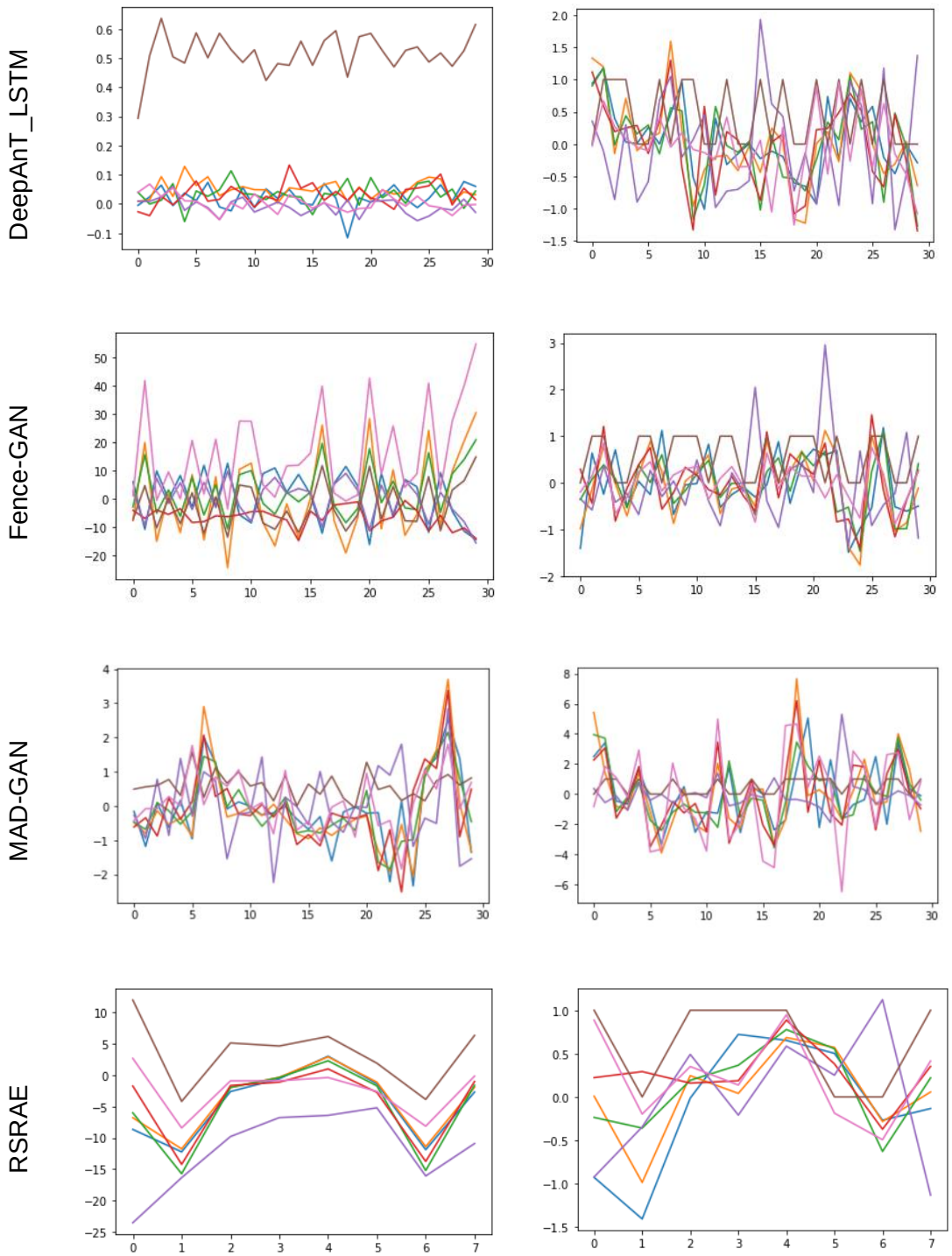


Figure 21. qualitative comparison of generated samples on AAPL

In Fig. 21, one may observe how the modified objective of different NNs reflects on the generated samples. Both DeepAnT versions and MAD-GAN attempt to capture the true underlying distribution of the data with a varying degree of success. DeepAnT makes a prediction based on the real values observed in the past sequence, while MAD-GAN is trained in an adversarial manner to predict based on latent space samples.

It can be seen how Fence-GAN captures the boundary of the underlying distribution via modified training loss seen in 3.3.3. For RSRAE, one may observe the smoother curves resulting from the outlier removal introduced by the RSR layer.

Chapter 7: **Overview**

This section is primarily dedicated to a discussion of possible advancements toward improvements in solving the market anomaly detection problem. Subsequently, conclusions are drawn from the experimental results.

7.1 Current Limitations and Future Work

This section aims at suggesting improvements that can be made to address the limitation of current models found during the experimental and theoretical research. It also highlights what objects were either not achieved or outside the scope of this project, but would be beneficial to explore in the future.

