technische universität
dortmund

fakultät für
informatik

Bachelor's Thesis

Performance Evaluation and Comparative Study of Variational Autoencoder Variants in Anomaly Detection

Apeksha Poudel

November 12, 2025

Supervisors:

Prof. Dr. Emmanuel Müller

Dr. Simon Klüttermann

Department of Computer Science
Lehrstuhl 9 - Data Science and Data Engineering
Technische Universität Dortmund
https://ls9-www.cs.tu-dortmund.de/

Contents

Acknowledgements **d**

1 Introduction **1**

1.1 Motivation and Research Question 1

1.2 Structure of Thesis 2

2 Theoretical Foundations **4**

2.1 Anomaly Detection 4

2.2 Autoencoders (AEs) and Variational Autoencoders (VAEs) 5

2.2.1 Autoencoders (AEs) 5

2.2.2 Variational Autoencoder (VAEs) 6

3 Variational Autoencoder Variants **9**

3.1 Normal VAE 9

3.2 Overcomplete VAE 10

3.3 2-HVAE and 3-HVAE 10

3.4 Beta VAE 11

3.5 Beta Total Correlation VAE 12

3.6 Sparse VAE 13

3.7 Conditional VAE 13

3.8 Factor VAE 14

4 Related Work **15**

5 Methodology **17**

5.1 Datasets 17

5.2 Evaluation Metrics 17

5.2.1 ROC AUC and PR AUC 17

5.2.2 Training time and Inference time 18
5.3 Benchmarking Framework 19
5.3.1 Implemented Models 19
5.3.2 Experimental Setup 20
5.4 Baseline Algorithms 20
5.5 Hyperparameter Optimization 21
5.5.1 Motivation and Challenges 21
5.5.2 Optimization Setup 22
5.6 Dataset Characteristics Analysis 24

6 Experiments and Results **25**
6.1 Performance with Default Parameters 25
6.2 Effect of Hyperparameter Optimization 29
6.2.1 Overall Results 29
6.2.2 Optimization History 32
6.2.3 Hyperparameter Analysis 34
6.3 Comparison with Baselines 37
6.4 Influence of Dataset Characteristics 39
6.5 Summary of Findings 43

7 Conclusion and Future Work **45**

List of Figures **48**

List of Tables **49**

References **50**

Acknowledgements

The authors gratefully acknowledge the computing time provided on the Linux HPC cluster at Technical University Dortmund (LiDO3), partially funded in the course of the Large-Scale Equipment Initiative by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) as project 271512359.

1 Introduction

Anomaly detection (AD) is a fundamental task in data analysis, aiming to identify rare or unusual patterns that deviate from expected behavior [1]. Such anomalies often correspond to important real-world events, including fraudulent transactions, medical diagnoses, or cybersecurity threats [2], where timely detection is essential. As data continues to grow in scale, dimensionality, and complexity, detecting anomalies has become increasingly challenging, especially in high-dimensional or non-linear feature spaces where traditional methods often fail to generalize.

Recent progress in deep learning has introduced possibilities for handling these challenges through representation learning [14]. Among these, Autoencoders (AEs) and, in particular, Variational Autoencoders (VAEs) have emerged as a powerful approach for unsupervised anomaly detection [19, 4]. By learning to reconstruct input data through a compressed latent representation, VAEs can model the underlying data distribution and distinguish normal from abnormal instances based on reconstruction errors or likelihood estimates [19]. Over time, numerous VAE variants have been proposed to address specific limitations of the original formulation. These include extensions that improve the structure of the latent space, encourage disentanglement, enhance reconstruction, or introduce hierarchical representations. Examples include $\beta - VAE$ [16], Factor VAE [18], Hierarchical VAEs [38, 42], and Conditional VAE [37], among others. However, despite this growing diversity, there is no systematic benchmark that fairly compares their effectiveness under consistent conditions. As a result, it remains unclear which variants perform best for anomaly detection and under what circumstances. This thesis addresses this gap by providing a comprehensive evaluation and comparative study of multiple VAE variants to better understand their strengths, limitations, and suitability for different anomaly detection scenarios.

1.1 Motivation and Research Question

Over the past decade, research on Variational Autoencoders has produced a growing number of variants, each claiming advantages in reconstruction accuracy, interpretability,

or generalization. Yet, the systematic performance comparisons of these variants on large, diverse AD benchmarks remain limited. This absence of a unified evaluation framework has made it difficult to compare their actual effectiveness or understand which VAE designs are best suited for different types of data. Existing studies often assess only a few models and are confined to small-scale or image-based datasets, limiting the generalizability of their conclusions.
Furthermore, hyperparameter optimization (HPO), which can significantly influence model outcomes, is often overlooked or inconsistently applied, further complicating comparisons. This thesis is therefore motivated by the need for a fair, large-scale, and reproducible benchmark that can evaluate multiple VAE variants under consistent experimental conditions while accounting for the impact of HPO.
This work investigates the following three Research Questions (RQs):

- **RQ1**: Which VAE variants achieve the best performance for anomaly detection across diverse datasets?
- **RQ2**: How do hyperparameters affect the performance and stability of different VAE variants?
- **RQ3**: How do dataset characteristics influence the generalizability of VAE variants in anomaly detection tasks?

This thesis addresses these research questions through a comprehensive experimental framework that systematically evaluates multiple VAE variants across diverse datasets. By analyzing their performance, efficiency, and sensitivity to hyperparameters, the study provides deeper insights into their strengths, limitations, and practical applicability in anomaly detection.

1.2 Structure of Thesis

Chapter 2 introduces the theoretical foundations of this work, providing an overview of anomaly detection and the fundamental principles of Autoencoders and Variational Autoencoders. Chapter 3 presents the VAE variants examined in this study, outlining their architectural differences. Chapter 4 reviews related research on VAE-based anomaly detection and highlights the existing gaps that this thesis aims to address. Chapter 5 details the methodological framework, including datasets, evaluation metrics, benchmarking setup, and hyperparameter optimization process. Chapter 6 presents and analyzes the experimental results, discussing the comparative performances of the models and the factors

influencing their behavior. Finally, Chapter 7 concludes the thesis by summarizing the main findings and suggesting directions for future research.

2 Theoretical Foundations

This chapter outlines the essential theoretical and technical foundations on which this work is based and that are relevant for its understanding.

2.1 Anomaly Detection

Anomalies, also known as outliers, are data or event instances that differ significantly from the general patterns or distribution of a dataset. Any deviation from the usual patterns, such as terrorist events, fraud events, network intrusion events, or pandemic situations, is referred to as an anomaly [41]. An anomalous event is also referred to as an exceptional, abnormal, or unusual event [41]. To spot these rare or unexpected cases, we use a process called anomaly detection. Anomaly Detection (AD) is the process by which an advanced algorithm identifies certain data or data patterns to be anomalous [1]. Anomaly detection plays a vital role in various real-world applications mentioned above, such as fraud detection, cybersecurity, and health monitoring, where detecting unusual behavior early can prevent significant loss or damage.

Anomalies can be categorized into three different types [7] based on their characteristics. **Point anomalies** refer to the individual data points that significantly differ from the rest of the data, such as a single fraudulent financial transaction. **Contextual anomalies** occur when data points are only anomalous in a particular context, such as an unusually high temperature in winter, which would be normal in summer. Another type of anomaly is **Collective anomalies**. These are formed by a group of related data points that may appear normal individually but are anomalous when considered together (e.g., a sequence of unusual network requests indicating intrusion). Recognizing these categories is essential for selecting suitable detection approaches.

Several approaches to anomaly detection have been proposed in the literature. These can be broadly divided into **shallow methods** and **deep learning-based methods**. Shallow methods include statistical techniques that model data distributions, as well as other algorithms that are simple to implement, require fewer computational resources, and can perform surprisingly well. However, these approaches often struggle when the data is

high-dimensional, non-linear, or has complex patterns. Deep learning-based methods are designed to address such cases by capturing non-linear, high-dimensional, and complex patterns in the data. Among these, autoencoders and their probabilistic extensions, such as Variational Autoencoders (VAEs) [19], have become increasingly popular. An overview of these methods is provided in Section 2.2.

2.2 Autoencoders (AEs) and Variational Autoencoders (VAEs)

Autoencoders and their probabilistic extensions have been widely applied in anomaly detection tasks. This section provides a brief introduction to these models and explains their role in detecting anomalies.

2.2.1 Autoencoders (AEs)

Autoencoders are unsupervised neural networks designed to learn a compressed representation of input data and reconstruct it with as much accuracy as possible. Autoencoders have been widely used for dimensionality reduction [22] and more recently for anomaly detection tasks. It consists of two main components: an **encoder**, which maps the input **x** to a lower-dimensional **latent representation z**, and a **decoder**, which reconstructs the input $\hat{\mathbf{x}}$ from this representation. The latent space acts as a bottleneck that forces the model to capture the most essential features of the input data.
The primary objective of an autoencoder is to minimize the reconstruction error between the input and its reconstruction, which is typically measured using a loss function such as Mean Squared Error (MSE) for continuous-valued data or Binary Cross-Entropy (BCE) for binary or categorical data [23]. Formally, these loss functions are defined as follows:
Loss function using MSE:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{M}\sum_{i=1}^{M}\|x - \hat{x}\|^2$$

Loss function using BCE:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{M}\sum_{i=1}^{M}[x\log(\hat{x}) + (1-x)\log(1-\hat{x})]$$

where, M is the number of observations in the training dataset, x is the input, and $\hat{x}$ is the reconstructed input.

Autoencoders can be used for a variety of tasks, such as denoising, image super-resolution, anomaly detection, and clustering [32]. They are particularly useful for anomaly detection because they are usually trained only on normal data. This means they learn how normal data should look. When we give them something unusual, they likely cannot reconstruct it well, leading to a higher reconstruction error. This reconstruction error can therefore serve as an effective anomaly score for identifying unusual instances.

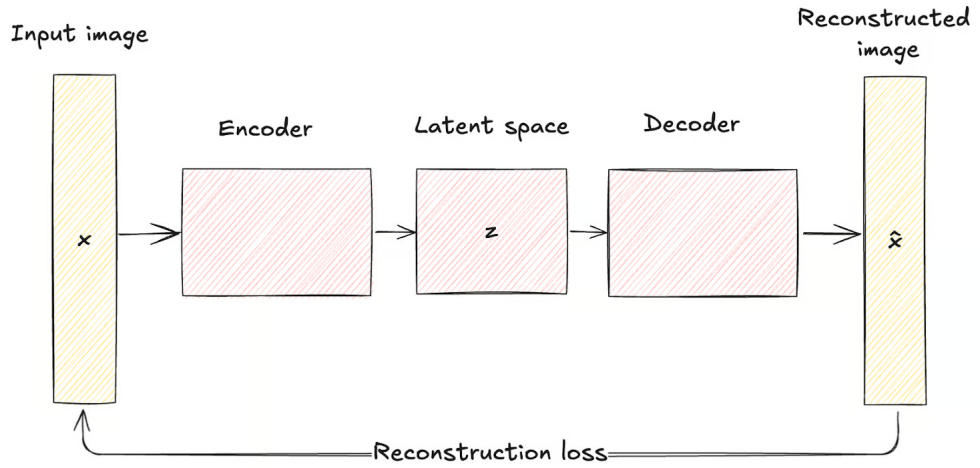
However, basic autoencoders have certain limitations. They may overfit to the training data, which can lead to accurate reconstructions of anomalous inputs as well. In addition, they lack a probabilistic interpretation of the latent space, making it difficult to model variability or generate new samples in a principled way. These limitations motivate the use of probabilistic extensions such as Variational Autoencoders (VAEs), which are discussed in the following section.

