Irving Fisher Committee on
Central Bank Statistics

BIS

12th biennial IFC Conference: "Statistics and beyond: new data for decision making in central banks"

22-23 August 2024

Variational autoencoders for multivariate time series outlier detection[1]

Tamara Fajt Mayer, Giovanni Raimondo Quaratino and Juan Francisco Javier Cordero Romero,
European Central Bank

[1] This contribution was prepared for the conference. The views expressed in this publication are those of the authors and do not necessarily represent the official views of the Committee, its members, or the BIS.

Variational autoencoders for multivariate time-series outlier detection

Tamara Fajt Mayer, Giovanni Raimondo Quaratino, Juan Francisco Javier Cordero Romero

List of Abbreviations

ARIMA	Autoregressive Integrated Moving Average
DSD	Data Structure Definition
ECB	European Central Bank
ELBO	Evidence Lower Bound
GAN	Generative Adversarial Network
IBSI	Individual Balance Sheet Items
KL	Kullback-Leibler
MFI	Monetary Financial Institution
MSE	Mean Squared Error
RIAD	Register of Institutions and Affiliates Database
VAE	Variational Autoencoders

Abstract: This paper shows how deep generative learning techniques can help in time series outlier detection for a large dataset. In particular, we employ those techniques on entity-level balance sheets of euro area banks. We show that variational autoencoders (VAEs), though more commonly used for data generation, can be effectively utilised in time series anomaly detection through a reconstruction error-based approach. Notably, VAEs are able to capture the complex temporal dependencies and high-dimensional patterns in the dataset while proving to be relatively fast to train and robust to noisy or missing data. This contrasts with traditional ARIMA-based methods which are less suitable for this dataset due to its size and which do not capitalize on its multivariate nature. We further demonstrate how the model can be integrated into a statistical production pipeline without a significant workload increase. This is facilitated by adjustable thresholds based on reconstruction error that limit the number of observations identified as outliers.

Keywords: *variational autoencoders, generative deep learning, multivariate time series, machine learning, statistical production systems, anomaly detection*

Contents

Introduction 5

Literature Review 5

Dataset overview 7

Methodology 8

- Foundational Concepts of Variational Autoencoders 8
  - Architecture 9
  - The reparametrization trick 10
  - Evidence Lower Bound and the Kullback-Leibler divergence 11
  - VAEs for anomaly detection 11
- Methods 11
  - Data Preprocessing 11
  - Handling Missing Values 11
  - Removing Duplicates and Irrelevant Data 11
  - Subsetting for Flow Data 11
  - Adding Categorical Variables Using Multi Hot Encoding 12
  - Creating the training splits 13
- Model Training 14
  - Reparameterization Trick and Forward Pass 14
  - Hyperparameter Optimization 15
- Extracting the Anomalies 16

Results 16

- Reconstruction error-based evaluation 17
- Normalization of reconstruction errors 19

Operationalizing VAEs in a production pipeline 20

- Pipeline development 20

Conclusion 21

Bibliography 22

Annex 23

- Hyperparameter tuning results 23
- IBSI Data Structure Definition (DSD) 24

Introduction

Central banks are entrusted with the critical responsibility of making evidence-based monetary policy decisions. To achieve this, they rely on increasing amounts of financial and monetary data. In the euro area, the Individual Balance Sheet Items (IBSI) dataset was introduced in the aftermath of the financial crisis to increase the granularity of available data by making bank-level balance sheet information available to the European Central Bank (ECB) decision-making bodies.

With the recent advancements in generative AI algorithms and their broad applications to large datasets, we set out to explore the feasibility of whether these could be employed for the outlier validation of the Individual Balance Sheet Items (IBSI) data. The size and complexity of the data posed practical challenges, necessitating the selection of a model that was easy and fast to train, as well as robust to missing data. Given the absence of any prior outlier detection algorithm used for this dataset and the consequent lack of available labels, an unsupervised model was required. Furthermore, we were interested in a model that could effectively capture the inherently multivariate nature of balance sheet data. Of significant importance was the need to constrain the number of flagged outliers in order not to overwhelm data producers at the national central banks. To this end, we sought to identify a model that could accurately detect anomalies while also allowing us to set thresholds that would limit the number of flagged outliers.

The findings presented in this paper indicate that variational autoencoders cover all these requirements and that they perform well in the challenging circumstances. We propose a statistical production pipeline that would allow findings to be shared with national central banks and to gather feedback on the detected outliers, thereby resulting in improved data quality and better decision-making.

The rest of this paper is organized as follows: Section 2 offers a literature review that contextualizes our approach within the existing body of research. Section 3 focuses on the IBSI dataset, giving an overview of the main characteristics of the data and its reporting population, while section 4 presents the methods used in the study, including the model architecture, preprocessing steps, training and hyperparameter tuning. Section 5 shows the key results and performance metrics. The incorporation of the model into the production pipeline is discussed in the sixth section. The seventh section concludes the paper with a discussion of the results and potential future applications.

Literature Review

Anomaly detection is a critical analytical process aimed at identifying patterns in data that do not conform to expected behaviour. These anomalies, or outliers, are often indicative of critical, non-normative events such as system faults, fraudulent activity, or emergent novel patterns (Chandola et al. (2009)). The significance of anomaly detection transcends various fields, from detecting network intrusions in cybersecurity to identifying rare diseases in healthcare (Ahmed et al. (2016)). In the financial and banking sectors, the detection of anomalies is of paramount importance due to the potential for such irregularities to signal financial fraud, operational disruptions, or market manipulation (Aleskerov et al. (1997)).

The complexity inherent to multivariate time series datasets introduces significant challenges in anomaly detection. Ding et al. (2014) argue that the high dimensionality

of the data can mask the distinction between noise and true anomalies, and Hyndman and Athanasopoulos (2018) note that temporal dependencies within the data require sophisticated models to account for lagged effects and evolving trends over time. Additionally, these challenges are further compounded by the varying practices and reporting standards of financial institutions and ever-changing nature of financial markets conditions, necessitating adaptive and robust anomaly detection methods capable of handling the complexities of multivariate financial time series data (Aggarwal (2013)).

Over the last decades, traditional methods for identifying anomalies in time series data have often relied on statistical models that assume some underlying distribution or pattern within the data. One of the earliest and most straightforward approaches is the use of threshold-based rules proposed by Basseville and Nikiforov (1993), where values falling outside predefined bounds are flagged as anomalies. These bounds can be set using measures such as the mean and standard deviation of historical data. More sophisticated methods include time series decomposition, where data is separated into trend, seasonal, and residual components, with anomalies identified in the residual component (Hyndman and Athanasopoulos (2018)).

Autoregressive Integrated Moving Average (ARIMA) models, along with their variations, enable the forecasting of future values and the detection of outliers through the examination of variances from the predicted values (Box et al. (2015)). They have seen wide application in various domains and serve as the outlier detection methods for the country-level balance sheet datasets administered by statistical compilers at the ECB. Additionally, in the context of financial and banking data, the use of statistical quality control visualization methods such as the Shewhart control chart or the Cumulative Sum (CUSUM) chart has been prevalent (Montgomery, 2009). Such methods have proven particularly useful for detecting shifts in the mean or variance over time, which can indicate fraudulent activities or operational inconsistencies.

However, the application of traditional approaches for anomaly detection in multivariate financial time series data can often be challenged by the high dimensionality and complex temporal dependencies of the datasets. Moreover, the inherent constraints of these methods, due to strong assumptions about data distribution, often present a challenge when analysing non-linear data. Due to these limitations, and as a result of the increase of the computational capacity, substantial research has been developed during the last years in the field of unsupervised learning for anomaly detection. Li and Jung (2023) argue that, in contrast to traditional and supervised learning methods, deep learning-based anomaly detection approaches learn hierarchical discrimination features from the time series data and therefore eliminates the need for domain experts to manually develop features.

Variational Autoencoders (VAEs) fall within this spectrum of deep learning methods employed for anomaly detection. VAEs are a class of deep learning models that leverage neural networks to encode high-dimensional data into a lower-dimensional, latent representation, from which the original data can be reconstructed (Doersch (2016)). Their strength lies in their probabilistic foundations; they do not merely learn a deterministic mapping, but rather model the data generation process as a probability distribution, enabling them to capture the underlying data distribution and generate new samples that are similar to the input data (Rezende et al. (2014)).

This feature is particularly relevant for anomaly detection, as VAEs can learn to reconstruct normal data accurately while struggling to do the same for anomalous data, thus allowing anomalies to be identified through the reconstruction error (An and Cho (2015)).

In order to have a comprehensive overview of the deep learning models currently employed for anomaly detection, Generative Adversarial Networks (GANs) need to be included in the discourse. GANs have emerged as a compelling alternative, leveraging adversarial techniques to generate synthetic data and identify anomalies by assessing deviations from this generated norm. The MAD-GAN model, as detailed by Li et al., employs a GAN architecture tailored for multivariate time series, showcasing its potential to detect intricate anomalies through adversarial training (Li et al., 2018). VAEs, still, could be preferred due to their robust probabilistic framework and their ability to measure reconstruction errors, which directly assist in anomaly identification: this reconstruction error provides a quantitative basis for anomaly detection and can be more intuitive and stable compared to the discriminative models used by GANs, where stability and mode collapse can be significant challenges.