7.1.1 Further Experimentation

While the experimental research in this project has brought forward useful observations, there remains a list of additional objectives that can contribute to making the results more conclusive. Firstly, the issues with the PyOD library rendered their MV measurements on the financial dataset unusable. This needs to be addressed by either implementing the models manually or debugging issues within the library source code, which is challenging. Additionally, when the EM and MV criteria are fully established, further experiments should be conducted to study the effectiveness of tested models in relation to metrics other than volatility.

While the ordering of algorithms is retrieved successfully, it is hard to gauge whether the model did well by itself judging solely by the EM/MV measurements. Further study into proper model optimisation for EM/MV is needed. This was out of the scope of this project but is essential in attempting to build a production-ready ML system.

The TadGAN (Geiger et al., 2020) model appears promising but was not known at the start of the experiments. This should be included in the experiment to accurately account for all recent SOTA models.

7.1.2 Learning Group Dependent Behaviour

A pressing limitation of current SOTA AD models is the inability of the generators to learn from multiple time-series. As such, the generator cannot capture complex group dependent behaviour without human expertise and excessive feature engineering. This can bring strong bias into the learning process (Barcelos, 2021). As seen in 4.1 and 4.2, applying current AD models to financial markets is currently heavily dependent on such human expertise.

Salinas, Flunkert & Gasthaus (2017) developed DeepAR, which is capable of learning from large numbers of multivariate time-series for forecasting. Incorporating DeepAR or similar advanced forecasting methods in AD can help further expand the capabilities of recent SOTA models. There appears to be an initial attempt at applying DeepAR to AD, although no conclusive results are published (Song, 2021). Geiger et al. (2020) propose a TadGAN, which is stated to outperform DeepAR for AD tasks, however, it is unclear whether DeepAR was used for capturing group dependent behaviour in their experiment.

7.1.3 Self-Supervised Learning

Self-supervised learning is a promising alternative to doing fully unsupervised learning (FAIR, 2021) that is a de-facto mixture of supervised and unsupervised learning which predicts its own labels and uses them for training. It has been used for AD in computer vision (Google Research, 2021) and time-series (Rafiee et al., 2022; Schneider et al., 2022) with promising results. It is unclear whether current self-supervised methods can better support learning group dependent behaviour.

7.1.4 Reinforcement Learning

The environment model of the financial markets is best suited for an RL approach as discussed in 4.1. RL has recently been taken into consideration for time-series AD tasks thanks to its generic framework and incremental self-learning property (Yu & Sun, 2020). This is reflected in recent literature with numerous successful attempts at using RL for AD and time-series classification tasks (Martinzez et al., 2018; Zhong, Cenk Gursoy & Velipasalar, 2019; Zha et al., 2020; Wu & Ortiz, 2021). However, current RL agents generally need relatively correct instructions for learning an effective detection policy (Yu & Sun, 2020). As such, further research into incorporating self-supervision into RL (Levine, 2021) or effective fully unsupervised training is needed to implement RL for effective market anomaly detection.

7.1.5 Operating Under Uncertainty

The concept of uncertainty in anomaly detection was briefly introduced in section 2.5.3 and expanded on in the context of market anomaly detection in section 4.1. There has been a recent attempt at incorporating probabilistic approaches to unsupervised anomaly detection (Yong & Brintrup, 2022). However, this work is very recent and other attempts have not been found. Therefore, there is space for experimentation with novel architectures incorporating Bayesian methods to known AD models. Notably, an adaptation of probabilistic GAN models for AD might be an interesting venue. Applied research is also absent in this area.

7.2 Conclusions

The research has concluded that recent generative anomaly detection models significantly outperform traditional statistical and density-based methods in an environment with complex real-world data. However, application in a productionised system would still require significant human expertise in modelling the operational environments for the models. As such, they are deemed to be not yet fully capable of identifying complex abstract patterns in the broader unconstrained environment of financial markets and the further research is needed in both theory and application.

Venues for further research that can advance the current SOTA for better performance on the market anomaly detection problem notably includes adopting mechanisms for learning group dependent behaviour without the need for feature engineering and incorporating the notion of uncertainty into unsupervised models via Bayesian methods. Further research into fully unsupervised RL agents for AD is another venue which started to be explored intensively in the last two years.

Hopefully, the reader has found the presented results and derived conclusions informative, and the produced work will help advance efforts toward solving the market anomaly detection problem with novel efforts in ML.

References

Aaron7sun (2019) *Daily News for Stock Market Prediction*. 2019. https://www.kaggle.com/aaron7sun/stocknews?ref=hackernoon.com [Accessed: 21 November 2021].

Aggarwal, C.C. (2016) An introduction to outlier analysis. In: *Outlier Analysis*. Cham, Switzerland, Springer. pp. 1–40.

Ahmad, M.A., Eckert, C. & Teredesai, A. (2018) *Interpretable Machine Learning in Healthcare*. In: 15 August 2018 Association for Computing Machinery (ACM). pp. 559–560. doi:10.1145/3233547.3233667.