2.2.2 Variational Autoencoder (VAEs)

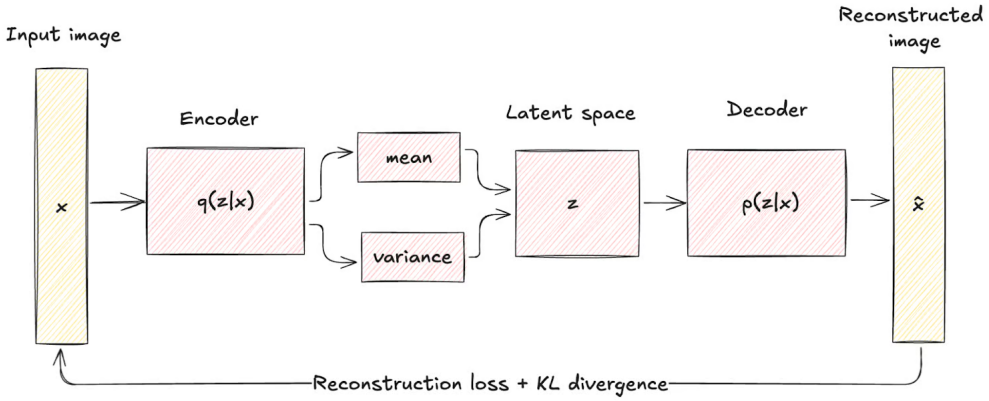
As discussed in the previous section, basic autoencoders have several limitations, including the lack of a structured, probabilistic latent space. This means similar inputs do not necessarily map to similar latent points, making it difficult to generate new data or interpret the patterns captured in the latent space. VAEs address these limitations by introducing a probabilistic latent space, which encourages structure and enables meaningful data generation.

VAEs are built on the idea of regular autoencoders but take a probabilistic approach [19]. Instead of mapping an input to a single point, the encoder outputs a distribution, typically a Gaussian distribution defined by mean (μ) and variance (σ^2) for each latent dimension. The mean determines the center of the distribution for that dimension, while the variance controls its spread. From this distribution, a latent vector $\mathbf{z}$ is sampled and passed through the decoder to reconstruct the input.

The architectures of both models are shown in Figure 2.1. In a standard autoencoder (Figure 2.1(a)), the encoder maps the input to a fixed point in a latent space, and the decoder reconstructs the input from this point. In contrast, a VAE (Figure 2.1(b)) produces the parameters of a probability distribution for each latent dimension, from which the latent vector is sampled before reconstruction. The VAE loss combines a reconstruction term, which measures the quality of reconstruction, with a KL divergence term, which regularizes the latent space to follow a chosen prior distribution.



(a) Autoencoder: maps inputs to a fixed latent point.



(b) VAE: models latent space as a distribution.

Figure 2.1: Comparison between AE and VAE architectures. Image adapted from [29]

Training a VAE involves two steps:

- **Reconstruction loss:** measures how well the decoder can reconstruct the input from the sampled latent vector.
- **Kullback-Leibler (KL) Divergence:** measures how close the learned latent distribution $q(z|x)$ is to a prior distribution $p(z)$.

Mathematically, the total loss function looks like this:

$$\mathcal{L}(x) = \underbrace{\mathbb{E}_{q(z|x)}\left[\log p(x|z)\right]}_{\text{Reconstruction Loss}} - \underbrace{D_{\text{KL}}\left(q(z|x) \,\|\, p(z)\right)}_{\text{KL Divergence}}$$

The optimized term L in the above equation is called the ELBO (Expectation Lower Bound) [32]. The first term ensures good reconstruction, and the second keeps the latent

space well-behaved.

This design allows VAEs to learn complex data distributions and model uncertainty in a way that traditional autoencoders cannot. For anomaly detection, this design is particularly useful; when an input does not fit the learned distribution, it often results in either a **high reconstruction error** or **low sample likelihood** [4]. These properties make VAEs a strong foundation for anomaly detection tasks, and their flexibility has inspired the development of numerous VAE variants. These variants build upon the basics of the VAE framework, introducing improvements to the latent space structure, reconstruction quality, or disentanglement of features. These aspects will be explored in the upcoming sections.

3 Variational Autoencoder Variants

Building on the foundation of the standard Variational Autoencoder, researchers have proposed several variants aimed at addressing specific limitations, such as improving reconstruction quality, refining latent space structure, or enhancing feature disentanglement. In this thesis, we focus on a set of representative VAE variants that are widely discussed in the literature and relevant to anomaly detection.

3.1 Normal VAE

The *Normal VAE* refers to the standard Variational Autoencoder [19], which serves as the baseline model in this study and was introduced in detail in Section 2.2.2. A VAE consists of two neural networks: an encoder that maps the input data x into a latent distribution parametrized by mean and variance, and a decoder that reconstructs the input from the sampled latent vector z.
The Normal VAE is trained by minimizing a loss function that combines the reconstruction error and KL divergence:

$$\mathcal{L}(x) = \mathbb{E}_{q(z|x)} \left[\log p(x|z)\right] - D_{\mathrm{KL}} \left(q(z|x) \,\|\, p(z)\right)$$

Here, the first term encourages the model to reconstruct the input as closely as possible, while the KL divergence pushes the approximate posterior $q(z|x)$ closer to the prior $p(z)$, usually a standard normal distribution.
For anomaly detection, the Normal VAE is trained on normal samples to capture their distribution. Inputs that deviate from this learned distribution typically produce higher reconstruction errors or lower likelihoods, which can be used as anomaly scores. In our implementation, we used the PyOD framework [45], with the default latent dimension set to 2, meaning the encoder compresses each input into a two-dimensional latent representation.
All other VAE variants examined in this thesis build upon the core framework of the Normal VAE, either by changing the architecture or by modifying the training objective

to address specific limitations. This model, therefore, serves as the baseline for evaluating the impact of such modifications.

3.2 Overcomplete VAE

The *Overcomplete Variational Autoencoder* builds directly upon the architecture of the Normal VAE, with the key difference being the size of the latent space. In most standard autoencoder and VAE settings, the latent dimension is chosen to be smaller than the input dimension. This bottleneck forces the model to compress information, capture the most relevant features, and avoid trivial identity mappings [44].

The Overcomplete VAE, in contrast, deliberately sets the latent dimension to be larger than the input dimension. In our implementation, the latent dimension is chosen as twice the number of features, resulting in a more expressive latent representation. This design choice was inspired by findings from [21], where an overcomplete latent space was shown to significantly improve anomaly detection performance.

By allowing the model to encode more detailed representations of the input data, the Overcomplete VAE can potentially improve reconstruction quality and anomaly detection accuracy. However, the larger latent space also increases the computational cost and may raise the risk of overfitting.

3.3 2-HVAE and 3-HVAE

The *Hierarchical Variational Autoencoder (HVAE)* extends the standard VAE [19] by introducing a layered structure of latent variables, enabling the model to capture more complex and multiscale variations in the data [42, 38]. Our implementation follows the Ladder VAE framework [38] and related designs such as NVAE [42], where lower-latent variables depend on both input and higher-level latent variables.

Figure 3.1 illustrates the structure of a two-layer HVAE. The encoder performs a bottom-up inference, where the first latent layer $\mathbf{z_1}$ is derived from the input x using q($\mathbf{z_1}$|x), and the second latent layer $\mathbf{z_2}$ is then inferred based on ($\mathbf{z_1}$ through q(($\mathbf{z_2}$|$\mathbf{z_1}$). In contrast, the decoder follows a top-down generative process, first generating $\mathbf{z_1}$ from $\mathbf{z_2}$ via p($\mathbf{z_1}$|$\mathbf{z_2}$) and then reconstructing the input x through p(x|$\mathbf{z_1}$). This hierarchical relationship allows the model to capture both high-level abstract features and lower-level details across the latent space [42].

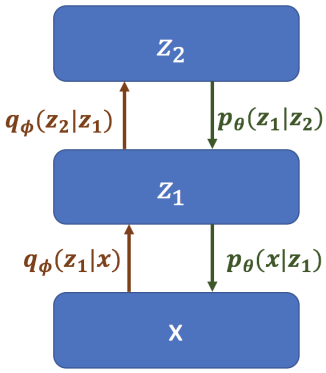


Figure 3.1: Illustration of two-layer Hierarchical Variational Autoencoder (HVAE). The encoder (left, brown arrows) infers latent variables $\mathbf{z_1}$ and $\mathbf{z_2}$ in a bottom-up manner, while the decoder (right, green arrows) reconstructs the data through a top-down generative process. Image Adapted from [39]

In the **2-HVAE** variant, the model uses two latent layers ($\mathbf{z_1}$ and $\mathbf{z_2}$), where $\mathbf{z_1}$ depends on both the input $\mathbf{x}$ and $\mathbf{z_2}$. In the **3-HVAE** variant, we introduce an additional latent layer $\mathbf{z_3}$, allowing the model to learn more abstract, high-level representations before refining them through the lower layers.

The training objective for a hierarchical VAE with latent layers $\mathbf{n}$ can be expressed as:

$$\mathcal{L}(x) = \mathbb{E}_{q(z_1,\ldots,z_n|x)}\left[\log p\left(x \mid z_1, \ldots, z_n\right)\right] - \sum_{i=1}^{n} D_{\mathrm{KL}}\left(q(z_i \mid x, z_{i+1}, \ldots, z_n) \,\|\, p(z_i)\right)$$

Compared to the baseline VAE, which has only a single KL divergence term for one latent layer, the hierarchical formulation includes a KL divergence term for each latent layer, allowing the model to regularize multiple levels of representation.

By modeling data hierarchically, both variants can represent patterns at different levels of abstraction, thereby improving anomaly detection performance compared to a single-layer VAE. However, the added depth increases computational cost and may require more careful training strategies (e.g., KL annealing) to avoid optimization issues. We implemented this model in PyTorch and utilized techniques such as KL annealing to aid in training. We then evaluated its performance using ROC AUC and PR AUC scores, as mentioned earlier.

3.4 Beta VAE

The *Beta VAE (β-VAE)* is a modification of the standard Variational Autoencoder introduced in 2017 [16]. It introduces an adjustable hyperparameter β that balances latent

channel capacity and independence constraints with reconstruction accuracy [16]. This means that the model can focus more on learning disentangled representations, where each dimension in the latent space captures a separate factor of variation in the data.
To achieve this, the original VAE loss function is modified as follows:

$$\mathcal{L}_{\beta\text{-VAE}} = \mathbb{E}_{q(z|x)}[\log p(x|z)] - \beta \cdot D_{\mathrm{KL}}(q(z|x)\|p(z))$$

Compared to the baseline VAE loss function, the only change here is the multiplier β on the KL term, which shifts the model's focus toward a more structured latent space. While stronger regularization ($\beta > 1$) can improve generalization and separation of normal versus anomalous patterns, it may also reduce reconstruction quality. In our experiments, we systematically investigate the effect of the β hyperparameter to understand how this trade-off influences anomaly detection performance.