Dataset overview

Our input data consists of the Individual Balance Sheet Items (IBSI) dataset transmitted to the European Central Bank, containing confidential statistical data at monthly frequency. The dataset encompasses approximately 300 balance sheet items reported by around 2850 banks resident in the euro area.[1] The items include bank deposits with varying maturities and originating from different sectors, debt securities issued by credit institutions, equity, shares, and securities held by banks; capital reserves of banks, loans extended by credit institutions to both private and public sectors, and external assets and liabilities, among others.

Table 1: Main assets and liabilities reported in IBSI

ASSETS	**LIABILITIES**
• Cash • Loans to euro area residents (MFIs, General Government, Households, NFCs, OFIs, IF, ICPF) • Debt securities held (MFIs, General Government, Private Sector) • Money market fund shares • Equity and other investment funds shares • External assets	• Deposits placed by euro area residents (MFIs, Central Government, Households, NFCs, OFIs, IF, ICPF, Other General Government) • Debt securities issued • Capital and Reserves • External liabilities

The transmission of the IBSI dataset to the ECB from national central banks commenced in 2012, following the approval by the ECB Governing Council, back data were provided from July 2007 onwards. The dataset initially comprised a panel of around 250 credit institutions which were identified as the most relevant for a meaningful coverage of monetary statistics at a national and euro area level. The panel has since been expanded several times, most recently in June 2020 when the

[1] This represents a subset of the balance sheet items and the reporting population covered by ECB Regulation (EU) 2021/379 on the balance sheet items of credit institutions and of the monetary financial institutions sector (recast) (ECB/2021/2). This Regulation is the primary source for the compilation of euro area monetary statistics.

Governing Council decided to extend the coverage to all euro area credit institutions, except those that are not subject to derogated reporting requirements due to their small size, or have total assets of EUR 100 million or less.[2] Regular monthly reporting of these data commenced with the reference period March 2021, and in addition back data were provided to (at least) December 2019. The coverage of the dataset in terms of numbers of reporting agents and total assets is provided in in Figure 1.
IBSI currently uses a dynamic sample of banks that is updated on a monthly basis, using the Register of Institutions and Affiliates Database (RIAD) as its reference data.

Consequently, changes in the euro area banking landscape—such as the

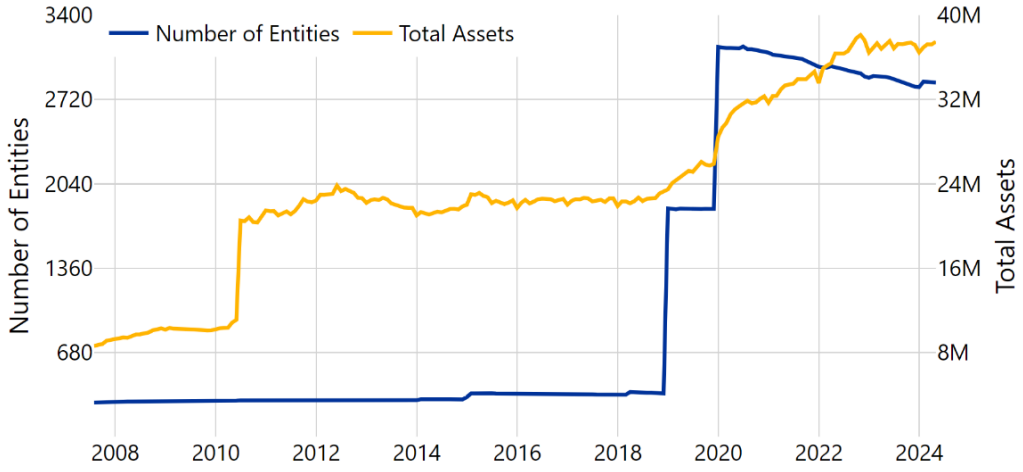


Figure 1: Coverage of IBSI dataset over the years, across number of entities and total assets (in millions of euros)

establishment of new credit institutions, revocation of banking licenses, and corporate restructuring through mergers and acquisitions—directly influence the composition of the reporting population. Because of the fluctuating nature of the sample, the availability of data over time can vary across entities. Thus, we consider in our study all possible items for every period for which they are available, and we replace the unavailable values with zeros with the aim of obtaining a rectangular dataset.
Furthermore, since our focus is on the anomaly detection during the stage of data reception, we exclusively conduct the study on raw data transmitted by national central banks, thus excluding any derived series compiled by the ECB. Hence, our final sample comprises a longitudinal set of 164 balance sheet items reported by approximately 2850 entities across a date range spanning 158 distinct periods, from July 2007 to January 2024.

Methodology

Foundational Concepts of Variational Autoencoders

Variational Autoencoders are a class of generative models that have significantly influenced the field of unsupervised learning. The key innovation of VAEs lies in their

[2] The threshold of EUR 100 million is applicable on request of the relevant national central bank, with permission of the ECB Governing Council.

ability to model complex data distributions using a probabilistic framework, which facilitates the generation of new data points that resemble the training data, and in their ability to learn complex data distributions, VAEs have emerged as a powerful tool in the scope of anomaly detection. The fundamental idea is to train the VAE on enough data to allow the model to learn to represent normality accurately. Anomalies can then be detected based on reconstruction errors or deviations in the latent space. This section provides an in-depth look at how VAEs are applied to anomaly detection.

Architecture

The architecture of a VAE consists of two primary components.

1. **Encoder**. The encoder network maps the input data x to the parameters of the variational distribution $q(z|x)$. Typically, the encoder outputs the mean and log-variance of a Gaussian distribution, enabling the sampling of latent variables z.
2. **Decoder**. The decoder network maps the latent variables z back to the data space, reconstructing the input data. The decoder outputs the parameters of the likelihood distribution $p(x|z)$.

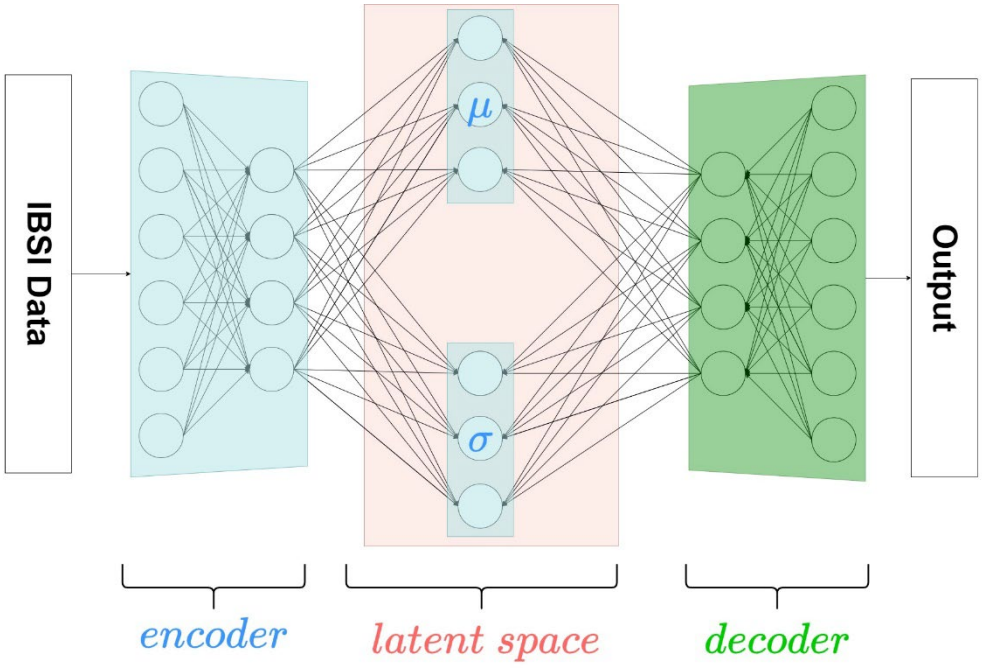


Figure 2: Variational autoencoder architecture

To make greater sense of this architecture, it is necessary to underline that at the heart of VAEs is the concept of latent variable models. In these models, the observed data is assumed to be generated from some unobserved latent variables through a probabilistic process.

The generative process in a VAE first requires the sampling of a latent variable z from a prior distribution $p(z)$, typically chosen to be a standard normal distribution $\mathcal{N}(0, I)$. This choice is strategic: it simplifies the learning process, as the standard normal distribution is mathematically tractable and symmetrical, making it easier to handle

within the loss functions used during training. Moreover, utilizing a standard normal distribution aligns with the assumption that the latent variables z are independent and identically distributed, which aids in the stabilization of the model's training phase. This design choice is crucial because it directly impacts the ability of the encoder to effectively learn a meaningful and compact representation of the input data. By constraining the latent space to a standard normal distribution, the VAE encourages the encoder to compress variations in the input data into a fixed range of values, facilitating the decoder's ability to reconstruct the original input accurately from these condensed representations. Furthermore, this constraint ensures that the latent space does not grow arbitrarily large or complex, which would make the model less generalizable and more prone to overfitting. Additionally, sampling from a standard normal distribution facilitates the use of the reparameterization trick, a key technique in VAE training that allows for gradient-based optimization of stochastic nodes. This trick involves expressing the random variable as a deterministic variable that is differentiable with respect to the parameters of the distribution, thus enabling efficient backpropagation and more stable training outcomes.
The observed data x is generated from the conditional distribution $p(x|z)$.The primary challenge in latent variable models is the intractability of the posterior distribution $p(z|x)$, which is required for inference. VAEs address this challenge using variational inference, which approximates the true posterior distribution with a variational distribution $q(z|x)$. The objective is to make $q(z|x)$ as close as possible to $p(z|x)$.
The sampling of z in the encoder is made differentiable using the reparameterization trick.