Ahmed, M., Choudhury, N. & Uddin, S. (2017) Anomaly detection on big data in financial markets. In: *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2017*. 31 July 2017 Association for Computing Machinery, Inc. pp. 998–1001. doi:10.1145/3110025.3119402.

Ahmed, M., Mahmood, A.N. & Islam, M.R. (2016) A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems*. 55, 278–288. doi:10.1016/j.future.2015.01.001.

Akyildirim, E., Matteo, G., Teichmann, J. & Zhou, S. (2022) *Identification of Stock Market Manipulation with Deep Learning*. doi:10.36227/techrxiv.19111730.v1.

Alpaydin, E. (2014) *Introduction to Machine Learning*. Cambridge, MA, MIT Press.

Bank of England (2019) *Machine learning in UK financial services*. www.fca.org.uk/news/speeches/future-regulation-ai-consumer-good.

Barcelos, G. (2021) *Understanding Bias in Machine Learning Models*. 2021. https://arize.com/blog/understanding-bias-in-ml-models/ [Accessed: 2 May 2022].

Bergmeir, C. & Benítez, J.M. (2012) On the use of cross-validation for time series predictor evaluation. *Information Sciences*. 191, 192–213. doi:10.1016/j.ins.2011.12.028.

Bloomberg (n.d.) *Bloomberg Enterprise Datasets*. https://www.bloomberg.com/professional/datasets/.

bmonikraj (2020) *DeepAnT code repository*. 2020. https://github.com/bmonikraj/medium-ds-unsupervised-anomaly-detection-deepant-lstmae [Accessed: 25 April 2022].

Breunig, M.M., Kriegel, H.-P., Ng, R.T. & Sander, J. (2000) LOF. In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data - SIGMOD '00*. 2000 New York, New York, USA, ACM Press. pp. 93–104. doi:10.1145/342009.335388.

Brook, D. & Cho, S. (2022) *Tech-Sector Volatility: Fundamentals versus Fears*. 2022. https://www.gsam.com/content/gsam/us/en/institutions/market-insights/gsam-connect/2022/Tech-Sector_Volatility_Fundamentals_versus_Fears.html [Accessed: 26 April 2022].

Campos, G.O., Zimek, A., Sander, J., Campello, R.J.G.B., Micenková, B., Schubert, E., Assent, I. & Houle, M.E. (2016) On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery*. 30 (4), 891–927. doi:10.1007/s10618-015-0444-8.

Chandola, V., Banerjee, A. & Kumar, V. (2009) Anomaly Detection : A Survey. *ACM Computing Surveys*.

Cunningham, P. & Delany, S.J. (2020) *k-Nearest Neighbour Classifiers: 2nd Edition (with Python examples)*. doi:10.1145/3459665.

David, P. (2018) *S&P500 Daily Prices 1986 - 2018*. 2018. https://www.kaggle.com/pdquant/sp500-daily-19862018 [Accessed: 21 November 2021].

Do, C.B. (2019) *The Multivariate Gaussian Distribution*.

Doshi-Velez, F. (2009) *The Infinite Partially Observable Markov Decision Process*.

FAIR (2021) *Self-supervised learning: The dark matter of intelligence*. https://ai.facebook.com/blog/self-supervised-learning-the-dark-matter-of-intelligence/.

Geiger, A., Liu, D., Alnegheimish, S., Cuesta-Infante, A. & Veeramachaneni, K. (2020) *TadGAN: Time Series Anomaly Detection Using Generative Adversarial Networks*. http://arxiv.org/abs/2009.07769.

Ghahramani, Z. (2004) *Bayesian Machine Learning*. 2004. http://mlg.eng.cam.ac.uk/zoubin/bayesian.html [Accessed: 29 November 2021].

Goix, N. (2016a) *EMMV code repository*. 5 July 2016. https://github.com/ngoix/EMMV_benchmarks [Accessed: 9 April 2022].

Goix, N. (2016b) *How to Evaluate the Quality of Unsupervised Anomaly Detection Algorithms?* http://arxiv.org/abs/1607.01152.

Goix, N., Sabourin, A. & Clémençon, S. (2015) *On Anomaly Ranking and Excess-Mass Curves*.

Goldstein, M. & Dengel, A. (2012) *Histogram-based Outlier Score (HBOS): A fast Unsupervised Anomaly Detection Algorithm Time-Series Generation with GANs View project Histogram-based Outlier Score (HBOS): A fast Unsupervised Anomaly Detection Algorithm*. https://www.researchgate.net/publication/231614824.

Goldstein, M. & Uchida, S. (2016) A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PLoS ONE*. 11 (4). doi:10.1371/journal.pone.0152173.

Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014) *Generative Adversarial Nets*. http://www.github.com/goodfeli/adversarial.