3.5 Beta Total Correlation VAE

The *Beta Total Correlation Variational Autoencoder* ($\beta - TC\,VAE$) is an extension of the standard $\beta - VAE$ framework, proposed by Chen et al. [8], to learn more disentangled representations. Unlike $\beta - VAE$, which applies a single penalty to the full KL divergence term, $\beta - TCVAE$ decomposes the KL divergence into three separate components [8]:

- **Mutual Information (MI)** - measures how much information the latent variables $\mathbf{z_i}$ contain about the input $\mathbf{x}$; controlling this term helps balance reconstruction accuracy and latent compression.
- **Total Correlation (TC)** - measures the statistical dependence between latent dimensions; penalizing TC encourages the model to learn independent latent factors rather than entangled ones.
- **Dimension-wise KL (DW-KL)** - ensures each z_i stays close to the prior $p(z_i)$, maintaining overall regularization and preventing overfitting.

The Loss function is modified as :

$$\mathcal{L}_{\beta\text{-TCVAE}} = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})] - \alpha \cdot I_q(\mathbf{x};\mathbf{z}) - \beta \cdot \mathrm{KL}\left(q(\mathbf{z}) \parallel \prod_j q(z_j)\right) - \gamma \cdot \sum_i D_{\mathrm{KL}}\left(q(z_i) \parallel p(z_i)\right)$$

This can be summarized as:

$$\mathcal{L}_{\beta\text{-TCVAE}} = \text{Reconstruction Loss} + \alpha \cdot \text{MI} + \beta \cdot \text{TC} + \gamma \cdot \text{DW-KL}$$

Here, α, β, and γ control the relative importance of each component. Instead of applying the β penalty to the full KL term, $\beta - TC$VAE applies it only to the Total Correlation component, allowing the model to learn independent latent dimensions without overly penalizing mutual information or regularization.

3.6 Sparse VAE

The *Sparse Variational Autoencoder (SVAE)* modifies the standard VAE to promote *sparse* latent representations, meaning that only a few latent dimensions are active for any given input [13]. This is achieved by adding an L1 penalty or using sparse priors such as a Laplace distribution on the latent variables, which pushes most latent activations toward zero. In practice, this means that only a small subset of latent dimensions carries meaningful information, while the rest remain inactive [13]. Such sparsity can improve interpretability and make the learned features more selective.
The SVAE loss function extends the standard VAE objective with a sparsity term:

$$\mathcal{L}_{\text{SVAE}}(x) = \mathbb{E}_{q(z|x)}\left[\log p(x|z)\right] - \beta \cdot D_{\text{KL}}\left(q(z|x) \,\|\, p(z)\right) - \lambda \cdot \mathbb{E}_{q(z|x)}\left[\|z\|_1\right]$$

Here, β controls the weight of the KL divergence term, while λ regulates the strength of the L1 penalty applied to the latent variables. By encouraging most latent dimensions to remain near zero, the SVAE produces compact and interpretable latent codes, which can be particularly useful in anomaly detection tasks where identifying the most relevant features is crucial.

3.7 Conditional VAE

The *Conditional Variational Autoencoder (CVAE)*, introduced by Sohn et al. [37], extends the standard VAE by incorporating an additional conditioning variable $\mathbf{y}$ into both encoder and decoder. This variable can represent class labels, attributes, domain info, or other relevant side information that helps guide the reconstruction process.
In a CVAE framework, the encoder learns the posterior distribution $\mathbf{q}(\mathbf{z}|\mathbf{x}, \mathbf{y})$ using both input x and the condition y, while the decoder uses z together with y to reconstruct x.
The loss function is:

$$\mathcal{L}_{\text{CVAE}}(x, y) = \mathbb{E}_{q(z|x,y)}\left[\log p(x|z, y)\right] - D_{\text{KL}}\left(q(z|x, y) \parallel p(z|y)\right)$$

Here, y influences both the prior $p(z|y)$ and the reconstruction process, enabling the model

to generate outputs consistent with the given condition.
For anomaly detection, conditioning can be useful when additional information, such as known categories, operational modes, or environmental settings, is available. However, the benchmark datasets used in this work do not include such additional characteristics. Therefore, the conditional vector was defined with a single dimension (y = 1) and initialized uniformly across all samples. This setup maintains architectural compatibility with the CVAE framework while allowing consistent comparison with other VAE variants under the same experimental conditions.

3.8 Factor VAE

The *Factor Variational Autoencoder (FVAE)*, proposed by Kim and Minh [18], extends the standard VAE by encouraging the latent variables to be statistically independent. This is achieved by adding a penalty term to the objective function that targets the Total Correlation (TC) of the latent vector $\mathbf{z}$. Total correlation measures how much the different dimensions of $\mathbf{z}$ depend on each other; lower values indicate more independent latent dimensions.
Formally, the total correlation is defined as the KL divergence between the joint distribution of the latent variables and the product of their marginals [18]:

$$\mathrm{TC}(z) = D_{\mathrm{KL}}\left(q(z) \,\middle\|\, \prod_{j=1}^{d} q(z_j)\right)$$

where q(z) is the joint distribution of all latent dimension and $\prod_j q(z_j)$ is the product of their marginals. A higher TC indicates stronger statistical dependence between latent dimensions.
The loss function of Factor VAE is given by:

$$\mathcal{L}_{\mathrm{FactorVAE}}(x) = \mathbb{E}_{q(z|x)}\left[\log p(x|z)\right] - D_{\mathrm{KL}}\left(q(z|x) \parallel p(z)\right) - \gamma \cdot \mathrm{TC}(z)$$

Here, γ controls the strength of the total correlation penalty. However, directly computing TC is difficult because the posterior q(z) is unknown. To approximate it, Factor VAE introduces an additional discriminator that distinguishes between true latent samples and samples where latent dimensions have been randomly permuted. This allows the model to minimize TC during training indirectly. By reducing redundancy in the latent space, FactorVAE encourages disentangled and interpretable representations, thereby improving its ability to separate normal and abnormal patterns in anomaly detection tasks.

4 Related Work

Variational Autoencoders (VAEs) have become a central framework for unsupervised anomaly detection due to their ability to model complex data distributions and identify rare deviations from normal patterns. Since their introduction by Kingma and Welling [19], numerous VAE variants have been proposed, each aiming to improve certain aspects such as reconstruction quality, disentanglement, or latent structure regularization. Although these developments have expanded the applicability of VAEs, comprehensive comparative evaluations across variants remain limited.

Most existing benchmarks and comparative studies focus primarily on image domains. Neloy and Turgeon [24] compared eleven autoencoder variants, including several VAEs on Fashion MNIST and MNIST, highlighting trade-offs between reconstruction accuracy and representation efficiency. Similarly, Nguyen et al. [25] compared a standard VAE, a Gaussian Random Field VAE (VAE-GRF), and a Vision Transformer VAE (ViT-VAE) on MVTEC and MiAD defect-detection datasets, showing that transformer-based VAEs achieved the best detection accuracy but required careful hyperparameter tuning. While these works demonstrate the potential of advanced VAE architectures, they remain limited to small, domain-specific image benchmarks, leaving open questions about their generalization to broader anomaly detection tasks.

To extend VAEs beyond image data, several studies have explored more robust and structured variants of this approach. Akarmi et al. [3] introduced a β-VAE-based approach for handling outliers in real-world tabular datasets. Pol and the team [28] proposed a Conditional VAE with feature-wise variance learning that improved detection performance on hierarchical CERN trigger data, while Huang and the team [5] developed a Hierarchical VAE (HVAE) with multi-scale conditioning to guide latent representations. Broader surveys, such as the *Meta-Survey on Outlier and Anomaly Detection* [27], emphasize challenges in consistent model evaluation. Likewise, Shmuel et al. [33] compared machine learning and deep learning methods on diverse tabular datasets and found that deep models often struggle to outperform classical algorithms without careful architecture design and parameter tuning. These findings highlight the need for systematic benchmarking on tabular data, where the heterogeneous features make representation learning particularly

challenging [34].

Tabular data remains dominant in real-world anomaly detection tasks across finance, cybersecurity, manufacturing, and healthcare. Improving model robustness and generalization on this data type is, therefore, key to making deep generative models practically useful. Recent large-scale benchmarks such as ADBench [15] have reinforced the importance of reproducible evaluation on tabular datasets, revealing that deep models often lag behind classical algorithms without proper optimization [34, 27]. Building on this foundation, the present work extends the benchmarking effort specifically to Variational Autoencoders and their variants.

Despite significant progress, several challenges remain. Prior work often evaluates models in isolation and under inconsistent experimental setups [17]. Moreover, hyperparameter optimization (HPO), a major factor influencing performance, is frequently overlooked or applied inconsistently, causing bias in reported results [11, 36]. Consequently, it remains unclear which VAE architectures are most effective for anomaly detection and under what conditions they generalize best.

This thesis addresses these gaps through a large-scale, systematic, and reproducible benchmark of nine VAE variants under unified experimental conditions. To ensure fairness, a two-phase cross-validation HPO strategy was introduced, allowing each model to be tuned and validated on separate sets of datasets. This approach not only accounts for the influence of hyperparameter tuning but also identifies a single, generalizable hyperparameter configuration per model that performs robustly across unseen datasets. By bridging the methodological inconsistencies of prior studies and extending evaluation to a diverse collection of 121 tabular datasets [15], this work provides new insights into the relative strengths, limitations, and practical applicability of VAE architectures for anomaly detection.

5 Methodology

This chapter describes the experimental setup used to evaluate the performance of different VAE variants. The goal is to ensure a fair and consistent evaluation across a large and diverse set of datasets. We first introduce the datasets and evaluation metrics, then describe the benchmarking framework, including model implementation and training procedures. Finally, we outline the baseline algorithms, the hyperparameter optimization strategy, and the analysis of dataset characteristics.

5.1 Datasets

For this study, we used 121 datasets from the ADBench benchmark, which provides a standardized and widely adopted collection for anomaly detection research [15]. ADBench ensures a consistent evaluation environment and avoids the bias that may arise from testing on a limited set of datasets. The datasets cover a wide range of domains, including healthcare, finance, network intrusion, and sensor monitoring, and show significant variation in sample sizes, the number of features, and anomaly ratios. The dimensionality ranges from low to high, providing a diverse and challenging benchmark for evaluating model performance under different data conditions. Before training, all datasets were preprocessed consistently. Features were normalized to the range [0, 1], and the provided training-test splits were used to maintain comparability across models.

5.2 Evaluation Metrics

To fairly evaluate VAE variants, this study uses two types of metrics: performance metrics (ROC AUC and PR AUC), which handle class imbalance, and efficiency metrics (Training and Inference time), which capture practical usability.

5.2.1 ROC AUC and PR AUC

To assess how well a model detects anomalies, especially in imbalanced datasets, we need more reliable metrics than simple accuracy. Accuracy can be misleading when anomalies

are rare, as a model may achieve very high accuracy by simply predicting the majority class and still miss nearly all anomalies. For this reason, more informative metrics are required. Two of the most commonly used evaluation metrics to evaluate the performance of algorithms in anomaly detection are ROC AUC and PR AUC.