The reparametrization trick

The reparameterization trick is a crucial technique employed in Variational Autoencoders to facilitate the efficient training of the model through gradient-based optimization. As stated, in the VAE framework, the encoder network approximates the posterior distribution $q(z|x)$ over the latent variables z given the input data x. Typically, this posterior is assumed to follow a Gaussian distribution characterized by a mean μ and a standard deviation σ. Directly sampling z from this distribution during training poses a challenge because it introduces stochasticity that makes backpropagation infeasible.
To overcome this, Kingma and Welling (2014) introduced the reparameterization trick, which enables the differentiation of the stochastic sampling process. The trick involves expressing the sampled latent variable z as a deterministic function of the mean μ, the standard deviation σ, and an auxiliary noise variable ϵ drawn from a standard normal distribution $\mathcal{N}(0, I)$. Specifically, the reparameterization is given by:

$$z = \mu + \sigma \cdot \epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, I).$$

This transformation allows the gradient of the loss function to be backpropagated through the deterministic nodes of the network, ensuring that the optimization process remains tractable. By shifting the randomness from the distribution parameters μ and σ to the noise variable ϵ, the reparameterization trick effectively separates the stochastic and deterministic components of the model. This separation is crucial for VAE's ability to learn meaningful latent representations through gradient descent.

Evidence Lower Bound and the Kullback-Leibler divergence

In the training phase, VAE is trained on a dataset containing instances of the data, optimizing the Evidence Lower Bound (ELBO) to balance reconstruction loss and Kullback-Leibler (KL) divergence. The ELBO is derived as follows:

$$\log p\,(x) \geq E_{q(z|x)}[\log p\,(x|z)] - \mathrm{KL}\big(q(z|x)|p(z)\big).$$

Here, the first term is the reconstruction loss, which measures how well the model can reconstruct the input data from the latent variables. The second term is the KL divergence between the variational distribution $q(z|x)$ and the prior $p(z)$, which acts as a regularizer to ensure that $q(z|x)$ does not deviate too much from the prior.

VAEs for anomaly detection

After training, the VAE will reconstruct the training data well but fail to reconstruct anomalies effectively. The reconstruction error, which is the difference between input and output, serves as an indicator of anomaly. Additionally, anomalous data points often lie in regions of the latent space that are sparsely populated or far from the regions occupied by normal data points. Analysing the density or the distance in the latent space can help identify anomalies.

Methods

Data Preprocessing

Before training our VAE model, a series of preprocessing steps were performed to ensure the quality and consistency of the input data. The preprocessing was implemented using PyTorch, with custom transformations applied to the dataset.

Handling Missing Values

Firstly, we addressed missing values in the dataset. The original data contained special codes indicating non-available data (NC) and no data (ND). We replaced NC values with 0 to denote a lack of activity, and ND with NaN to indicate truly missing data points.

Removing Duplicates and Irrelevant Data

Subsequently, we removed duplicate entries from the dataset to prevent data leakage by biasing our model with redundant information. As previously mentioned, the dataset incorporates the IBSI series key as index. A series key serves as a unique identifier for each time series and follows the definitions from the corresponding SDMX data structure definition (DSD). Practically speaking, the series keys in our dataset are composed of 12 dimensions separated by periods. During the preprocessing, we focused on the fifth dimension, which represents the unique identifier of the reporting bank and grouped the data accordingly by bank.

Subsetting for Flow Data

Balance sheet information is typically categorized into two main data types: stocks, which represent the financial position at a specific point in time, and flows[3], which capture the changes in financial positions over a period. To streamline the dataset for our analysis and make it easier for our model to process, we converted all stock data

[3] Flows are derived as changes in stocks, and where available, also taking into account adjustments.

into flow data. This allowed us to maintain uniformity across the dataset, focusing solely on flow data to track the temporal dynamics.

Adding Categorical Variables Using Multi Hot Encoding

For the task at hand, incorporating categorical variables effectively is also a crucial step for capturing the underlying patterns in the data. As already mentioned, the dataset is structured with balance sheet item series keys as indices and reference periods as columns, at monthly frequency for each balance sheet item series key. Each series key, such as "M.XX.N.A.YY.L40.A.1.Z5.0000.Z01.E " represents a specific balance sheet component of a bank (resident in country XX and with identifier YY) and is parsed to extract its breakdowns. In particular, the sixth dimension, which signifies a specific item, e.g. "L40" (debt securities issued by MFI YY), is encoded using Multi Hot Encoding.

Multi Hot Encoding, also known as Binary Encoding, is a technique designed to balance the trade-offs between One-Hot Encoding and Ordinal Encoding. Initially, the categorical variables are ordinally encoded, converting them into integer values. These integer values are then transformed into binary representations. Each digit of the binary number is treated as a separate feature, resulting in multiple binary columns per categorical variable. Mathematically, if a categorical variable C with N unique values is encoded, each value c_i is first mapped to an integer k_i, which is then converted to its binary form. The resulting binary digits form new columns b_{ij}, where b_{ij} in $\{0, 1\}$.

For example, given a categorical variable which represents three distinct balance sheet items ["L21", "L22", "L23"], the ordinal encoding might map them to [0, 1, 2]. Converting these to binary gives [00, 01, 10], respectively. These binary digits are then split into separate columns. This approach reduces the dimensionality introduced by One-Hot Encoding while avoiding the unintended ordinality imposed by Ordinal Encoding, thus mitigating the risk of creating misleading patterns in the latent space representation of the VAE. The formal representation of the binary encoding process can be described as:

$$C \rightarrow \text{Ordinal Encoding} \rightarrow k_i \rightarrow \text{Binary Conversion} \rightarrow b_{ij},$$

where C is the set of categorical values, k_i are the ordinally encoded integers, and b_{ij} are the binary digits forming the new feature columns. Multi Hot Encoding is particularly advantageous when dealing with high-cardinality categorical variables, as it efficiently reduces the number of dimensions, preserving the meaningfulness of the data without introducing spurious relationships. This method is therefore beneficial for our use case as it contributes to enhancing the VAE's capacity to detect anomalies.

Table 2: Multi Hot Encoding of IBSI dataset

Dataset (Untransformed)		
Index	Balance Sheet Item	...
0	A2C	...
1	A20	...
2	L22	...
3	L40	...
4	A51	...
5	L21	...
6	L7C	...
7	A10	...
8	L2A	...
9	A7A	...

Dataset (Multi Hot Encoded)						
Index	Balance Sheet Item	Balance Sheet Item_0	Balance Sheet Item_1	Balance Sheet Item_2	Balance Sheet Item_3	...
0	A2C	0	0	0	0	...
1	A20	0	0	0	1	...
2	L22	0	0	1	0	...
3	L40	0	0	1	1	...
4	A51	0	1	0	0	...
5	L21	0	1	0	1	...
6	L7C	0	1	1	0	...
7	A10	0	1	1	1	...
8	L2A	1	0	0	0	...
9	A7A	1	0	0	1	...

1. The number of modalities of the column "Balance Sheet Item" (10) is converted to binary digits (1010).
2. The number of binary digits (4) is the number of new columns needed.
3. The modalities (0-9) are encoded to binary (0000-1001).

Creating the training splits

We begin this process by addressing the challenge of temporal consistency, a key aspect of time series data, through a padding strategy. This involves appending zeros to sequences that do not meet the required length, thus standardizing the input size for our Variational Autoencoder (VAE) and allowing it to effectively learn the temporal dynamics. To accommodate the large range of financial figures present in balance sheet items, we apply a StandardScaler for normalization, ensuring that each feature is weighted equally in the model's learning process.