Google Research (2021) *Discovering Anomalous Data with Self-Supervised Learning*. https://ai.googleblog.com/2021/09/discovering-anomalous-data-with-self.html.

Guillem96 (2022) *madgan-pytorch code repository*. 2022. https://github.com/Guillem96/madgan-pytorch [Accessed: 25 April 2022].

Gulenko, A., Wallschläger, M., Schmidt, F., Kao, O. & Liu, F. (2016) Evaluating Machine Learning Algorithms for Anomaly Detection in Clouds. In: *Proceedings, 2016 IEEE International Conference on Big Data : Dec 05-Dec 08, 2015, Washington D.C., USA*. 2016 p.

Harper, D. (2021) *Forces That Move Stock Prices*. 2021. https://www.investopedia.com/articles/basics/04/100804.asp [Accessed: 3 May 2022].

Hassan Mahmud, M.M. (2010) *Constructing States for Reinforcement Learning*.

Holmstrom, M., Liu, D. & Vo, C. (2016) *Machine Learning Applied to Weather Forecasting*.

Hu, X., Chu, L., Pei, J., Liu, W. & Bian, J. (2021) Model complexity of deep learning: a survey. *Knowledge and Information Systems*. 63 (10), 2585–2619. doi:10.1007/s10115-021-01605-0.

Inoue, J., Yamagata, Y., Chen, Y., Poskitt, C.M. & Sun, J. (2017) Anomaly detection for a water treatment system using unsupervised machine learning. In: *IEEE International Conference on Data Mining Workshops, ICDMW*. 15 December 2017 IEEE Computer Society. pp. 1058–1065. doi:10.1109/ICDMW.2017.149.

iSmile Technologies (2021) *Anomaly Detection: (AD) in Stock Prices with LSTM Auto-Encoders*. https://www.ismiletechnologies.com/technology/anomaly-detection-ad-in-stock-prices-with-lstm-auto-encoders/.

Kenton W (2022) *S&P 500*. 2022. https://www.investopedia.com/terms/s/sp500.asp [Accessed: 28 February 2022].

Lai, C.-H., Zou, D. & Lerman, G. (2019) *Robust Subspace Recovery Layer for Unsupervised Anomaly Detection*. http://arxiv.org/abs/1904.00152.

Latif, M., Arshad, S., Fatima, M. & Farooq, S. (2011) *Market Efficiency, Market Anomalies, Causes, Evidences, and Some Behavioral Aspects of Market Anomalies*. www.iiste.org.

Lecun, Y., Bengio, Y. & Hinton, G. (2015) Deep learning. *Nature*.521 (7553) pp.436–444. doi:10.1038/nature14539.

Levine, S. (2021) *Understanding the World Through Action: RL as a Foundation for Scalable Self-Supervised Learning*. 2021. https://medium.com/@sergey.levine/understanding-the-world-through-action-rl-as-a-foundation-for-scalable-self-supervised-learning-636e4e243001 [Accessed: 2 May 2022].

Li, D. (2019) *MAD-GAN code repository*. 2019. https://github.com/LiDan456/MAD-GANs [Accessed: 25 April 2022].

Li, D., Chen, D., Goh, J. & Ng, S. (2018) *Anomaly Detection with Generative Adversarial Networks for Multivariate Time Series*. http://arxiv.org/abs/1809.04758.

Li, D., Chen, D., Shi, L., Jin, B., Goh, J. & Ng, S.-K. (2019) *MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks*. https://github.com/LiDan456/MAD-GANs.

Li, J., Di, S., Shen, Y. & Chen, L. (2021a) FluxEV: A Fast and Effective Unsupervised Framework for Time-Series Anomaly Detection. In: *WSDM 2021 - Proceedings of the*

14th ACM International Conference on Web Search and Data Mining. 3 August 2021 Association for Computing Machinery, Inc. pp. 824–832. doi:10.1145/3437963.3441823.

Li, J., Izakian, H., Pedrycz, W. & Jamal, I. (2021b) Clustering-based anomaly detection in multivariate time series data. *Applied Soft Computing*. 100. doi:10.1016/j.asoc.2020.106919.

Liang, H., Song, L., Wang, J., Guo, L., Li, X. & Liang, J. (2021) Robust unsupervised anomaly detection via multi-time scale DCGANs with forgetting mechanism for industrial multivariate time series. *Neurocomputing*. 423, 444–462. doi:10.1016/j.neucom.2020.10.084.

Lim, B. & Zohren, S. (2021) Time-series forecasting with deep learning: A survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*.379 (2194). doi:10.1098/rsta.2020.0209.

Makridakis, S., Spiliotis, E. & Assimakopoulos, V. (2018) Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLoS ONE*. 13 (3). doi:10.1371/journal.pone.0194889.

Marjanovic, B. (2017) *Stock Market Dataset*. 2017. https://www.kaggle.com/borismarjanovic/price-volume-data-for-all-us-stocks-etfs?ref=hackernoon.com [Accessed: 20 November 2021].