The **ROC curve** is a two-dimensional depiction of classifier performance [12], that plots the True Positive Rate (TPR) and the False Positive Rate (FPR) across different threshold values, where:

$$TPR = \frac{TP}{TP + FN} \quad , \quad FPR = \frac{FP}{FP + TN}$$

Here, TP, FN, FP, and TN represent the number of true positives, false negatives, false positives, and true negatives, respectively. The ROC AUC is the area under the curve, ranging from 0.5 (random guessing) to 1.0 (perfect separation between normal and anomalous samples). Unlike accuracy, ROC AUC is not affected by class imbalance, since it measures the probability that a randomly chosen anomaly is ranked higher than a randomly chosen normal sample. This property makes it independent of the relative number of anomalies in the dataset [26].

The **Precision-Recall (PR)** curve can show performance differences between balanced and imbalanced datasets, and it can be useful in revealing the early-retrieval performance [9]. It plots Precision against Recall for different thresholds:

$$Precision = \frac{TP}{TP + FP} \quad , \quad Recall = \frac{TP}{TP + FN}$$

While recall measures how many actual anomalies were detected, precision tells us how many of the detected anomalies were correct. The PR AUC gives an aggregate measure of this trade-off. In contrast to ROC AUC, PR AUC focuses entirely on the positive class and is therefore more sensitive to the performance on rare anomalies [31].

5.2.2 Training time and Inference time

Performance metrics like ROC AUC and PR AUC help evaluate how well a model behaves, but they don't capture how efficient the model is to use in practice. In many real-world applications, both high accuracy and low computational cost are essential.

Training time is the total time required for a model to learn from the training data. This depends on the complexity of the model, the size of the dataset, and also on the availability of hardware resources. For deep learning models, such as Variational Autoencoders, training can be computationally intensive, particularly when working with high-dimensional data or large-scale architectures. In our study, comparing training time

helped us to understand the computational demands of each variant and their scalability.

On the other hand, **Inference time** is the time taken to make predictions after the model has been trained. In anomaly detection, low inference time is crucial for applications that require rapid responses, such as fraud detection or industrial monitoring.

In this study, we evaluate both training and inference times to assess the overall computational efficiency of each VAE variant. While training time provides insight into model scalability, inference time is especially critical for real-time anomaly detection scenarios where quick responses are essential.

5.3 Benchmarking Framework

We developed a dedicated benchmarking framework to ensure a fair and consistent comparison of the VAE variant. Building on the evaluation metrics described in Section 5.2, this framework outlines the implementation of models and the conduct of experiments in a reproducible manner. It consists of two main components: the set of implemented VAE variants and the experimental setup used for training and evaluation.

5.3.1 Implemented Models

A total of nine VAE variants were included in this study: Normal VAE, Overcomplete VAE, 2-Hierarchical VAE, 3-Hierarchical VAE, ß-VAE, Factor VAE, ß-TCVAE, Conditional VAE, and Sparse VAE. The implementations were adapted from an existing PyTorch repository [40], originally designed for image datasets. Since all datasets in this study are tabular, the models were modified to use fully connected encoder-decoder networks instead of convolutional layers. We kept the original loss functions and training objectives to stay true to the theoretical design of each model.

At the same time, the architectures were made flexible to handle the wide range of feature dimensions and data types found in tabular datasets. The overall structure and layer sizes were chosen to balance model capacity and stability, ensuring a fair comparison between variants. Batch normalization and dropout were applied where appropriate to improve training stability and reduce overfitting, particularly on smaller datasets. These adjustments kept each VAE variant true to its original design while making it practical and reliable for our large-scale benchmarking.

5.3.2 Experimental Setup

Each dataset came with a predefined split into training and test sets. For every dataset, the framework loads the training features($\mathbf{x}$), test features($\mathbf{tx}$), and test labels($\mathbf{ty}$) provided by ADBench. To standardize the input space, features were normalized to the range $[\mathbf{0}, \mathbf{1}]$ using a MinMax scaler fitted on the training data and applied to the test data, thereby preventing any form of data leakage. Models were trained on the training split and evaluated on the corresponding test split to maintain fairness across all experiments.
Optimization was carried out with the Adam optimizer, with learning rate, batch size, and the maximum number of epochs set according to the configuration of each model. For every run, the framework recorded ROC AUC and PR AUC as performance metrics, along with training and inference time to capture efficiency. All results were saved on a per-dataset basis, and plots were generated to highlight both within-group comparisons of the VAE variants and cross-group comparisons against baseline algorithms. This systematic setup ensured that the evaluation process remained consistent, reproducible, and unbiased across the wide range of datasets.

5.4 Baseline Algorithms

To put the performance of the VAE variants into perspective, four widely used baseline algorithms for anomaly detection were included: **k-Nearest Neighbors (kNN)**, **Isolation Forest**, **Autoencoder (AE)**, and **Principal Component Analysis (PCA)**. These methods are widely used in anomaly detection and represent a balance between traditional distance-based, tree-based, and reconstruction-based approaches.
The *kNN* method detects anomalies by measuring the distance between each sample and its neighbors, with points far away from most others flagged as anomalies [30]. Despite its simplicity, kNN remains a strong and competitive baseline, often performing remarkably well across a wide range of datasets, as demonstrated in recent large-scale anomaly detection benchmarks such as ADBench [15]. Although distance-based methods may not scale well for very high-dimensional data, they provide a valuable reference for evaluating the improvement of VAEs over simpler techniques.
The *Isolation Forest* algorithm works on the principle that anomalies are easier to isolate than normal samples. By randomly splitting the data, anomalous samples tend to be separated in fewer steps, while normal samples require more splits. This makes Isolation Forest not only effective but also efficient and scalable, which has contributed to its popularity in real-world applications. Recent studies include Isolation Forest among the baselines and show that no single unsupervised method consistently outperforms the

others across all datasets [15, 6].

The *Autoencoders* is included as a reconstruction-based baseline. It shares the same overall encoder-decoder structure as the VAE but encodes inputs directly into a single latent vector without sampling or adding a regularization term such as the KL divergence. Anomalies are identified through the reconstruction error, where higher errors suggest a larger deviation from normal data. Including the AE helped us to separate the effect of probabilistic modelling in the VAEs. AE-based anomaly detection is one of the classical deep methods and has been widely studied. Recent surveys [24] highlight its trade-offs, reproducibility challenges, and performance variation across different architectures.

Finally, Principal Component Analysis (PCA) is used as a classical linear baseline. PCA projects data to a lower-dimensional space and reconstructs it using principal components. Samples with high reconstruction errors are considered anomalies. Although simple, PCA is still widely used due to its speed and interpretability, as demonstrated in Shyu et al. [35].

All baseline models were implemented and evaluated using the PyOD library [45], which provides standardized implementations of common anomaly detection methods. Each model is evaluated under the same preprocessing and experimental setup as the VAE variants to ensure a fair comparison. Performance is measured using ROC AUC and PR AUC, along with training and inference time, to capture both effectiveness and computational efficiency.

5.5 Hyperparameter Optimization

Hyperparameter Optimization (HPO) was performed to ensure a fair comparison of the different VAE variants. Rather than relying on default configurations, we systematically searched for the hyperparameters that achieved the best performance and then evaluated how well they generalized to unseen datasets. To avoid overfitting hyperparameters to a specific subset, we adopted a two-phase cross-evaluation procedure.

5.5.1 Motivation and Challenges

The main motivation for hyperparameter optimization was to ensure that the comparison between VAE variants was both fair and realistic. VAEs are highly sensitive to parameters such as latent dimensionality, learning rate, and batch size, and relying on default settings could produce misleading conclusions or unfairly disadvantage certain models. Proper tuning should not only improve performance but also help ensure that

only well-generalized hyperparameter configurations are considered suitable for practical deployment.

However, hyperparameter optimization for anomaly detection introduces unique challenges beyond those seen in standard machine learning tasks. Most datasets are highly imbalanced, with very few anomalies compared to normal samples. This makes it hard to define reliable validation criteria and increases the risk of overfitting to rare cases. In many situations, labeled anomalies are unavailable during training, which further complicates model selection.

Tuning across all datasets at once may lead to benchmark-specific overfitting, while tuning each dataset separately is computationally expensive and does not give a clear overall picture. Balancing these aspects requires a practical optimization strategy, described in the next section.

5.5.2 Optimization Setup

To design a fair procedure, the 121 datasets were divided randomly into two groups: Group A (60 datasets) and Group B (61 datasets). In **Phase 1**, hyperparameters were tuned on Group A, and then the best performing configuration was evaluated in Group B. In **Phase 2**, the process was reversed. This two-phase cross-evaluation procedure, developed as part of this work, ensured that every tuned configuration was tested on previously unseen datasets, reducing the risk of overfitting to a specific group and enabling a fair comparison across models.

To aggregate results fairly, only the evaluations on unseen datasets were considered. A weighted average was then computed to reflect the sizes of both groups, as shown in the following equation:

$$\text{Combined cross score} = \frac{(\text{Group B AUC from A} \times 61) + (\text{Group A AUC from B} \times 60)}{121}$$

Where **Group B AUC from A** is the average result of the configuration tuned on Group A when evaluated on Group B datasets (from Phase 1), and **Group A AUC from B** is the average result of the configuration tuned on Group B when evaluated on Group A datasets (from Phase 2). This formulation ensured that both groups contributed proportionally to the overall performance measure.

For the final selection, each VAE variant produced two candidate configurations (one from each phase). We reported the combined cross-score for transparency, but selected as the final configuration the one that achieved the highest performance on its unseen evaluation

group. In addition, per-dataset AUC values from both phases were recorded separately to provide a detailed breakdown of model behavior across datasets.

Hyperparameter Optimization was performed using FLAML [43] with the BlendSearch Strategy, which efficiently explores large and mixed search spaces while maintaining a balance between exploration and exploitation [43]. Each trial was trained on all datasets within the tuning group, and the objective function was defined as the mean ROC AUC and PR AUC across those datasets. To avoid unstable configurations from manipulating results, any failed or invalid trial was assigned a penalty score of -1 for the affected dataset, guiding the search toward stable and reliable configurations that performed consistently across diverse data conditions.

Table 5.1: Hyperparameter search space per VAE variant.