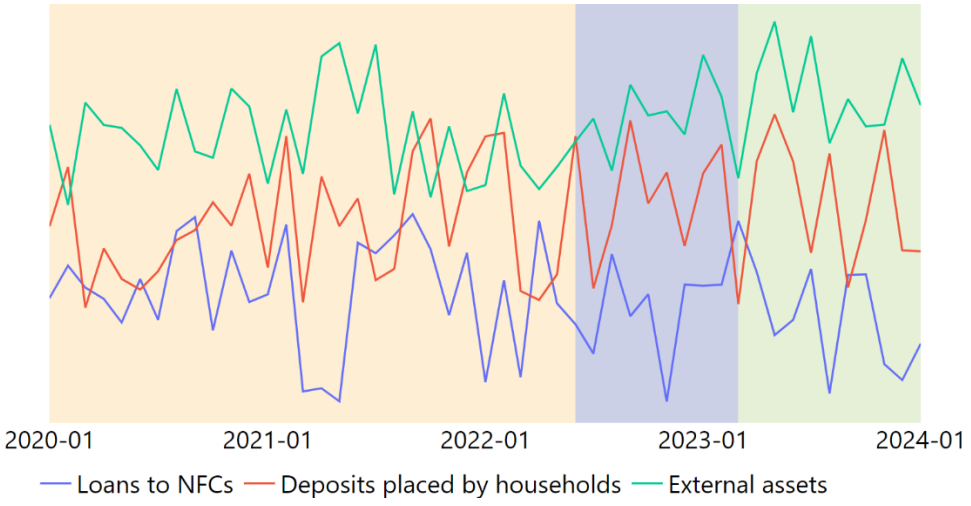


Figure 3: Example of train/validation/test sequential split with sliding window for one MFI

Considering the dual nature of our dataset, which includes both MFI-specific cross-sectional characteristics and temporal dependencies, we implemented a sliding window of 12 months to capture seasonal patterns and trends. This window slides along the dataset one month at a time, creating overlapping sequences that are then reshaped to fit the VAE's input structure. Each sequence becomes an independent observation, preserving the temporal order and preventing any leakage of future information into the training process. By splitting the data in this way, we ensure that the model is trained on data that accurately reflects the sequential nature of a real-world production scenario. For a high-level summary of the steps, it is possible to refer to Figure 4, containing the sequence diagram of the steps needed to generate the training splits.

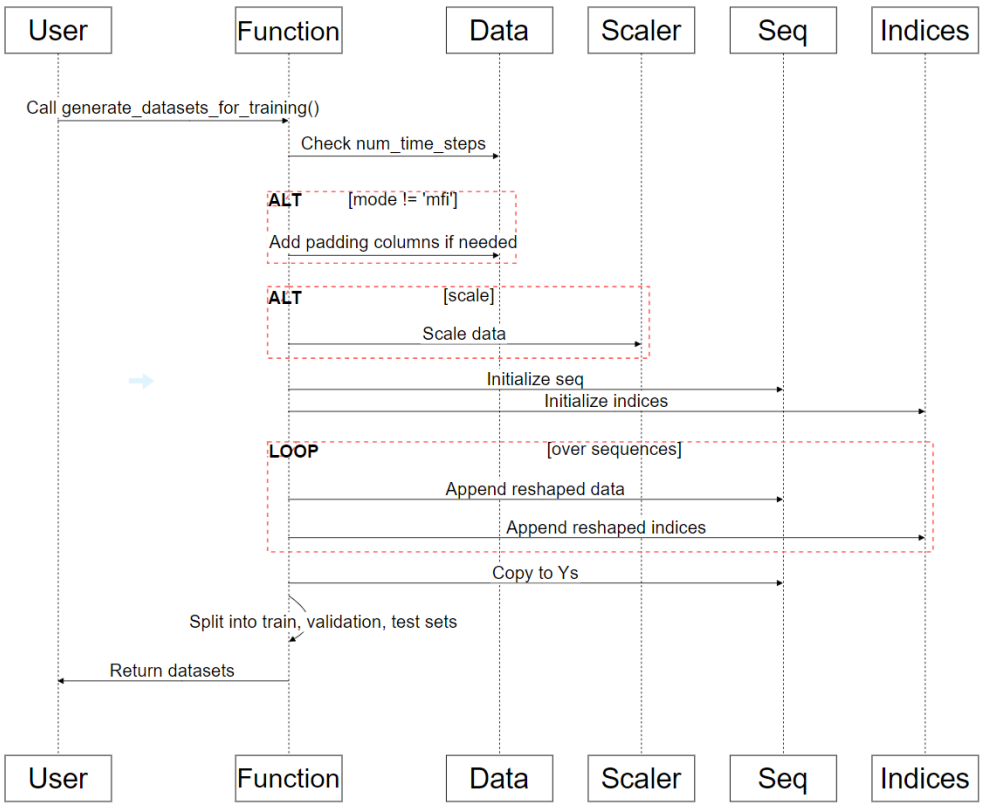


Figure 4: High level view of the steps needed to generate the training splits

Model Training

Reparameterization Trick and Forward Pass

In training a Variational Autoencoder (VAE), the reparameterization trick is a core component that facilitates backpropagation through stochastic nodes. The trick allows for the generation of a latent representation z by sampling from the latent space distribution, using computed mean and log variance vectors. During the forward pass, the input data x undergoes encoding to the latent space, yielding the mean and log variance parameters. Utilizing the reparameterization trick, the latent vector z is sampled and then decoded to reconstruct the input data. The VAE model outputs this reconstructed data alongside the latent space parameters, which are integral for calculating the loss function that guides model training. This process

ensures a differentiable pathway for the gradients to flow, enabling the VAE to learn the intricate structure of the data through its latent representations.

Hyperparameter Optimization

For the task at hand, we have designed our Variational Autoencoder (VAE) architecture within the PyTorch framework. Additionally, we dynamically configured this architecture using the hyperparameter optimization tool Optuna.

The number of layers in the encoder and decoder are determined by separate hyperparameters. For each layer in the encoder and decoder, the number of units are also hyperparameters, with the range set adaptively based on the size of the previous layer. Specifically, for each layer, the number of units is chosen between $\max\left(\frac{\text{previous\_layer\_size}}{128}, 12\right)$ and $\max\left(\frac{\text{previous\_layer\_size}}{4}, 48\right)$. After each linear layer, we introduce a non-linearity using the Rectified Linear Unit (ReLU) activation function, except for the last layer of the encoder, which directly outputs the parameters for the latent space distribution.

The encoder and decoder were optimized separately with the constraint that the number of decoder layers should remain smaller or equal to the number of encoder layers due to the fact that if the decoder in a VAE is too powerful, this can lead to overfitting. In such cases the decoder learns to reconstruct the training data too well, including its noise and anomalies. This can result in a loss of generalization, where the model performs poorly on unseen data. Additionally, as proposed by Tang et al. (2018), a too highly capable decoder might ignore the latent representation provided by the encoder, undermining the VAE's ability to learn meaningful and generalizable latent features, which is critical for tasks like data generation and anomaly detection. In our experiments we achieved the best performance when the model architecture comprised 10 encoder layers and 7 decoder layers.

Table 3: Model hyperparameters

Parameter	Type	Range
Number of layers in the encoder	Integer	[3, 12]
Number of units in each encoder layer	Integer	Dynamically calculated based on the size of the previous layer
Dropout rate for each encoder layer	Float	[0, 0.2]
Number of layers in the decoder	Integer	[3, n_encoder_layers]
Number of units in each decoder layer	Integer	dynamically calculated based on the size of the previous layer
Dropout rate for each decoder layer	Float	[0, 0.2]
Optimizer	Categorical	[Adam, SGD, Adagrad, AdamX, RMSProp]
Learning rate	Float (log scale)	[1e-5, 1]
Weight decay (L2 penalty)	Float	[0, 0.1]
Batch size	Categorical	[16, 32, 64, 128, 256, 512]

To further mitigate the risk of overfitting, we employ dropout regularization in both the encoder and decoder networks. The dropout rate for each layer is selected from

a uniform distribution within the range between 0 and 0.2. The output layer of the decoder is designed to match the dimensions of the input data.
Optuna is employed to optimize the hyperparameters of the VAE. It is an optimization framework that automates the search for the best hyperparameters by exploring the defined search space and evaluating the model performance. This automated search is crucial for enhancing VAE's ability to model complex time series data effectively.
The adoption of Optuna for hyperparameter tuning provides a systematic approach for identifying the optimal VAE architecture for our specific anomaly detection task. The optimization process aims to strike a balance between model complexity and generalization capabilities, ensuring robust performance. Table 3 displays the list of the hyperparameters considered for the tuning problem.

Extracting the Anomalies

Once the VAE has been trained, the next step is to utilize it for anomaly detection. The VAE is designed to reconstruct the input data while learning the underlying distribution. During inference, we leverage VAE's reconstruction capabilities to identify anomalies. We generate a reconstructed version of the input data and compare it against the actual data. Anomalies are detected by measuring the reconstruction error, which is the discrepancy between the reconstructed data and the real data.
To quantify this error, we calculate the mean squared error (MSE) between the reconstructed data points and the actual data points. We opt to use MSE because it quantitatively measures the squared discrepancies between the actual data points and their reconstructed counterparts, emphasizing larger errors more significantly than smaller ones. This characteristic is crucial for anomaly detection as it helps in accentuating outliers or anomalies which are typically represented by larger reconstruction errors. By squaring the differences, the MSE loss amplifies the impact of these anomalies on the overall model training, allowing the VAE to become more sensitive to deviations from the norm in the data distribution. This sensitivity is key for effectively identifying and quantifying anomalous data points during the model's inference phase.
The final stage in the anomaly detection process involves establishing a threshold to flag significant discrepancies as anomalies. We determine this threshold by selecting a high quantile of the weighted errors, such as the 99.9th percentile, which ensures that only the most extreme cases are flagged. By applying this quantile-based threshold, we capture the most significant deviations that are likely to represent true anomalies rather than mere noise or minor inconsistencies. The flagged anomalies are then compiled into a dataframe for further examination.