Martinzez, C., Perrin, G., Ramasso, E. & Rombaut, M. (2018) *A deep reinforcement learning approach for early classification of time series*.

Mathieu, M. & Lecun, Y. (2017) *Unsupervised Learning Under Uncertainty*.

Munir, M., Siddiqui, S.A., Dengel, A. & Ahmed, S. (2019) DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series. *IEEE Access*. 7, 1991–2005. doi:10.1109/ACCESS.2018.2886457.

Nandanwar, H., Chauhan, A., Pathl, D. & Meena, H. (2020) *Proceedings of the 2nd International Conference on Inventive Research in Computing Applications (ICIRCA 2020) : 15-17 July, 2020*. In: 2020 p.

Ngo, C.P., Winarto, A.A., Li, C.K.K., Park, S., Akram, F. & Lee, H.K. (2019) *Fence GAN: Towards Better Anomaly Detection*. http://arxiv.org/abs/1904.01209.

Ngo Phuc Cuong (2021) *Fence-GAN code repository*. 2021. https://github.com/phuccuongngo99/Fence_GAN [Accessed: 25 April 2022].

O'leary, C. (2021) *Code repository - EMMV*. July 2021. https://github.com/christian-oleary/emmv [Accessed: 9 April 2022].

PortfolioLab (n.d.) *PortfolioLab*. https://portfolioslab.com/ [Accessed: 2 May 2022].

pyod (n.d.) *PyOD documentation*. https://pyod.readthedocs.io/en/latest/index.html [Accessed: 28 April 2022].

pytorch (n.d.) *Optimizing Model parameters*. https://pytorch.org/tutorials/beginner/basics/optimization_tutorial.html [Accessed: 1 May 2022].

pytorch.org (n.d.) *PyTorch*. https://pytorch.org/docs/stable/index.html [Accessed: 16 March 2022].

Rafiee, N., Gholamipoorfard, R., Adaloglou, N., Jaxy, S., Ramakers, J. & Kollmann, M. (2022) *Self-Supervised Anomaly Detection by Self-Distillation and Negative Sampling*. http://arxiv.org/abs/2201.06378.

Recht, B. (2019) A Tour of Reinforcement Learning: The View from Continuous Control. *The Annual Review of Control, Robotics, and Annu. Rev. Control Robot. Auton. Syst*. 2, 253–279. doi:10.1146/annurev-control-053018.

Sahoo, K., Samal, A.K., Pramanik, J. & Pani, S.K. (2019) Exploratory data analysis using python. *International Journal of Innovative Technology and Exploring Engineering*. 8 (12), 4727–4735. doi:10.35940/ijitee.L3591.1081219.

Sak, H., Senior, A. & Beaufays, F. (2014) *Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling*.

Salinas, D., Flunkert, V. & Gasthaus, J. (2017) *DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks*. http://arxiv.org/abs/1704.04110.

Schneider, T., Qiu, C., Kloft, M., Latif, D.A., Staab, S., Mandt, S. & Rudolph, M. (2022) *Detecting Anomalies within Time Series using Local Neural Transformations*. http://arxiv.org/abs/2202.03944.

Shi, S. (2021) *Anomaly Detection in stock prices*. 2021. https://medium.com/swlh/machine-learning-application-time-series-anomaly-detection-for-stock-day-trading-f7c3aba9a494 [Accessed: 1 May 2022].

Siddique, T., Mahmud, M.S., Keesee, A.M., Ngwira, C.M. & Connor, H. (2022) A Survey of Uncertainty Quantification in Machine Learning for Space Weather Prediction. *Geosciences (Switzerland)*.12 (1). doi:10.3390/geosciences12010027.

Sirignano, J. & Spiliopoulos, K. (2020) Mean field analysis of neural networks: A central limit theorem. *Stochastic Processes and their Applications*. 130 (3), 1820–1852. doi:10.1016/j.spa.2019.06.003.

Song, Y. (2021) *DeepAR AD*. 2021. https://github.com/aws-samples/amazon-sagemaker-anomaly-detection-with-rcf-and-deepar [Accessed: 2 May 2022].

The python foundation (n.d.) *Python*. https://www.python.org/doc/essays/blurb/ [Accessed: 16 March 2022].

Thoma, M. (2018) *Gradient Visualisation*.

Thompson, N.C., Greenewald, K., Lee, K. & Manso, G.F. (2020) *The Computational Limits of Deep Learning*. http://arxiv.org/abs/2007.05558.

Vakili, M., Ghamsari, M. & Rezaei, M. (2020) *Performance Analysis and Comparison of Machine and Deep Learning Algorithms for IoT Data Classification*.

Warzynski, A., Falas, L. & Schauer, P. (2021) Excess-Mass and Mass-Volume anomaly detection algorithms applicability in unsupervised intrusion detection systems. In: *Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE*. 2021 IEEE Computer Society. pp. 131–136. doi:10.1109/WETICE53228.2021.00035.