Model	Latent dim.	Batch	LR	Epochs	Dropout	Further Parameters
Normal VAE	$\{2, 4, 8\}$	$\{16, 32, 64\}$	$[10^{-4}, 10^{-2}]$	$\{30, 50, 100\}$	$[0.0, 0.5]$	$\beta \in \{0.5, 1.0, 2.0\}$
Overcomplete VAE	{input_dim, $2 \times$ input_dim, 64}	$\{16, 32\}$	$[10^{-4}, 10^{-3}]$	$\{100, 200, 300\}$	$[0.0, 0.3]$	$\beta \in \{0.5, 1.0\}$
HVAE	$z_1 \in \{8, 16, 32\}$; $z_2 \in \{4, 8, 16\}$	$\{32, 64\}$	$[10^{-4}, 10^{-2}]$	$\{100, 200, 300\}$	$[0.0, 0.3]$	–
Deep HVAE	$z_1 \in \{8, 16, 32\}$; $z_2 \in \{4, 8, 16\}$; $z_3 \in \{2, 4, 8\}$	$\{32, 64\}$	$[10^{-4}, 10^{-2}]$	$\{100, 200, 300\}$	$[0.0, 0.3]$	–
β-VAE	$\{8, 16, 32\}$	$\{32, 64\}$	$[10^{-4}, 10^{-3}]$	$\{50, 100\}$	$[0.0, 0.3]$	$\beta \in \{1.0, 2.0, 4.0\}$
FactorVAE	$\{8, 16, 32\}$	$\{32, 64\}$	$[10^{-4}, 10^{-3}]$	$\{100, 150\}$	$[0.0, 0.3]$	β=1.0; γ tuned internally
β-TCVAE	$\{8, 10, 16\}$	$\{32, 64\}$	$[10^{-4}, 10^{-3}]$	$\{100, 200\}$	–	$\beta \in \{2.0, 4.0, 6.0\}$
SparseVAE	$\{8, 16, 32\}$	$\{32, 64\}$	$[10^{-4}, 10^{-3}]$	$\{50, 100\}$	$[0.0, 0.3]$	$\beta \in \{1.0, 2.0\}$; $\lambda_1 \in [10^{-5}, 10^{-2}]$
CVAE	$\{8, 10, 16\}$	$\{32, 64\}$	$[10^{-4}, 10^{-3}]$	$\{50, 100\}$	$[0.0, 0.3]$	–

A search space was defined for each VAE variant to account for both shared and model-specific parameters. The common parameters include latent dimension, learning rate, batch size, number of epochs, and dropout rate. Additional parameters, such as β (for β-VAE and β-TCVAE) or the L1 regularization weight (for Sparse VAE), were included where relevant. Table 5.1 summarizes the hyperparameter ranges used for each model. The learning rate is sampled on a log-uniform scale to allow efficient exploration across several orders of magnitude. Together, these design choices enabled a consistent and reproducible tuning process across all VAE architectures.

5.6 Dataset Characteristics Analysis

In addition to comparing models directly, we also examined how dataset properties influence the performance of different VAE variants. For this purpose, key characteristics of each dataset were extracted, including the number of samples, the number of features (dimensionality), and the anomaly ratio (proportion of outliers). Further, we considered measures of feature correlation, principal component explained variance (PCA EV1-EV5), and basic statistics such as the mean and standard deviation of feature values, which capture scale and dispersion.

These characteristics were then merged with the recorded performance metrics of each model to enable a joint analysis. This approach helped us to study whether certain dataset conditions, such as high dimensionality or extreme class imbalance, favor specific VAE variants. In addition, it provided insights into how well the models generalize across datasets of varying size and complexity. To explore how dataset characteristics relate to model performance, we used Spearman's rank correlation coefficient (ρ). This measure captures monotonic relationships without assuming linearity, which makes it well-suited for datasets that vary in scale and distribution.

By linking performance to dataset characteristics, the study aims to identify patterns that can guide researchers in selecting the most suitable VAE variant for a given type of data. While this section outlines the methodology for extracting and integrating these characteristics, the findings and interpretation are discussed in detail in Chapter 6.

6 Experiments and Results

This chapter presents the experimental results of the benchmarking study. We evaluate nine VAE-based models that we described in Section 3, together with the baselines kNN, Autoencoder, PCA, and Isolation Forest. Model performance is measured as mentioned in Section 5.2 using the ROC AUC and PR AUC as primary accuracy metrics, while training and inference times are reported to assess efficiency. The results cover performance with default and tuned configurations, the effect of hyperparameter optimization along with the optimization history, comparison with baselines, and the influence of dataset characteristics in VAE variants.

6.1 Performance with Default Parameters

To begin the evaluation, all VAE variants were trained and tested with their default hyperparameter settings. This provided us with a baseline view of how the models perform in their initial configuration, without any tuning or optimization.
Table 6.1 summarizes the default hyperparameters applied to each variant. The settings differ across models, with hierarchical VAEs using multiple latent layers, the Overcomplete VAE scaling its latent dimension with the input size, and the disentanglement-focused variants relying on higher latent dimensions together with additional regularization terms such as β or L1 penalties. These differences shape the performance and efficiency outcomes discussed in this section.

Table 6.1: Default hyperparameters used for VAE variants

Model	Latent dimension	Batch	LR	Epochs	Further Parameters
NormalVAE	2	32	1e-3	50	dropout=0.1, beta=1.0, contamination=0.1
OvercompleteVAE	max(2*input_dim, 64)	18	5e-4	266	dropout=0.1, beta=1.0, contamination=0.1
2-HVAE	z1=16, z2=8	64	1e-3	200	hidden_dims=[256]
3-HVAE	z1=16, z2=8, z3=4	64	1e-3	200	hidden_dims=[256]
SparseVAE	10	64	1e-3	50	beta=1.0, L1 λ=1e-3
BetaVAE	10	64	1e-3	50	β=4.0
BetaTCVAE	16	64	1e-3	50	β=6.0
FactorVAE	10	64	1e-3	50	-
ConditionalVAE	10	64	1e-3	50	condition dim=1

Fig. 6.1 and Fig. 6.2 report the average ROC AUC and PR AUC per VAE variant across all datasets. The result shows that hierarchical architectures (3-HVAE and 2-HVAE) achieve the strongest performance, with ROC AUC values of 0.770 and 0.768, respectively. A similar trend is visible in PR AUC, with 2-HVAE leading with 0.780, followed by 3-HVAE with 0.776. Overcomplete VAE and Normal VAE also perform competitively with values around 0.760 for both ROC AUC and PR AUC. In contrast, disentanglement-focused variants such as Factorvae, Sparsevae, and Beta-tcvae consistently fall behind, with BetaVAE recording the lowest ROC and PR AUC score with 0.729 and 0.728, respectively.

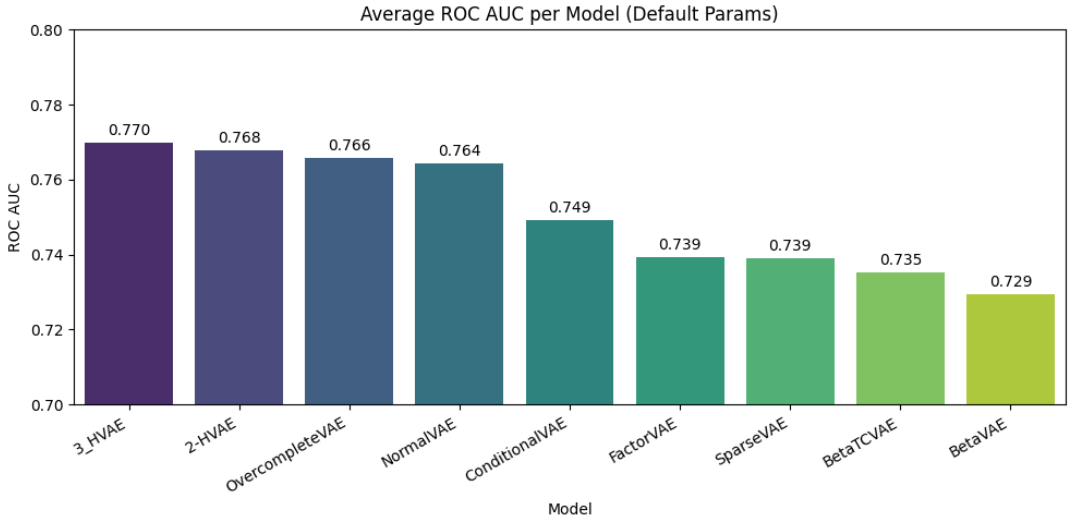


Figure 6.1: Average ROC AUC score per variant with default parameters across all datasets

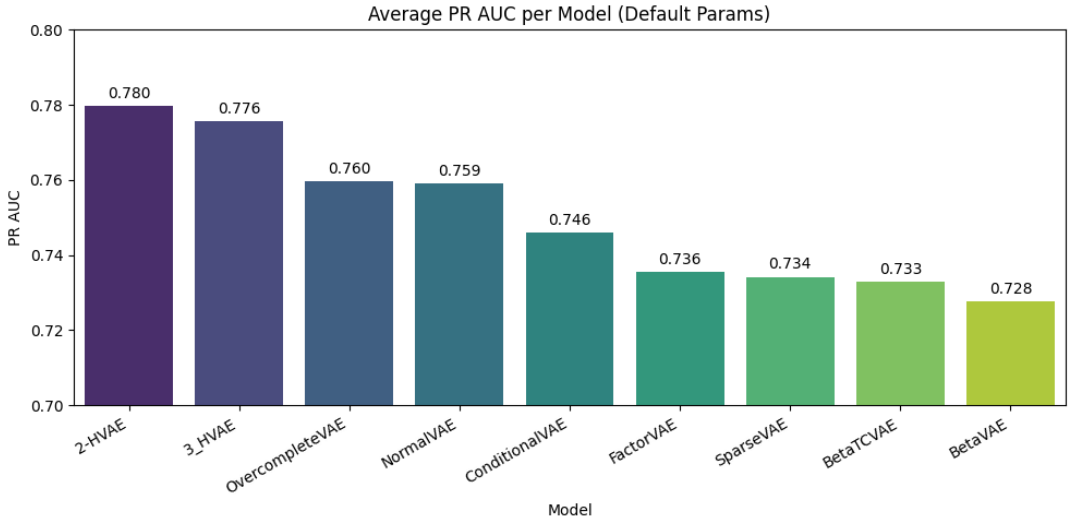


Figure 6.2: Average PR AUC score per variant with default parameters across all datasets

This trend is further supported by the first-place counts shown in Fig. 6.3. 2-HVAE

dominates with more than 30 wins, followed by 3-HVAE with about 24 wins. OvercompleteVAE also wins on a smaller but notable subset, whereas other variants rarely top the ranking. This highlights the strong competitiveness of hierarchical architectures in default settings.