Results

As discussed in the methodology section of this paper, we use reconstruction error to determine which of our data points are outliers. As no prior outlier detection procedure was in place for this dataset, we do not have access to any labels indicating which data points should be considered anomalies. This absence of ground-truth anomaly labels necessitated the employment of alternative evaluation strategies to assess the effectiveness of our model in identifying anomalous patterns within the data.

We split the evaluation of the outputs produced by the VAE implementation into two parts:

- The initial phase of evaluation relied on analysing the performance of our model based solely on the **reconstruction error** it produced. This method provided a preliminary indication of anomalies based on how well the model could reconstruct the input data. For this phase, we used **synthetically generated data points** with known outliers. This allowed for the application of standard evaluation metrics such as precision and recall, providing a quantifiable framework for performance assessment.

- Once we were convinced the model was sufficiently capturing the anomalous data, we focused on a more **practical and application-oriented evaluation**. This phase involved the calibration of the **volume-weighted thresholds** to ensure that the flagged anomalies are the most relevant for practical financial analysis by excluding low-volume data points and reducing false positives.

Reconstruction error-based evaluation

As is generally the case when performing outlier detection, anomalies represent a very small fraction of data points in the IBSI dataset. To simulate this, we added a new reference period to our test dataset, duplicating the values for the previous period, with minor deviations (randomly distributed between 0.1% and 10% difference in absolute terms). These values represent our "normal", non-anomalous, data.
We then randomly selected between 0.01% and 0.1% of data points and introduced anomalies to them. This was done by multiplying the value of the previous period by 3 to 20 times the standard deviation of the series. Precision (P), Recall (R), and F1 score (F1) were used to evaluate anomaly detection performance:

$$\boldsymbol{P} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FP}}, \qquad \boldsymbol{R} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FN}}, \qquad \boldsymbol{F1} = \boldsymbol{2} \times \frac{\boldsymbol{P} \times \boldsymbol{R}}{\boldsymbol{P} + \boldsymbol{R}},$$

where *TP* is the number of true positives, *TN* the number of true negatives, *FP* the number of false positives and *FN* the number of false negatives. We used these metrics instead of the commonly used ROC-AUC[4] score due to how highly imbalanced our dataset is and our focus on avoiding false positives.
The selected reconstruction error threshold has a substantial effect on the performance of our VAE model. Tuning the reconstruction error also helps in achieving the desired balance between precision and recall. When the reconstruction error threshold is decreased, the model performs better on recall but at the expense of precision, making the model more sensitive to anomalies. On balance, the decision was made in favour of keeping precision high to reduce the high burden on statistical compilers of dealing with falsely identified anomalies.

[4] ROC-AUC stands for Receiver Operating Characteristic - Area Under the Curve, a metric used to evaluate binary classification models. The ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings. The AUC (Area Under the Curve) quantifies the overall ability of the model to discriminate between positive and negative classes, ranging from 0 to 1. With 1 indicating perfect classification.

Table 4 shows how different reconstruction error thresholds impact the precision and recall performance of our model on the data with synthetic outliers.

Table 4: Impact of reconstruction error on precision and performance

	Threshold > 95%			Threshold > 99.35%			Threshold > 99.9%		
Label	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
False	1.00	0.96	0.98	1.00	1.00	1.00	0.99	1.00	1.00
True	0.14	0.94	0.24	***0.98***	***0.89***	***0.93***	1.00	0.12	0.21

Figure 5 shows the distribution of the reconstruction loss on our test data with synthetic outliers and the selected reconstruction error threshold (99.35 percentile).

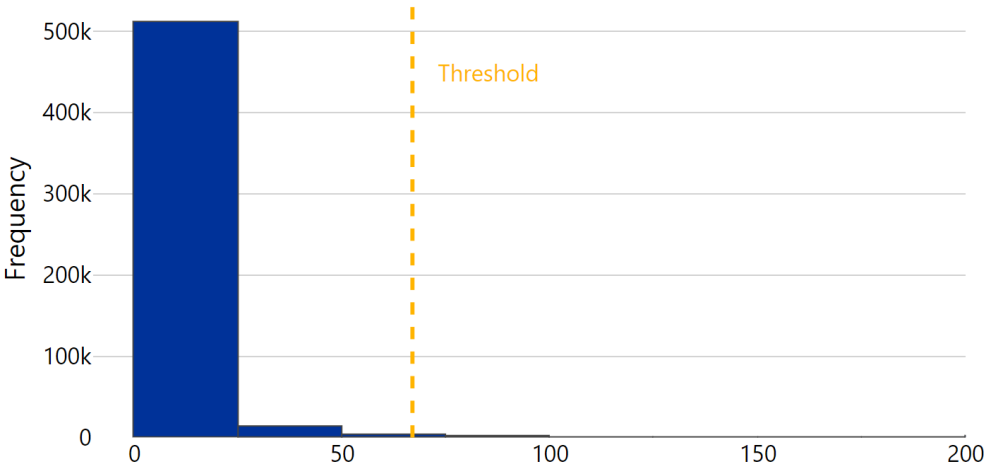


Figure 5: Histogram of reconstruction error

When assessing viability of using VAE for outlier detection for the IBSI dataset we were also interested in comparing the performance as well as training and evaluation time of our new model against the existing ARIMA implementation.
Table 5 shows the performance comparison between ARIMA and VAE on synthetic outliers[5] for one reference period, containing 529932 datapoints.

Table 5: Performance comparison between ARIMA and VAE

	ARIMA			VAE		
Label	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
False	1.00	0.99	0.99	1.00	1.00	1.00
True	0.15	0.57	0.24	***0.95***	***0.89***	***0.92***
Duration	16 hours 38 minutes			10 minutes[6]		

The comparison shows significant differences in both time efficiency and performance metrics. The ARIMA procedure struggled with the dataset's scale and took considerably longer to process one reference period of the IBSI dataset, with a duration of 16 hours and 38 minutes, while successfully completing the outlier

[5] Comparison performed on a subset of data for twelve countries, due to issues with running the current implementation of the ARIMA algorithm for the remaining eight euro area countries.

[6] Includes both training and inference, using Nvidia T4 GPU.

detection on only twelve out of the twenty euro area countries. In contrast, the VAE model completed both training and inference in just 10 minutes, demonstrating a substantial improvement in time efficiency.
Performance-wise, the VAE model outperformed ARIMA in detecting synthetic point outliers. While ARIMA showed decent recall, its precision was quite poor. VAE, on the other hand, achieved perfect precision and recall scores for non-outlier points (labelled False) and significantly higher precision and recall for outlier points (labelled True). The F1 score, which balances precision and recall, was also much higher for the VAE, with a high score of 0.92 compared to ARIMA's 0.24 for outlier detection.
These results suggest that the VAE model is not only faster but also more accurate and reliable for outlier detection in the IBSI dataset compared to the existing ARIMA implementation.

Normalization of reconstruction errors

Having established that our model can reconstruct the IBSI dataset accurately, we now aim to establish an implementation of variational autoencoders for IBSI outlier detection that can be used in statistical production. Our main objective was ensuring that the outputs produced by our new process do not unnecessarily increase the workload of national central banks. In this context, it is essential to assess the significance of the flagged anomalies relative to their volume. The goal is to direct the NCBs' attention to discrepancies that could indicate underlying issues of substantial importance or necessitate corrective action.
This step involves weighing the reconstruction error of each series relative to the total assets volume of the respective country. To ensure a balanced identification of outliers across countries, we normalized the volume share per country, since we aimed to achieve a proportionate number of outliers relative to the number of series (and banks) per country, while preventing an excessive count of outliers in countries with a larger number of series:

$$E_{weighted} = E(1 + 100\alpha^2),$$

where E is the reconstruction error and α is the normalized share of the anomaly. This

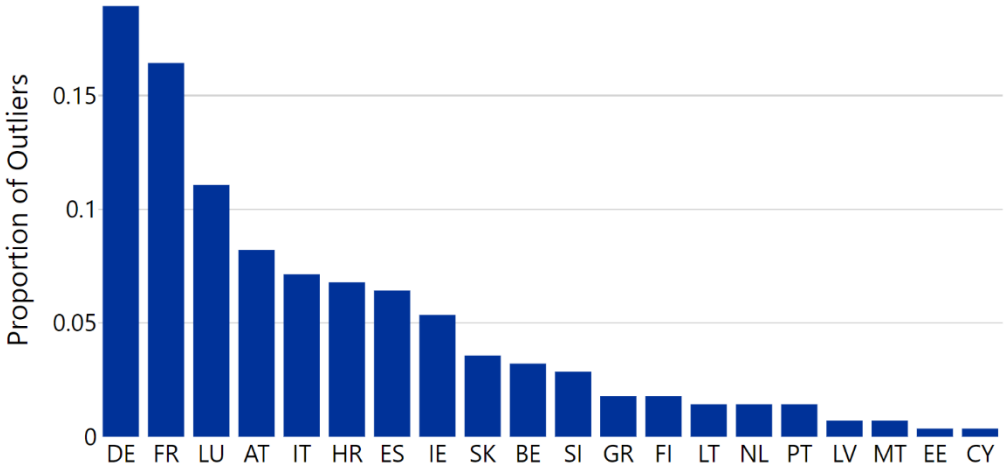


Figure 6: Proportion of flagged outliers by reference area

normalization process is designed to maintain outlier detection consistency and avoid

overwhelming countries with more data series with an unmanageable number of outliers while also allowing us to set a common weighted anomaly threshold.
The rationale behind the weighing is to adjust the sensitivity of the anomaly detection to the scale of each financial institution's operations in the context of the country where it operates. This ensures that the anomalies we end up with are both statistically significant and practically relevant.
Figure 6 shows the outliers per country for reference period February 2024, after applying the weighted threshold. Additionally, the final number of outliers submitted to the data compilers will be reduced, as we are developing a method to exclude outliers associated with mergers and acquisitions by leveraging information present in RIAD.