Wolfram MathWorld (n.d.) *Normal Distribution*. https://mathworld.wolfram.com/NormalDistribution.html [Accessed: 26 April 2022].

Wu, T. & Ortiz, J. (2021) *RLAD: Time Series Anomaly Detection through Reinforcement Learning and Active Learning*.

Yong, B.X. & Brintrup, A. (2022) *Bayesian autoencoders with uncertainty quantification: Towards trustworthy anomaly detection*. http://arxiv.org/abs/2202.12653.

Yu, M. & Sun, S. (2020) Policy-based reinforcement learning for time series anomaly detection. *Engineering Applications of Artificial Intelligence*. 95. doi:10.1016/j.engappai.2020.103919.

Zablocki, M. (2021) *RSRAE - pytorch repository*. 2021. https://github.com/marrrcin/rsrlayer-pytorch [Accessed: 1 May 2022].

Zha, D., Lai, K.H., Wan, M. & Hu, X. (2020) Meta-AAD: Active anomaly detection with deep reinforcement learning. In: *Proceedings - IEEE International Conference on Data Mining, ICDM*. 1 November 2020 Institute of Electrical and Electronics Engineers Inc. pp. 771–780. doi:10.1109/ICDM50108.2020.00086.

Zhang, A., Lipton, Z.C., Li, M. & Smola, A.J. (2020) *Dive into Deep Learning*. https://d2l.ai.

Zhang, X.F. (2006) Information uncertainty and stock returns. *Journal of Finance*. 61 (1), 105–137. doi:10.1111/j.1540-6261.2006.00831.x.

Zhong, C., Cenk Gursoy, m. & Velipasalar, Senem. (2019) *Deep Actor-Critic Reinforcement Learning forAnomaly Detection*.

Zou (2020) *RSRAE - original repository*. 2020. https://github.com/dmzou/RSRAE [Accessed: 1 May 2022].

Appendices

7.3 Appendix A – Project Plan

Task Name	Start Date	Due Date	Description
Project title	27/09/21	10/Oct/21	Finalise project idea and define an initial project title
Write Project Plan	27/09/21	17/Oct/21	Write project plan & definition
Submit Initial Project Plan	**NA**	**18/Oct/21**	**Submission Deadline**
Finalised title	18/Oct/21	31/Oct/21	Submit final project title in intranet
Reading Material	18/Oct/21	31/Oct/21	Gather relevant background reading materials in anomaly detection, machine learning, and finance
Introduction	25/Oct/21	07/Nov/21	Finish writing Section 1 - Introduction
Dataset	18/Oct/21	14/Nov/21	Gather data sources for the model and describe them in report
Literature review	31/Oct/21	28/Nov/21	Finish writing literature review based on background materials gathered
Background reading	*NA*	*28/Nov/21*	*Milestone*
Submit Interim Report	**NA**	**29/Nov/21**	**Submission Deadline**
Presentation	29/Nov/21	05/Dec/21	Prepare presentation slides and practice
Submit Progress Presentation	**NA**	**06/Dec/21**	**Submission Deadline**
Prepare dataset	14/Feb/22	06/Mar/22	Have the dataset in prepared in a format that is directly usable by torch
More background reading	21/Feb/22	13/Mar/22	More theory
Create Second Draft	20/Feb/22	20/Mar/22	Finalise second draft for submission
Submit Draft Report	**NA**	**21/Mar/22**	**Submission Deadline**
Improvements	28/Mar/22	17/Apr/22	Propose improvements to existing architectures
Experimentation	21/Mar/22	17/Apr/22	Training various models on the prepared dataset
Finished experimentation	*NA*	*17/Apr/22*	*Milestone*
Evaluation	21/Mar/22	24/Apr/22	Evaluate the results of my work and discuss next steps
Final edits	25/Apr/22	01/May/22	Spell-checking, formatting, references, etc.
Final Report	*NA*	*1/May /22*	*Milestone*
Submit Final Report	**NA**	**04/May/22**	**Submission Deadline**
Video showcase	1/May/22	05/May/22	Prepare project video showcase
Submit Project Video	**NA**	**06/May/22**	**Submission Deadline**

7.4 Appendix B – Risk Assessment

Risk	Likelihood	Severity	Impact	Prevention / Mitigation
Missed deadline	Low	High	• Late submission • Loss of marks	• Good time management
Changes to the project plan	Low	Medium	• Extra work with a shorter timeline	• Good project management
Lack of data	Medium	High	• Inability to build the intended model	• Ensuring the type of data is publicly available in enough quantity before the project
Lack of relevant academic literature	Medium	Medium	• Less rigorous literature review • Possibly worse built model	• Using all available resources / more search
Not understanding underlying materials	Low	Low	• Loss of time required to catch up	• Enough preparation/reading
Failure to create a working model	Low	High	• Project is likely failed or will receive very low marks	• Enough preparation/reading
Failure to achieve good model precision	Medium	Low	• Need to address the root cause • Re-evaluate expectations according to findings	• Set a realistic expectation according to relevant academic literature
Software/Hardware failures	Low	Low	• Slow down work • Change to plan	• Use dependable tools
Illness	Unknown	Unknown	NA	NA
COVID-19	Medium	Low	• Moving to full remote work	• Adhering to sensible safety measures as a society