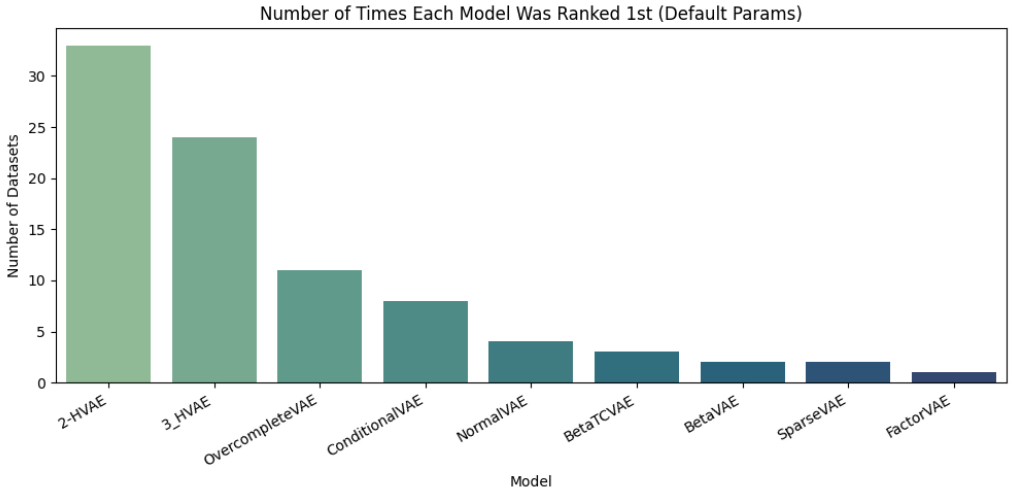


Figure 6.3: First Place counts for each variant across all datasets

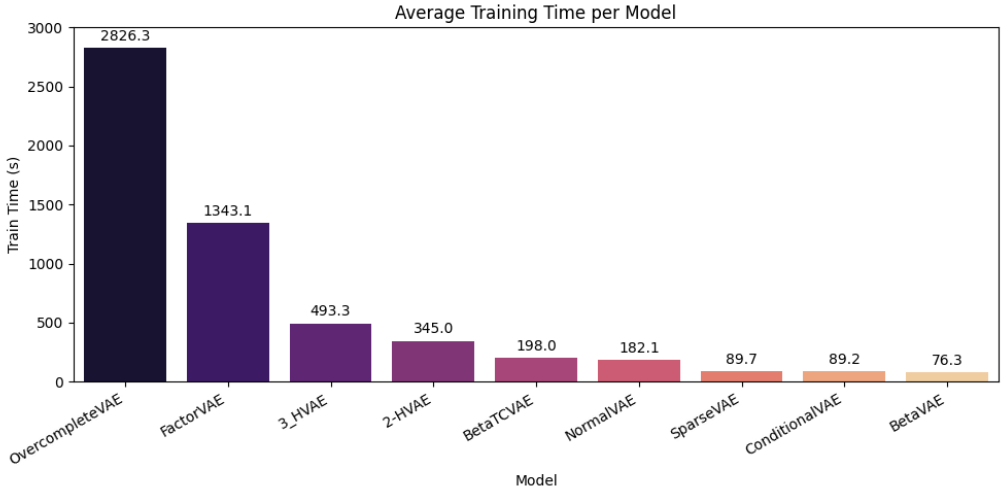


Figure 6.4: Training Time Plot showing the average time taken by each variant to train

Training and Inference efficiency, shown in Fig. 6.4 and Fig. 6.5, vary widely. Overcomplete VAE and FactorVAE are costly to train (2826.3s and 1343.1s on average), while BetaVAE, Conditional VAE, and SparseVAE finish under 100s. The reported training times represent the average duration to train one model on a single dataset, whereas inference time reflects the average time required to generate predictions on the validation set. Inference is generally fast, taking less than 0.06s, except for Overcomplete VAE

(0.224s) and NormalVAE (0.108s). Overall, these findings highlight the need to balance accuracy and efficiency, as some variants suffer notable runtime costs despite relatively simple architectures.

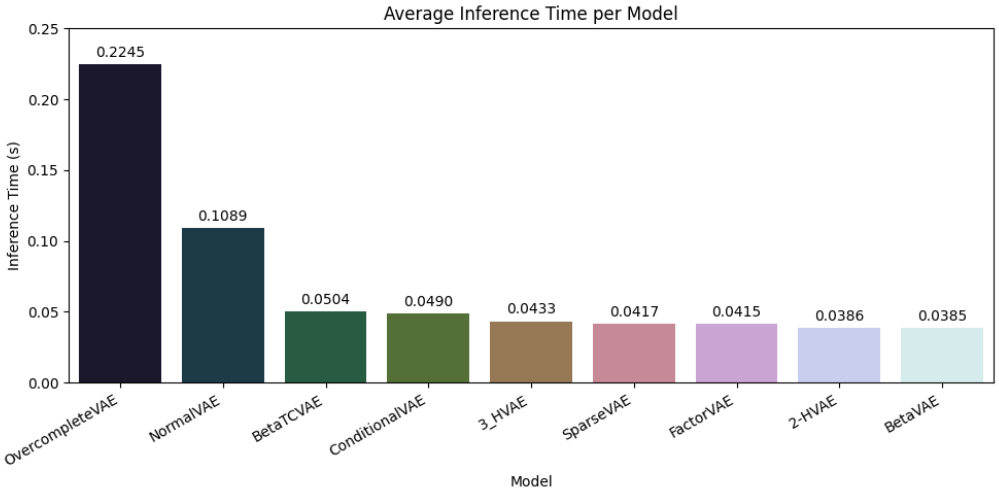


Figure 6.5: Inference Time Plot for each VAE variant

The critical difference (CD) diagram in Fig. 6.6 was produced following the procedure described by Demvsar in 2006 [10]. For each dataset, models were ranked according to their ROC AUC performance. A Friedman test was then applied to determine whether the observed differences across models were statistically significant. To further compare pairs of models, Wilcoxon signed-rank tests were performed with Bonferroni-Holm correction to control for multiple comparisons. The resulting average ranks and significant groupings were then summarized in the CD diagram, where bars connected by a line indicate that the corresponding models are not significantly different from each other. The plot was generated using the *harithaCD* library [20].

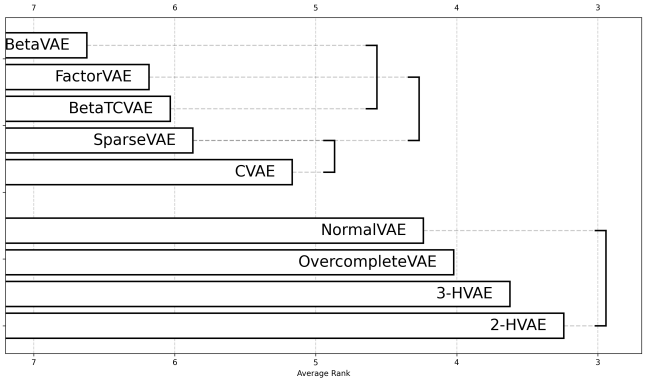


Figure 6.6: Critical Difference (CD) Plot showing the average ranking of VAE variants with default parameters across datasets

The diagram shows that 2-HVAE, 3-HVAE, Overcomplete VAE, and Normal VAE collectively achieve the best average ranks, forming the top-performing group with no significant differences among them. These hierarchical and overcomplete variants perform consistently well across datasets. CVAE ranks slightly lower but remains competitive, while BetaVAE, FactorVAE, BetaTCVAE, and SparseVAE form a cluster of weaker performance. Overall, the results indicate that hierarchical and normal architectures offer more stable performance, whereas disentanglement-focused VAEs perform less effectively under default settings.

In summary, the default parameter experiments reveal three key insights. First, hierarchical VAEs (2-HVAE and 3-HVAE) clearly outperform other variants on average, both in terms of ROC AUC and PR AUC. Second, OvercompleteVAE is competitive in accuracy but suffers from high training and inference costs. Third, disentangled variants such as FactorVAE, BetaVAE, and Beta-TCVAE perform weaker under the default configuration, indicating that they require careful tuning to reach their potential. This ranking is further supported by the critical difference analysis, which places 2-HVAE at the top and shows that disentanglement-focused models form a significantly weaker cluster. Building on these findings, the next section explores to hyperparameter optimization can close this performance gap and alter the relative ranking of the models.

6.2 Effect of Hyperparameter Optimization

6.2.1 Overall Results

Hyperparameter Optimization (HPO) was carried out to test whether the performance of the VAE variants could be improved beyond their default configurations. As described in Section 5.5.2, a two-phase tuning protocol was used, which is also shown visually in Fig. 6.7. The total of 121 datasets was randomly divided into two groups: Group A with 60 datasets and Group B with 61 datasets. In Phase 1, each model was tuned on Group A datasets and then evaluated on Group B datasets. In Phase 2, the process was reversed, creating two independent configurations that were cross-validated across both dataset groups. The optimization setup is summarized in Table 6.2, with further details provided in Section 5.5.2. This design ensured that tuning was not overfit to a single subset and that the resulting configurations generalized better.

Table 6.2: Hyperparameter optimization setup used for all VAE variants.

Setting	Value
Optimizer	FLAML (BlendSearch)
Metrics	ROC AUC, PR AUC
Optimization mode	Maximize
Trials per variant	70
Time budget	15 days ($15 \times 24 \times 60 \times 60$ seconds)
Resources per trial	2 CPUs

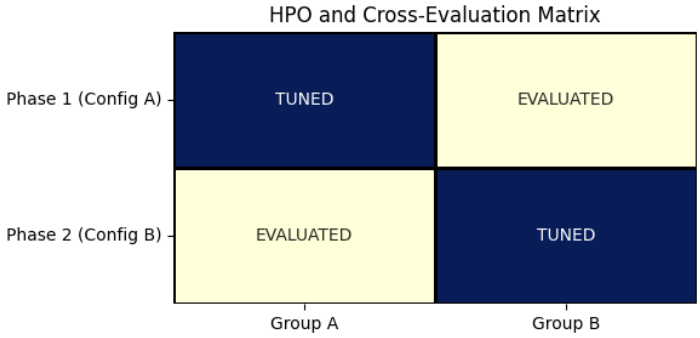


Figure 6.7: Heatmap showing the process of Hyperparameter Optimization

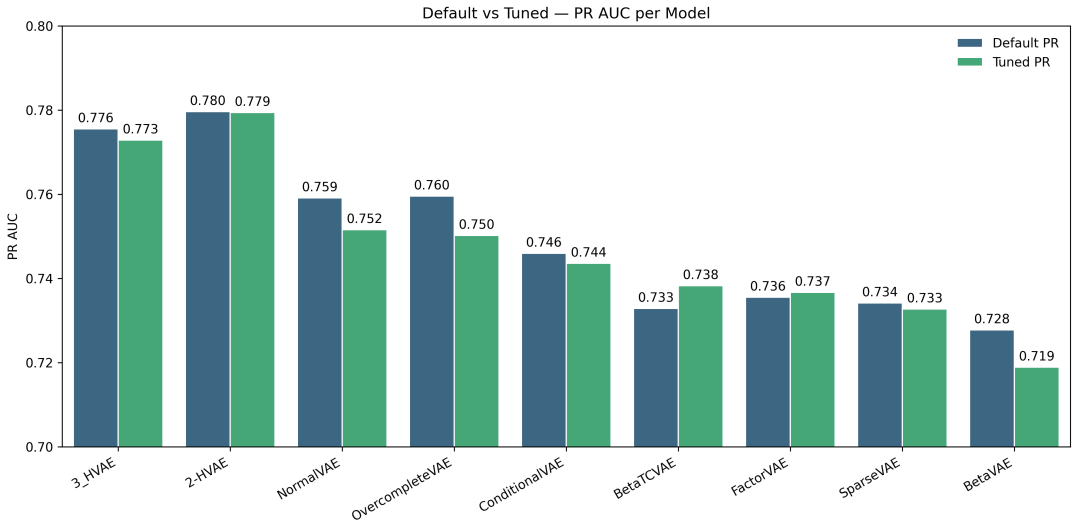


Figure 6.8: Combined PR AUC Score from Phase 1 and Phase 2 after HPO

Figure 6.8 and 6.9 present the results in terms of average PR AUC and ROC AUC across all datasets. The overall picture shows that tuning did not produce major improvements. For PR AUC, the 2-HVAE and 3-HVAE variants remain strongest, reaching 0.779 and 0.773, which is very close to their default performance. For ROC AUC, 3-HVAE achieves

the top score of 0.770, but this is identical to the best default results. Other variants, such as Normal VAE and Overcomplete VAE, reach around 0.755-0.756, which is slightly lower than their default scores. Disentanglement-focused variants (Factor vAE, Beta VAE, Beta TCVAE) remain at the lower end of the ranking with only marginal changes.