Operationalizing VAEs in a production pipeline

The goal when evaluating outlier detection algorithms for the IBSI dataset is ultimately to improve the quality of this data. To achieve this, VAE model will need to be operationalized and its outputs made available to the NCB compilers. In the following sections we describe how such operationalization of data and ML pipeline could be implemented in the context of a statistical production system.

Pipeline development

In the transition from experimental models to a production-ready implementation all the different phases of a machine learning model lifecycle must be taken into account. This encompasses model development, deployment, and monitoring. One of the promising open-source platforms that can be employed for this purpose is MLflow, which offers a set of features designed to streamline the machine learning workflow. It addresses the challenges of experiment tracking, model management, project packaging, and model serving, which are essential for maintaining the integrity and efficiency of ML systems in production environments.
MLflow's integration with data management systems, such as Apache Iceberg, is of particular interest. Apache Iceberg is a high-performance table format for large analytic datasets, designed for high scalability and reliability. It provides a unique capability called "time travel," which allows for querying historical versions of the data, enabling users to see data as it was at any point in time. This is particularly valuable for machine learning pipelines, as it ensures that the exact dataset versions used for model training can be reproducible and auditable.
Therefore, integrating MLflow and Apache Iceberg allows for facilitating a reproducible and auditable pipeline from data ingestion through to model inference, maintaining a comprehensive version history for both datasets and ML models. Already during the VAE model development and training phase, we used the MLflow tracking feature to log and compare the experimental runs, capturing all the essential information, including the model architecture and hyperparameters, as well as performance metrics. This logging helped us in the identification of the best-performing model, which was then registered within MLflow's Model Registry, where models can be version controlled and which can be used to deploy the models to different environments.
Once the inference results have been processed by the deployed model, they can be stored back into Iceberg tables or another database solution. As new IBSI data is

submitted and assessed for outliers, the performance of the VAE model should be monitored to detect any indications of performance degradation or data drift and assess when a retraining workflow should be initiated, to ensure the accuracy and reliability in outlier detection. Figure 7 shows a diagram of the pipeline in production.

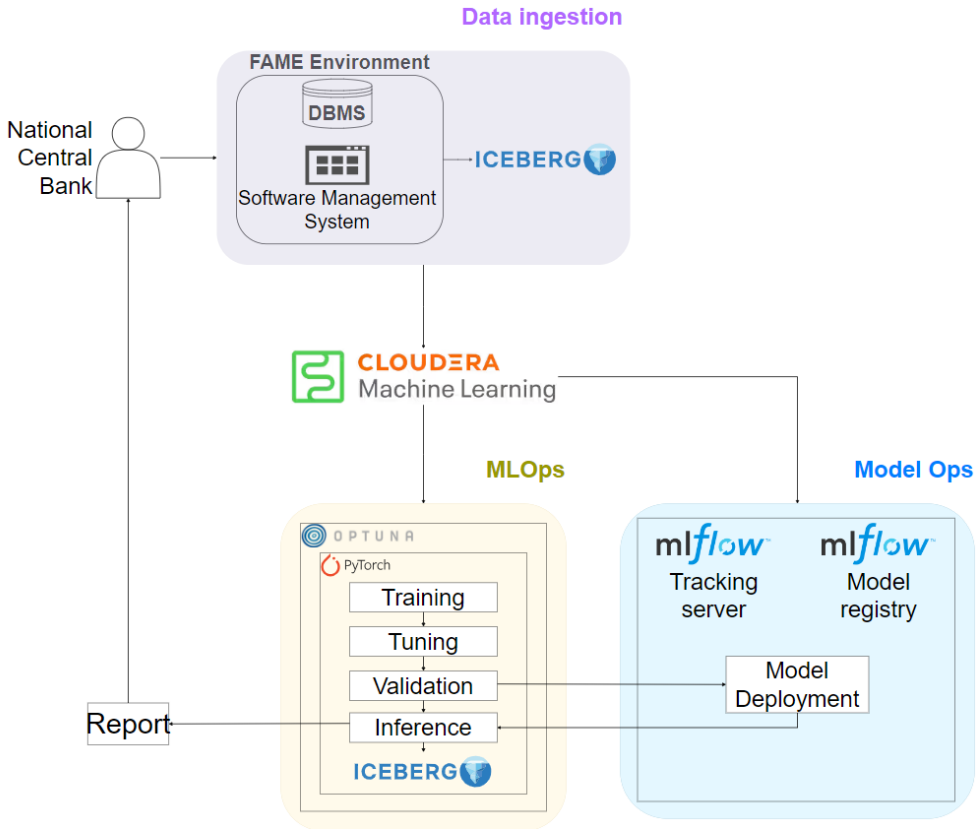


Figure 7: VAE pipeline in production

Conclusion

In this paper we demonstrated the significant potential of Variational Autoencoders (VAEs) in the realm of anomaly detection for multivariate time series data, specifically applied to the Individual Balance Sheet Items (IBSI) dataset of euro area banks. By leveraging the probabilistic framework of VAEs, we were able to effectively capture complex temporal dependencies and high-dimensional patterns, showcasing a robust and efficient alternative to traditional ARIMA-based methods. Adjustable thresholds based on volume-weighted reconstruction error may be employed to prioritise a manageable number of flagged outliers, thereby facilitating a streamlined quality assurance process with a focus on eliminating false positives.

Future work will focus on the implementation of the envisioned data and machine learning pipeline with potential methodological enhancements to the model. This includes exploring variations of VAEs, such as Conditional VAEs (CVAEs) and Recurrent VAEs (RVAEs) and expanding the scope to include other monetary statistics datasets, such as interest rate statistics.

The findings underscore the practical applicability of VAEs in monitoring data quality, as well as some advantages over current approaches which rely on ARIMA models.

Integrating VAEs into production pipelines at the ECB and national central banks would, with experience, lead to refinements in the anomaly detection process and more accurate and meaningful results.

Bibliography

1. Aggarwal, C. C. (2013). Outlier Analysis. Springer.
2. Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. Journal of Network and Computer Applications, 60, 19-31.
3. Aleskerov, E., Freisleben, B., & Rao, B. (1997). CARDWATCH: A neural network based database mining system for credit card fraud detection. Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr), 220-226.
4. An, J., & Cho, S. (2015). Variational Autoencoder based Anomaly Detection using Reconstruction Probability. Special Lecture on IE, 2(1), 1-18.
5. Basseville, M., & Nikiforov, I. V. (1993). Detection of Abrupt Changes: Theory and Application. Prentice-Hall.
6. Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). Time Series Analysis: Forecasting and Control. John Wiley & Sons
7. Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys (CSUR), 41(3), 15.
8. Ding, X., Zhang, Y., Liu, T., & Duan, J. (2014). Deep Learning for Event-Driven Stock Prediction. International Joint Conference on Artificial Intelligence (IJCAI), 2327-2333.
9. Doersch, C. (2016). Tutorial on Variational Autoencoders. arXiv preprint arXiv:1606.05908.
10. Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: principles and practice. OTexts.
11. Kingma, D. P., & Welling, M. (2013). Auto-Encoding Variational Bayes. arXiv:1312.6114.
12. 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 (Version 1). arXiv:1901.04997.
13. Li, G., & Jung, J. J. (2023). Deep learning for anomaly detection in multivariate time series: Approaches, applications, and challenges. In Information Fusion (Vol. 91, pp. 93–102). Elsevier BV.
14. Montgomery, D. C. (2009). Introduction to Statistical Quality Control. John Wiley & Sons.
15. Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. Proceedings of the 31st International Conference on Machine Learning (ICML).
16. Tang, S., Jin, H., Fang, C., Wang, Z., & de Sa, V. R. (2018). Speeding up context-based sentence representation learning with non-autoregressive convolutional decoding. arXiv:1710.10380

Annex

Hyperparameter tuning results

A comprehensive overview of the final model architecture after hyperparameter tuning with Optuna.