7.5 Appendix C – Source Code

The source code for this project is published as a GitHub repository on the following URL- https://github.com/martin-sedlacek/ml-for-financial-data

A user manual is provided in the README. This should guide the reader through setting up a Jupyter Notebook environment on their machine and installing relevant dependencies. Afterwards, the experimental results can be reproduced by running the python notebook for a given model. After configuring the hyperparameters and selecting a dataset, simply click the 'restart and run all cells' button (furthest right in the top toolbar).

List of used source code repositories:

- DeepAnT (bmonikraj, 2020)
- MAD-GAN (Li, 2019; Guillem96, 2022)
- Fence-GAN (Ngo Phuc Cuong, 2021)
- RSRAE (Zablocki, 2021; Zou, 2020)
- EMMV (Goix, 2016a; O'leary, 2021)

7.6 Appendix D – Full experimental results

Results highlighted in **bold** are indicative of the model performing best according to a given criterion (e.g., F1 score).

Model	F1	pre	rec	EM	MV
DeepAnT_CNN	0.8584	0.8040	0.9207	5.74523645E-04	1.21194561E+03
DeepAnT_LSTM	0.8628	0.8033	0.9321	1.28390786E-03	2.52773061E+02
Fence-GAN	**0.9950**	**1.0000**	0.9904	**5.26178170E-03**	**8.25217296E-25**
MAD-GAN	0.8690	0.8029	0.9473	3.31108289E-03	2.67678564E+02
RSRAE	0.8584	0.8030	0.9223	9.99998963E-05	1.14543415E-24
HBOS	0.9210	0.8700	0.9787	4.47154641E-03	1.13134762E+04
LOF	0.8906	0.8030	**1.0000**	1.00000000E-04	1.12792954E+04
OC-SVM	0.8910	0.8037	0.9998	5.14263144E-03	1.12772960E+04

Table 5. kdd99 dataset - full EMMV, PR, and F1 results

Model	EM	MV
DeepAnT-CNN	2.51674072E-04	4.35155068E+03
DeepAnT-LSTM	2.80843712E-03	3.55248716E+02
Fence-GAN	**5.30835120E-03**	4.1744487E+02
MAD-GAN	3.92087090E-03	**1.5629242E+02**
RSRAE	2.449457634E-03	8.16958967E+02
HBOS	1.10182336E-04	2.2289540E+05
LOF	1.07276790E-04	2.2289540E+05
OC-SVM	1.06304217E-04	2.2289540E+05

Table 6. AAPL dataset - full EMMV results

Model	EM	MV
DeepAnT-CNN	2.55489543E-04	4.77246301E+03
DeepAnT-LSTM	2.19510197E-03	3.99221453E+02
Fence-GAN	**5.48160361E-03**	**1.90663367E+02**
MAD-GAN	1.33253293E-03	1.28212306E+03
RSRAE	2.16043994E-03	3.462405186E+02
HBOS	1.33747141E-04	2.021738742E+05
LOF	1.15826932E-04	2.021738742E+05
OC-SVM	1.14992563E-04	2.021738742E+05

Table 7. GM dataset – full EMMV results

Model	EM	MV
DeepAnT_CNN	2.03424126E-04	1.16563207E+03
DeepAnT_LSTM	3.96238391E-04	1.25879873E+03
Fence-GAN	**4.60295259E-03**	1.80201897E+02
MAD-GAN	1.30829368E-03	**1.06034889E+02**
RSRAE	2.03035943E-03	1.09032880E+02
HBOS	1.02570526E-04	5.12536632E+04
LOF	1.02816634E-04	5.12536632E+04
OC-SVM	1.02029491E-04	5.12536632E+04

Table 8. AXP dataset - full EMMV results

7.7 Appendix E – Proprietary Dataset Sources

Stock Market Dataset

The primary source of data is provided by Marjanovic (2017). It contains historical daily prices, traded volumes, and open interest for all US stocks and ETFs traded on the NYSE from the date of their public listing until *2017-11-10*.

Date	Open	High	Low	Close	Volume	OpenInt
1984-09-07	0.42902	0.42388	0.41874	0.42388	23220030	0
1984-09-10	0.42388	0.42516	0.41366	0.42134	18022532	0
1984-09-11	0.42516	0.43668	0.42516	0.42902	42498199	0

Table 9. AAPL stock data sample

Table 9. legend:

- Date: date in *YYYY-MM-DD* format
- Open: price on day open
- High: highest trade price during the day
- Low: lowest trade price during the day
- Close: price on day close
- Volume: total number of shares traded in a day
- OpenInt: number of outstanding derivative contracts that have not been settled (note that all the values in this column are 0, this feature is therefore obsolete).