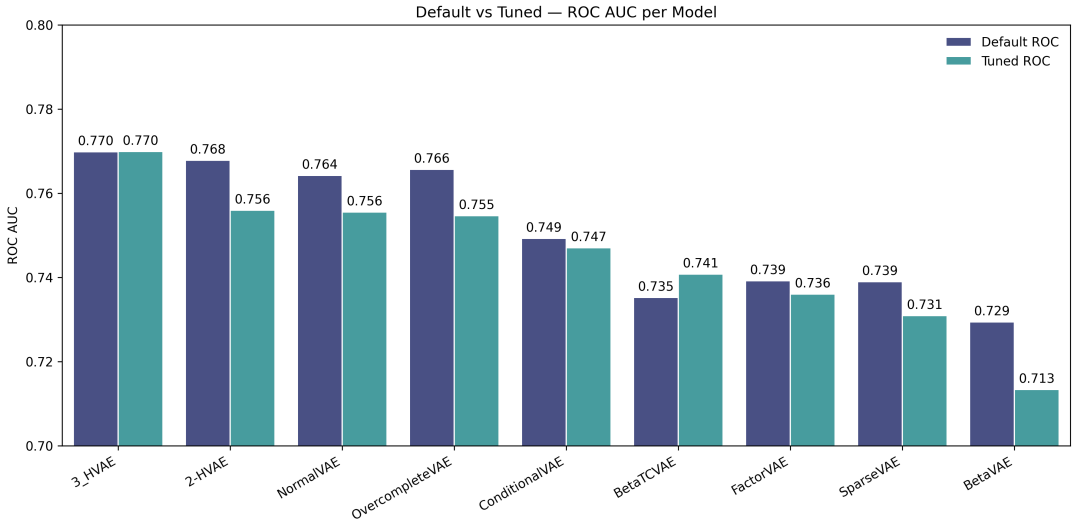


Figure 6.9: Combined ROC AUC Score from Phase 1 and Phase 2 after HPO

The number of first-place wins per dataset, shown in Fig. 6.10, highlights that hierarchical VAEs dominate again after tuning. 3-HVAE and 2-HVAE achieve the most wins (around 35 and around 28, respectively), while Conditional VAE also records some success on specific datasets. In contrast, Normal VAE and the disentanglement variants rarely achieve first place, suggesting that tuning does not improve their relative standing.

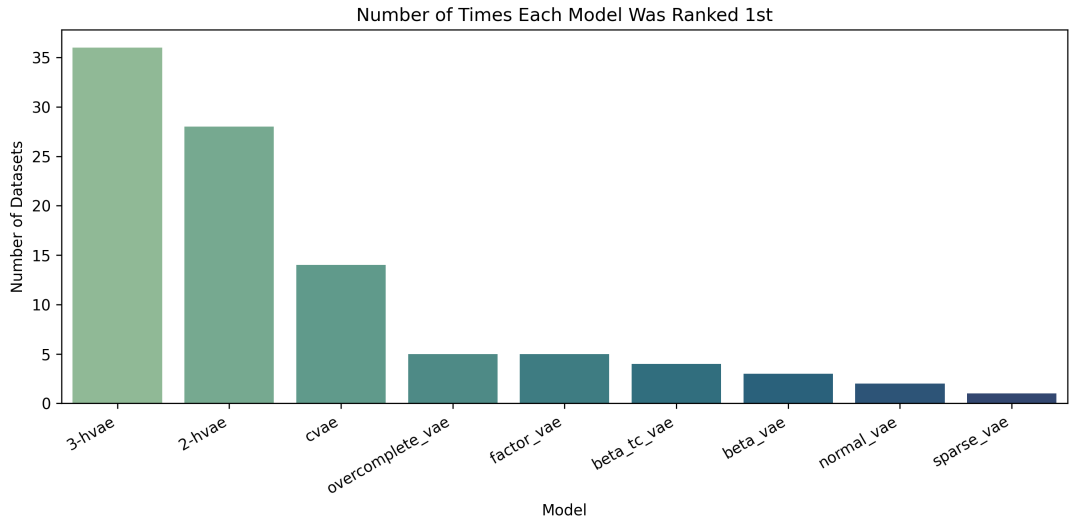


Figure 6.10: First Place counts for each variant after tuning the hyperparameters

The CD diagram in Fig. 6.11 confirms that the ranking structure remains largely unchanged after tuning. 2-HVAE and 3-HVAE again achieve the best average ranks, forming the top-performing group. OvercompleteVAE, NormalVAE, and Conditional VAE follow closely, while the disentanglement-based models (BetaVAE, FactorVAE, BetaTCVAE, and SparseVAE) remain clustered at the lower end. This again indicates that tuning has little effect on the relative performance of the models.

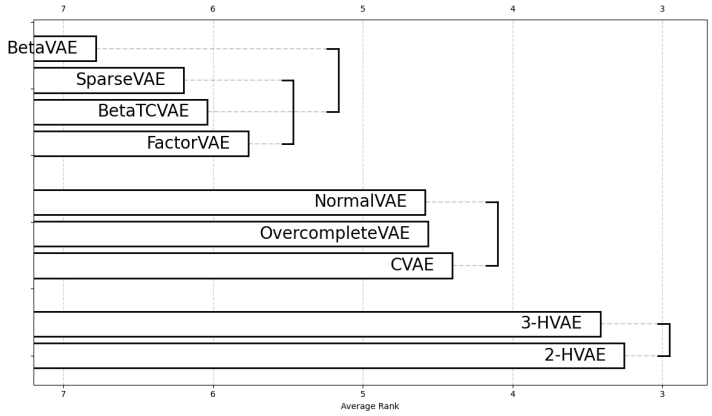


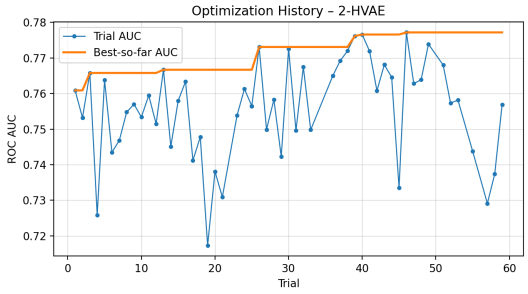
Figure 6.11: Critical Difference (CD) Plot showing the average ranking of VAE variants after HPO

Taken together, these results show that hyperparameter optimization did not substantially improve performance across the benchmark. Hierarchical VAEs remain the best-performing models, but their tuning provided little to no advantage, indicating that either the default settings are already near optimal or that the chosen search space did not contain parameters that could lead to meaningful gains. It might also suggest that the strength of these models comes less from fine-tuned hyperparameters and more from their architectural design, such as hierarchical depth, which provides robustness across different datasets. Another possible reason is that the two dataset groups (A and B) differ quite a bit, which could make it harder for tuned configurations to transfer well between them. It is also possible that, despite the large number of datasets, the available data are still insufficient for robust hyperparameter optimization, making it difficult for tuned parameters to generalize reliably beyond this benchmark.

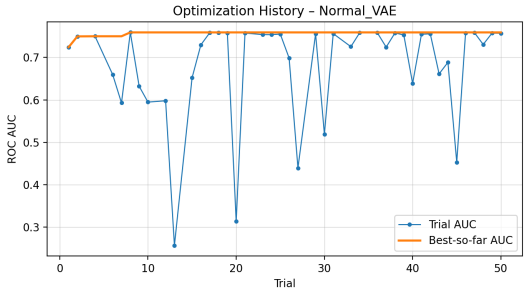
6.2.2 Optimization History

To better understand the tuning process, representative optimization histories are shown in Fig. 6.12. Each plot reports the ROC AUC achieved in individual trials (blue) as well as the best-so-far trajectory (orange). These visualizations highlight how different

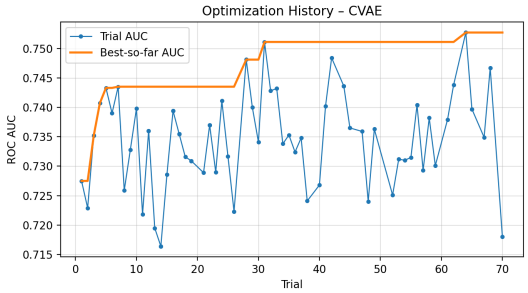
VAE variants respond to the search process and how quickly they converge toward stable performance.



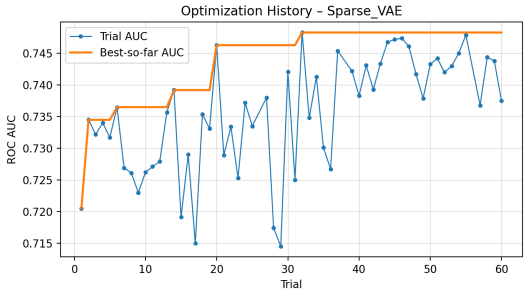
(a) Optimization history for 2-HVAE (Phase 1)



(b) Optimization history for NormalVAE (Phase 2)



(c) Optimization history for CVAE (Phase 2)



(d) Optimization history for Sparse VAE (Phase 1)

Figure 6.12: Optimization histories for selected VAE Variants showing ROC AUC per trial (blue) and the best-so-far performance (orange)

The 2-HVAE improves sharply in the early trials and stabilizes after about 25 trials. This indicates that the model quickly identifies good configurations and that further trials offer diminishing results. The stability of the curve reflects the advantage of architectural depth, which provides robustness and reduces the dependence on fine-grained hyperparameter adjustments. In contrast, the CVAE exhibits strong fluctuations across trials, with gradual improvements that extend beyond the optimization process. This indicates that the model is somewhat sensitive to hyperparameter choices, but it can still benefit from a longer search, maybe with more trials and a larger search space.

The Normal VAE plot reveals a different pattern. Here, the trial performance fluctuates dramatically, with several drops to very low AUC values. At the same time, the best-so-far curve forms an almost flat ceiling, rarely surpassing the early performance levels. This suggests that while Normal VAE can occasionally achieve reasonable results, its potential is capped by its simple architecture. With only a single latent layer and restricted capacity,

the model cannot effectively capture complex structures in the data, so hyperparameter tuning brings little additional benefit. In contrast, more expressive architectures, such as Hierarchical VAEs or Sparse VAE variants, provide greater flexibility, allowing them to achieve higher performance even without extensive tuning.

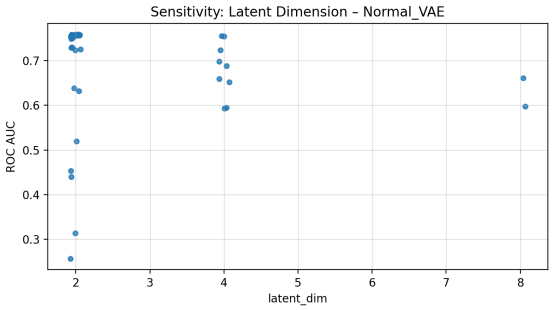
In summary, hierarchical VAEs quickly reached stable performance and proved robust to parameter variation. CVAE and NormalVAE were more sensitive, with Normal VAE showing the strongest instability, and Sparse VAE remained largely unaffected by tuning. These patterns suggest that only a subset of models benefit meaningfully from extensive hyperparameter optimization, and that architectural capacity plays a decisive role in determining this outcome.