Encoder Layers	Type	in features	out features	Dropout
	Linear	1896	35	
	ReLU			
	Dropout			0.1950
	Linear	35	29	
	ReLU			
	Dropout			0.0157
	Linear	29	30	
	ReLU			
	Dropout			0.0603
	Linear	32	27	
	ReLU			
	Dropout			0.1073
	Linear	27	32	
	ReLU			
	Dropout			0.0840
	Linear	32	48	
	ReLU			
	Dropout			0.0058
	Linear	48	22	
	ReLU			
	Dropout			0.0070
	Linear	22	17	
	ReLU			
	Dropout			0.1164
	Linear	17	42	
	ReLU			
	Dropout			0.0778
	Linear	42	19	
	ReLU			
	Dropout			0.1040
	ReLU			
Latent Space Layers	**Type**	**in features**	**out features**	**Dropout**
	mu	19	19	
	logvar	19	19	

Decoder Layers	Type	in features	out features	Dropout
	Linear	19	30	
	ReLU			
	Dropout			0.0414
	Linear	30	37	
	ReLU			
	Dropout			0.1648
	Linear	37	42	
	ReLU			
	Dropout			0.1771
	Linear	42	30	
	ReLU			
	Dropout			0.1544
	Linear	30	48	
	ReLU			
	Dropout			0.0682
	Linear	48	35	
	ReLU			
	Dropout			0.0937
	Linear	35	1896	

IBSI Data Structure Definition (DSD)

# of dimension	Concept Description	Concept
1	Frequency	FREQ
2	Reference area	REF_AREA
3	Adjustment indicator	ADJUSTMENT
4	BS reference sector breakdown	BS_REP_SECTOR
5	Individual MFI list	MFI_LIST_IND
6	Balance sheet item	BS_ITEM
7	Original maturity	MATURITY_ORIG
8	Data type	DATA_TYPE
9	Counterpart area	COUNT_AREA
10	BS counterpart sector	BS_COUNT_SECTOR
11	Currency of transaction	CURRENCY_TRANS
12	Balance sheet suffix	BS_SUFFIX

EUROPEAN CENTRAL BANK

EUROSYSTEM

Variational Autoencoders for Multivariate Time-Series Outlier Detection

Tamara Fajt Mayer

Giovanni Raimondo Quaratino

Juan Francisco Javier Cordero Romero

Overview

Context

The Eurosystem – i.e. the ECB and national central banks of the euro area – uses bank-level statistical information to deepen analysis of monetary and economic developments

Objective

Explore the feasibility of using deep generative learning techniques for outlier detection on bank balance sheet data

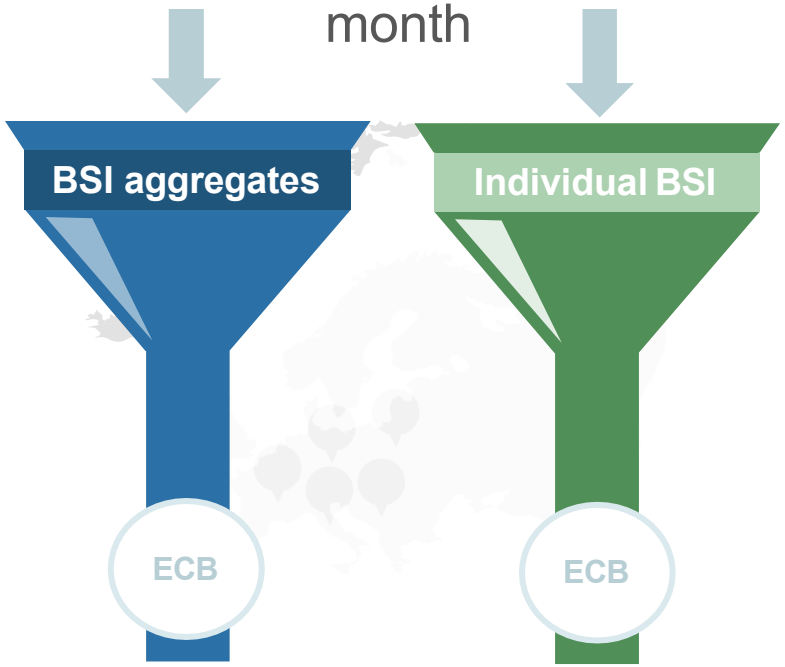
Next steps

The results indicate very robust and efficient detection of outliers, but these techniques would need to be integrated into production pipelines

The IBSI Dataset

Context

National central banks collect balance sheet items (**BSI**) data from euro area banks each month



Published BSI statistics, monetary aggregates, ...

~160 indicators transmitted for **~2850 individual banks**

Individual BSI transmission to the ECB began in **2012** to provide policy-makers and network of users with vital bank-level data

- July 2007 — Start of back data (initially for ~250 banks)
- June 2020 — Most recent significant expansion in coverage
- Jan. 2024 — Cut-off date for model training

IBSI indicators cover the main needs for the analysis of **monetary aggregates** and **credit**

Assets	Liabilities
Cash	Deposits
Loans	Capital and reserves
Debt securities held	Debt securities issued
External assets	External liabilities

- End-month **outstanding amounts**
- For key series, also adjustments to allow derivation of **transactions**

Data Preprocessing (1)

Objective

The IBSI dataset includes:

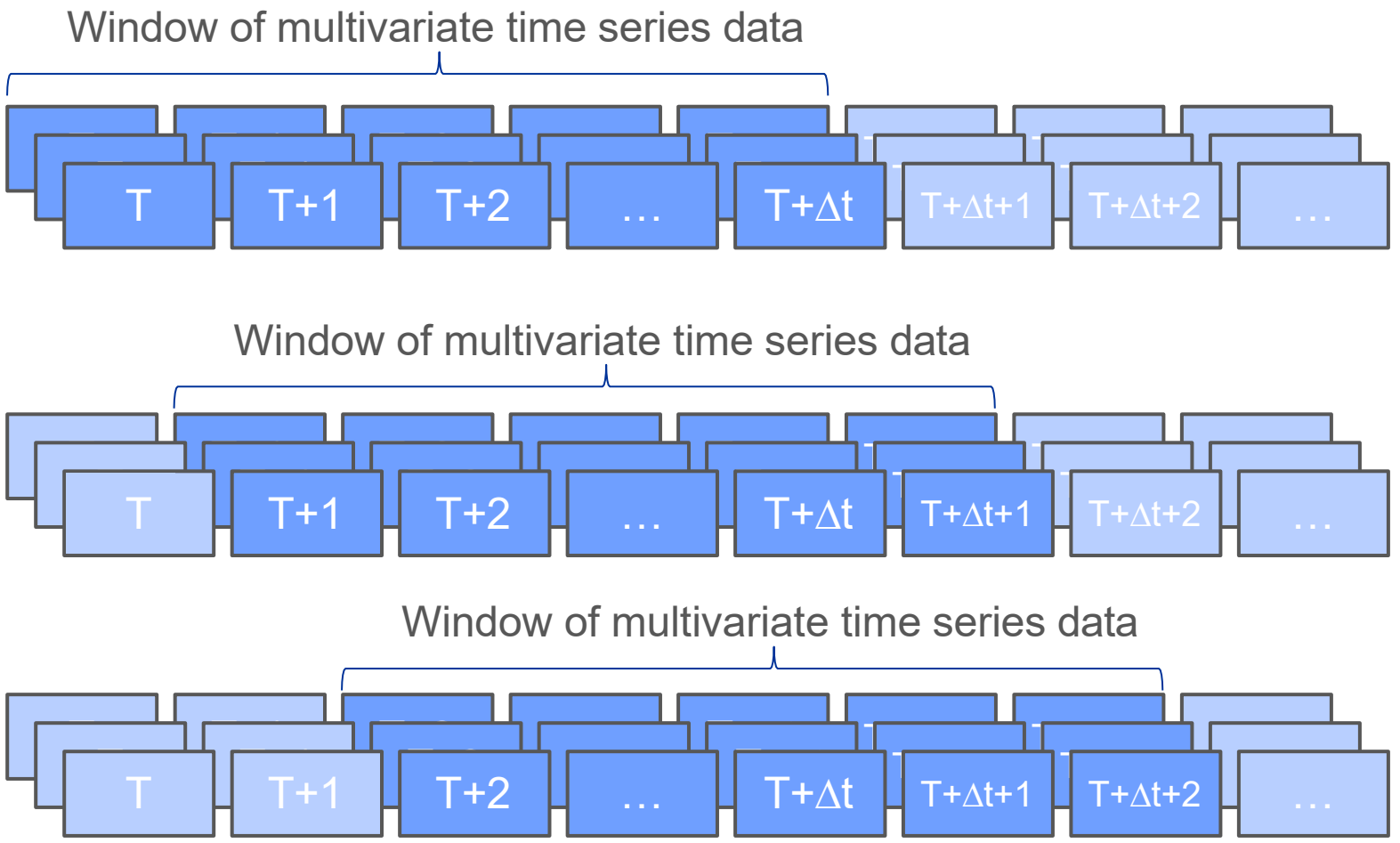
- Bank-specific characteristics
- Temporal dependencies

Data reshaping into training sequences:

- By window size (12) in steps of 1 (stacked)
- By MFI (158 series each)

Ensuring temporal consistency:

- Pad zeros to sequences that are too short
- Apply Z-score normalization



Data Preprocessing (2)

Objective

SDMX categorical variables from IBSI series keys were incorporated into the training data to improve performance

Accomplished with **Multi Hot Encoding** algorithm:

- Works by converting all unique values in a categorical column to **binary digits**, then encoding each value into its binary format, creating new columns
- **Advantages:** saves computational resources and avoids the high dimensionality and sparsity of one-hot encoding

Dataset (untransformed)

Index	Balance Sheet Item	…
0	A2C	…
1	A20	…
2	L22	
3	L40	
4	A51	
5	L21	
6	L7C	
7	A10	
8	L2A	
9	L7A	

Dataset (Multi Hot Encoded)