Daily News Sentiment Dataset

Anonymous author on Kaggle under the alias Aaron7sun (2019) provides a history of daily news headlines along with a sentiment assessment for the Dow Jones Industrial Average (DJIA) index from *2008-06-08* to *2016-07-01*.

Date	Label	Top1…Top25
2008-08-08	0	b"Georgia 'downs two Russian warplanes' as countries move to brink of war"
2008-08-11	1	b'Why wont America and Nato help us? If they wont help us now, why did we help them in Iraq?'
2008-08-12	0	b'Remember that adorable 9-year-old who sang at the opening ceremonies? That was fake, too.'

Table 10. sentiment dataset sample

Table 10. legend:

- Date: date in *YYYY-MM-DD* format
- Label: indication of positive (1) or negative (0) sentiment in news headlines
- Top_n: 25 columns containing most popular news headlines

The 25 headline columns are obsolete as the overall sentiment is already provided in the *label* column.

S&P500 Index Dataset

David (2018) provides historic closing points data for the S&P500 index (SPX) from *1986* to *2018.* SPX can be treated as a general benchmark for the overall market.

Date	Close
02-01-86	209.59
03-01-86	210.88
06-01-86	210.65

Table 11. SPX dataset sample

Table 11. legend:

- Date: date in *DD-MM-YY* format
- Close: closing price on the day

7.8 Appendix F – Model Architectures and Full Parametrization

Full parametrization and layer structure used for the nets to achieve the results shown is seen below. Note that these are given for the financial dataset, slight changes in number of inputs/outputs and hidden dimensions are used for kdd99 without PCA.

A full excel sheet with recorded results for most of the empirical fine-tuning can be found in the project codebase under the experimentations folder.

DeepAnT CNN

Operation	C Input	C Output	Activation	Kernel	Dropout
Predictor					
Conv	7	64		1	0
MaxPool			ReLU	2	0
Conv	64	64		1	0
MaxPool			ReLU	2	0
Linear	64	40	ReLU		0.25
Linear	40	1			
Optimizer	Adam(lr=1E-05, wd =5E-06)				
Detector					
Anomaly Threshold		0.8			

Figure 22. DeepAnT CNN full parametrization for AAPL, GM, AXP , note that for kdd99 different input and output features are specified, larger kernel size can be used unless dimensionality reduction is applied during data load

DeepAnT LSTM

Operation	Input Dim.	Output Dim.	Layers	Dropout
Predictor				
LSTM	7	128	4	0.2
LSTM	128	64	4	0.2
Linear	64	7		0
Optimizer	Adam(lr=1E-05, wd =5E-06)			
Detector				
Anomaly Threshold		0.5		

Figure 23. DeepAnT LSTM full parametrization

Fence-GAN

Operation	Input Dim.	Output Dim.	Activation	Dropout
Generator				
Linear	30	64	ReLU	0.2
Linear	64	128	ReLU	0.2
Linear	128	7		0
Latent Dimension	30			
Encirclement α	0.5			
Dispersion β	30			
Optimizer	Adam(lr=1E-04, wd =1E-03)			
Discriminator				
Linear	7	256	LeakyReLU	0.1
Linear	256	128	LeakyReLU	0.1
Linear	128	64	LeakyReLU	0.1
Linear	64	1	Sigmoid	0
LeakyReLU Slope		0.1		
Optimizer (Discriminator)		SGD(lr=8E-06,wd=1E-03, momentum=0.9)		
Anomaly Threshold		0.5		

Figure 24. Fence-GAN full parametrization

MAD-GAN

Operation	Input Dim.	Output Dim.	Layers	Dropout
Generator				
LSTM	250	500	4	0.2
Linear	500	7		0.2
Latent Dimension	250			
Hidden Dimension	500			
Optimizer	Adam(lr=1E-05, wd =5E-07)			
Discriminator				
LSTM	7	500	4	0.1
Linear + Sigmoid	500	7		0.1
Optimizer		Adam(lr=1E-05, wd =5E-07)		
Anomaly Threshold		0.5		

Figure 25. MAD-GAN full parametrization

RSRAE

Operation	Input Dim.	Output Dim.	Activation
Encoder			
Linear	7	32	LeakyReLU
Linear	64	128	LeakyReLU
Linear	128	*D*	LeakyReLU
RSR Layer	*D*	*d*	
Decoder			
Linear	*d*	64	LeakyReLU
Linear	64	32	LeakyReLU
Linear	128	128	LeakyReLU
Linear	32	7	
d dimension	128		
D dimension	512		
LeakyReLU Slope		0	
Optimizer		AdamW(lr=1E-02) + Scheduler(wd=5E-07)	
Anomaly Threshold		0.8	

Figure 26. RSRAE full parametrization