Index	Balance Sheet Item	Balance Sheet Item_0	Balance Sheet Item_1	Balance Sheet Item_2	Balance Sheet Item_3	…
0	A2C	0	0	0	0	…
1	A20	0	0	0	1	…
2	L22	0	0	1	0	…
3	L40	0	0	1	1	…
4	A51	0	1	0	0	…
5	L21	0	1	0	1	…
6	L7C	0	1	1	0	…
7	A10	0	1	1	1	…
8	L2A	1	0	0	0	…
9	L7A	1	0	0	1	…

The VAE Model

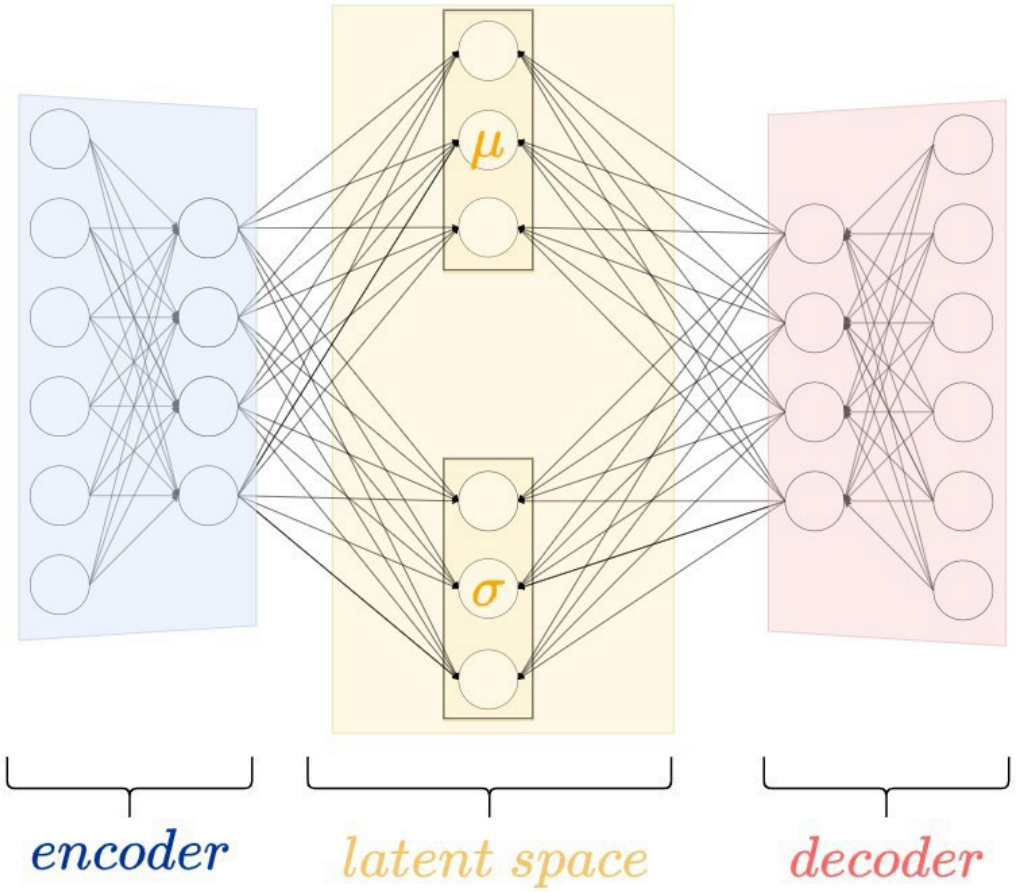
Objective

Variational Autoencoders (VAEs)

- **Probabilistic deep generative learning models** for anomaly detection
- If trained on sufficient data, the model can **accurately represent the IBSI dataset distribution**
- VAEs will learn to reconstruct the training data well but **fail to reconstruct anomalies** effectively

VAE training optimizes the **Evidence Lower Bound (ELBO)**, composed of two terms:

- **Reconstruction Term:** Measures how well the model reconstructs the observed data (**MSE**)
- **Regularization Term (KL divergence)**: Ensures the variational distribution stays close to the prior distribution, preventing overfitting



The Output

Objective

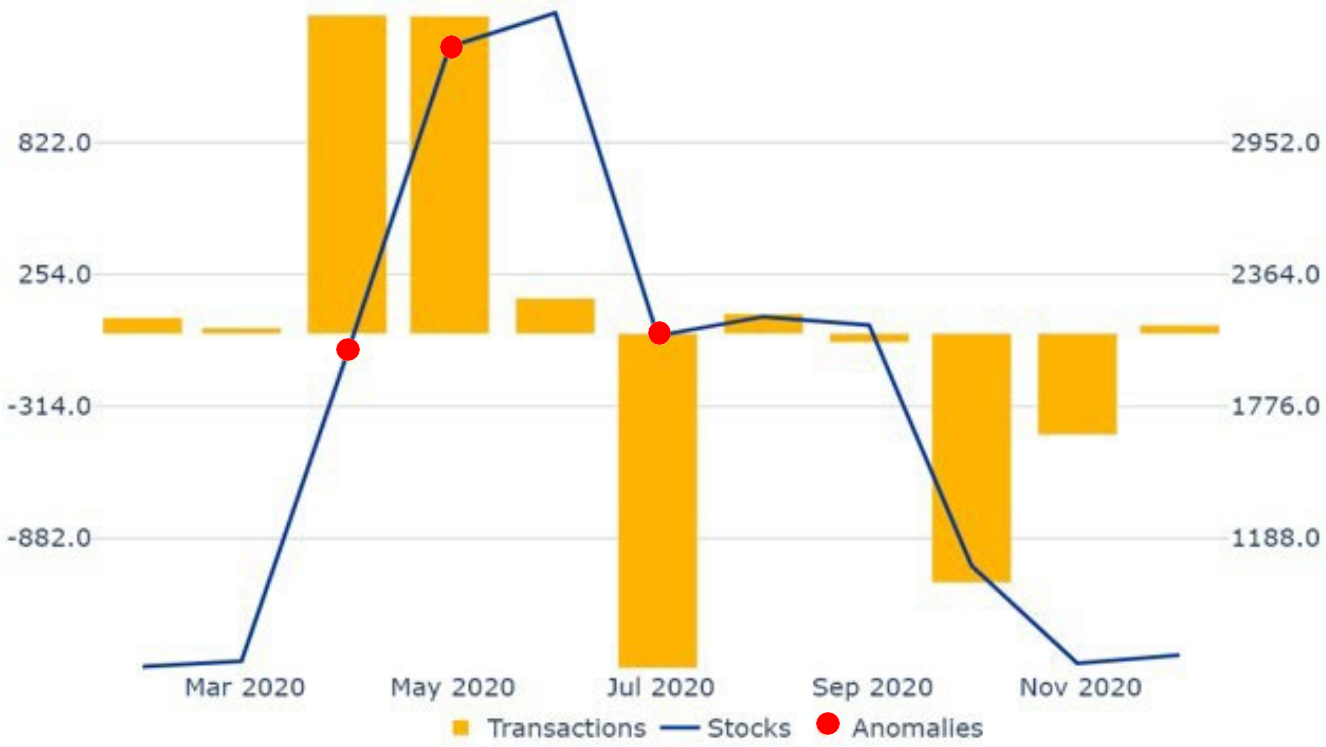
Reconstruction error

- The difference between input and output
- Serves as the **indicator of anomaly**

Normalization of reconstruction errors

- Assessing the significance of flagged anomalies relative to their volume enables compilers to focus efforts on most meaningful outliers
- For example, reconstruction errors (E) can be weighted by a normalized share of volume (α) to provide a **balanced number of outliers across countries**:

$$E_{weighted} = E(1 + 100\alpha^2)$$



MFI deposits - stocks (rhs) and flows (lhs) - with anomalies flagged by VAE

Benchmarks and Performance

Objective

Aim of the benchmarks:

- Compare the performance of the currently used ARIMA model and VAE on synthetic outliers
- Measured for **one reference period**, containing **529,932 data points**

Key findings:

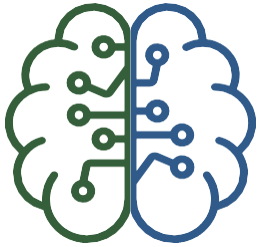
- Both models performed comparably in detecting non-outlier points
- VAE **significantly outperformed** ARIMA in detecting true outlier points
- VAE demonstrated **superior time efficiency and performance metrics**
- ARIMA model struggled with the dataset's scale

	ARIMA			VAE		
Label	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
False	1.00	0.99	0.99	1.00	1.00	1.00
True	0.15	0.57	0.24	**0.95**	**0.89**	**0.92**
Duration	16 hours 38 minutes			**10 minutes**		

Progress & Challenges

Next steps

Progress



Effective Anomaly Detection

- VAEs perform significantly better than ARIMA in detecting true outliers
- VAEs has much faster performance

Practical evaluation

- Calibrated & volume weighted thresholds minimize false positives and capture the most critical outliers

Coming up next

Production Integration

- Operationalizing VAE into the statistical production pipeline
- Model deployment and monitoring

Incorporating compiler feedback

- Investigate how to add accurate labels to the model to improve performance

Thank you!