6.2.3 Hyperparameter Analysis

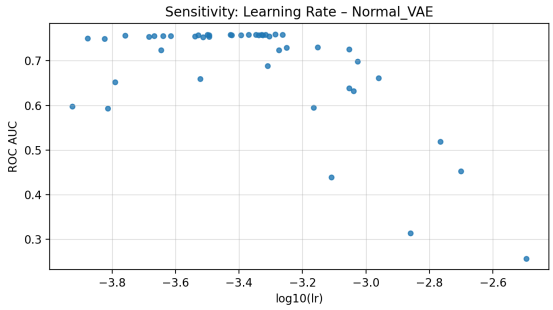
Beyond average results, it is important to examine how individual hyperparameters shaped performance. A sensitive analysis was therefore carried out to understand which parameters drive variability in model outcomes and which models are more robust to parameter choices. To illustrate this, Fig. 6.13 presents sensitivity plots for two representative parameters, latent dimension and learning rate, on two selected variants, Normal VAE (Phase 1) and Factor VAE (Phase 2).

For the Normal VAE, performance was highly sensitive to both parameters. In Fig. 6.13(a), results form clear clusters: latent dimensions of 2 and 4 produced consistently high ROC AUC values, whereas larger dimensions led to unstable and scattered performance. This clustering indicates that only a narrow range of latent dimensions is effective for Normal VAE, while deviations quickly reduce stability. A similar trend appears in Fig. 6.13(b) for the learning rate. Around $log_{10}(lr) \approx -3.3$ to -3.5, results cluster at the upper range of performance, but higher learning rates cause noticeable drops in ROC AUC. These patterns suggest that Normal VAE is more sensitive to hyperparameter choices than other variants, performing best within limited parameter ranges but showing less stability outside them.

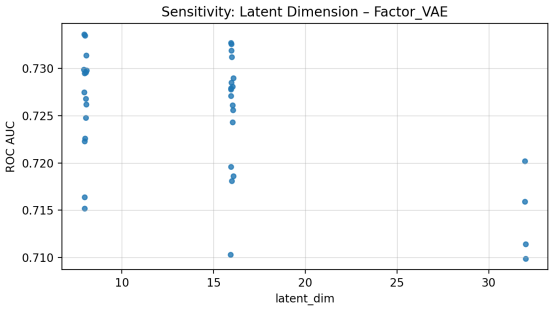
Factor VAE, by contrast, showed more scattered but stable clustering. In Fig. 6.13(c), latent dimensions between 8 and 16 form a dense cluster of competitive results, while only very large values lead to noticeable decline. Similarly, Fig. 6.13(d) shows that performance remains relatively consistent across the tested learning rates, without sharp collapses or outliers. These patterns indicate that Factor VAE is more robust and maintains stable performance across a broader configuration space.



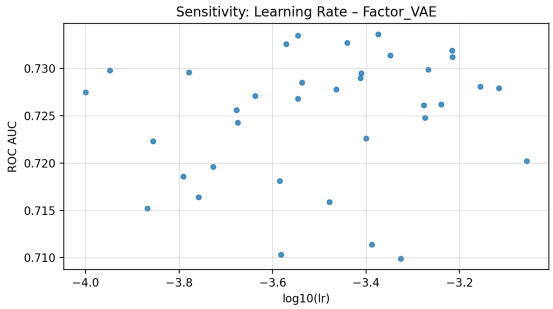
(a) Sensitivity to Latent dimension for Normal VAE (Phase 1)



(b) Sensitivity to Learning rate for Normal VAE (Phase 1)



(c) Sensitivity to Latent dimension for Factor VAE (Phase 2)



(d) Sensitivity to Learning rate for Factor VAE (Phase 2)

Figure 6.13: Sensitivity of ROC AUC to Latent dimension and Learning rate for selected VAE variants

Overall, the clusters observed in the sensitivity plots highlight the difference between fragile and robust models. Simple architectures like Normal VAE depend on narrow parameter ranges, while more expressive models such as Factor VAE tolerate broader settings without major loss in performance.

To complement these findings, the best hyperparameter configurations obtained during the two-phase tuning protocol are summarized in three tables. Table 6.3 lists the best settings identified in Phase 1, where tuning was performed on Group A datasets and evaluated on Group B. Table 6.4 shows the corresponding results from Phase 2, which reversed this process to ensure independence of the configurations. Finally, Table 6.5 consolidates these outcomes by reporting the overall best hyperparameter set for each variant, selected based on which phase achieved stronger generalization. Together, these tables provide a transparent view of the tuning process and clarify how the final configurations were derived.

Table 6.3: Best hyperparameter configurations for each VAE variant from Phase 1, reported with their corresponding ROC AUC and PR AUC.

Model	latent_dim	lr	batch_size	epochs	β	ROC AUC	PR AUC
NormalVAE	2	0.0005	32	100	2.0	0.754	0.748
OvercompleteVAE	64	0.0002	16	100	1.0	0.741	0.735
2-HVAE	z1=8, z2=8	0.0074	64	100	–	0.769	0.772
3-HVAE	z1=8, z2=4, z3=2	0.0039	64	300	–	0.768	0.766
BetaVAE	8	0.0005	64	100	1.0	0.704	0.710
FactorVAE	16	0.0008	32	100	-	0.727	0.727
Beta-TCVAE	16	0.0007	32	200	4.0	0.731	0.730
SparseVAE	16	0.0009	32	100	2.0	0.721	0.722
CVAE	8	0.0002	32	100	–	0.742	0.738

Table 6.4: Best hyperparameter configurations for each VAE variant from Phase 2, reported with their corresponding ROC AUC and PR AUC.

Model	latent_dim	lr	batch_size	epochs	β	ROC AUC	PR AUC
NormalVAE	2	0.0003	32	100	2.0	0.757	0.755
OvercompleteVAE	64	0.0008	16	200	1.0	0.768	0.765
2-HVAE	z1=16, z2=4	0.0011	64	100	–	0.749	0.786
3-HVAE	z1=8, z2=4, z3=2	0.0005	64	100	–	0.770	0.779
BetaVAE	16	0.0007	64	100	4.0	0.716	0.728
FactorVAE	8	0.0004	32	100	-	0.741	0.746
Beta-TCVAE	16	0.0009	32	200	6.0	0.749	0.746
SparseVAE	8	0.0005	32	100	1.0	0.736	0.743
CVAE	8	0.0001	32	50	–	0.752	0.749

Table 6.5: Best hyperparameter configurations for each VAE variant across both tuning Phases, reported with their corresponding ROC AUC and PR AUC.

Model	latent_dim	lr	batch_size	epochs	β	ROC AUC	PR AUC
NormalVAE	2	0.0003	32	100	2.0	0.756	0.752
OvercompleteVAE	64	0.0008	16	200	1.0	0.755	0.750
2-HVAE	z1=8, z2=8	0.0074	64	100	–	0.756	0.779
3-HVAE	z1=8, z2=4, z3=2	0.0005	64	100	–	0.770	0.773
BetaVAE	16	0.0007	64	100	4.0	0.713	0.719
FactorVAE	8	0.0004	64	100	-	0.736	0.737
Beta-TCVAE	16	0.0009	32	200	6.0	0.741	0.738
SparseVAE	8	0.0005	32	100	1.0	0.731	0.733
CVAE	8	0.0001	32	50	–	0.747	0.744

6.3 Comparison with Baselines

To place the VAE variants in context, we compared them against widely used anomaly detection baselines such as k-Nearest Neighbors (kNN), Isolation Forest, PCA, and a standard autoencoder. All baselines were implemented using the PyOD library [45] with default configurations, and anomaly scores were normalized to the [0,1] range before metric computation. Importantly, the baselines were evaluated under the same benchmark protocol as the VAE variants, ensuring a fair and consistent comparison across all methods. Figure 6.14 and 6.15 show the results for both default and tuned configurations.

In the default setting (Fig. 6.14), kNN achieves the highest ROC AUC (0.812), clearly outperforming all VAE variants. The Autoencoder baseline also performs strongly (0.772), achieving scores comparable to 2-HVAE and 3-HVAE. OvercompleteVAE and NormalVAE remain competitive, while disentangled variants such as FactorVAE, SparseVAE, BetaTCVAE, and BetaVAE fall behind both the stronger VAEs and baselines.

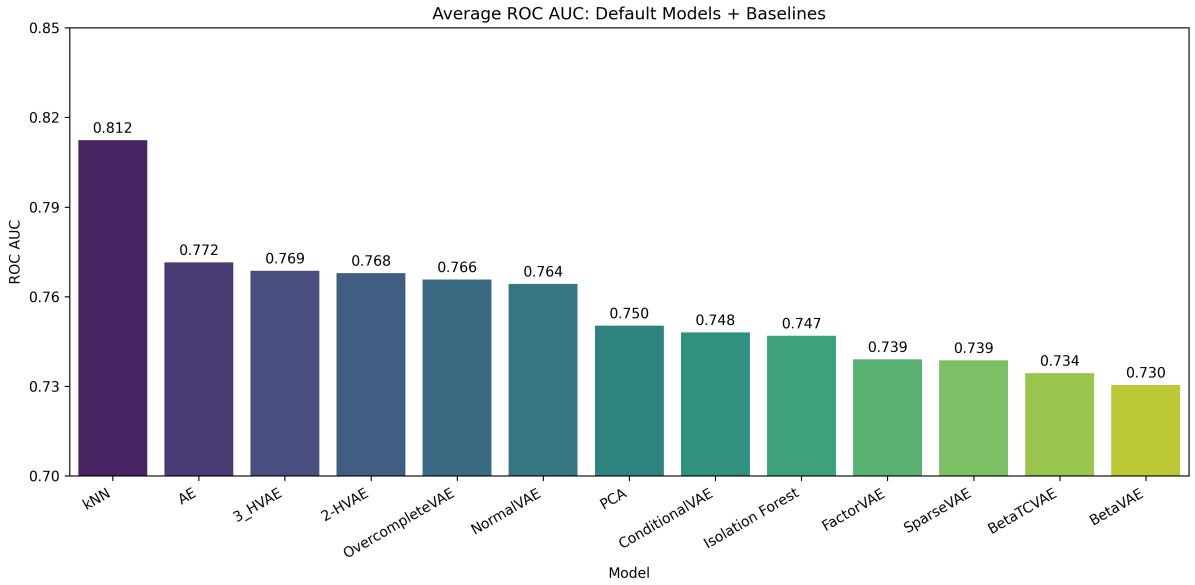


Figure 6.14: Average ROC AUC of VAE variants and classical baselines with default hyperparameters

After hyperparameter optimization (Fig. 6.15), the relative ranking shifts slightly, but the baselines remain highly competitive. kNN retains its leading position, while the Autoencoder continues to achieve scores similar to the best VAE variants. 2-HVAE and 3-HVAE improve slightly with tuning but do not surpass kNN. Meanwhile, disentangled variants only show modest improvements and remain below the baselines on average.

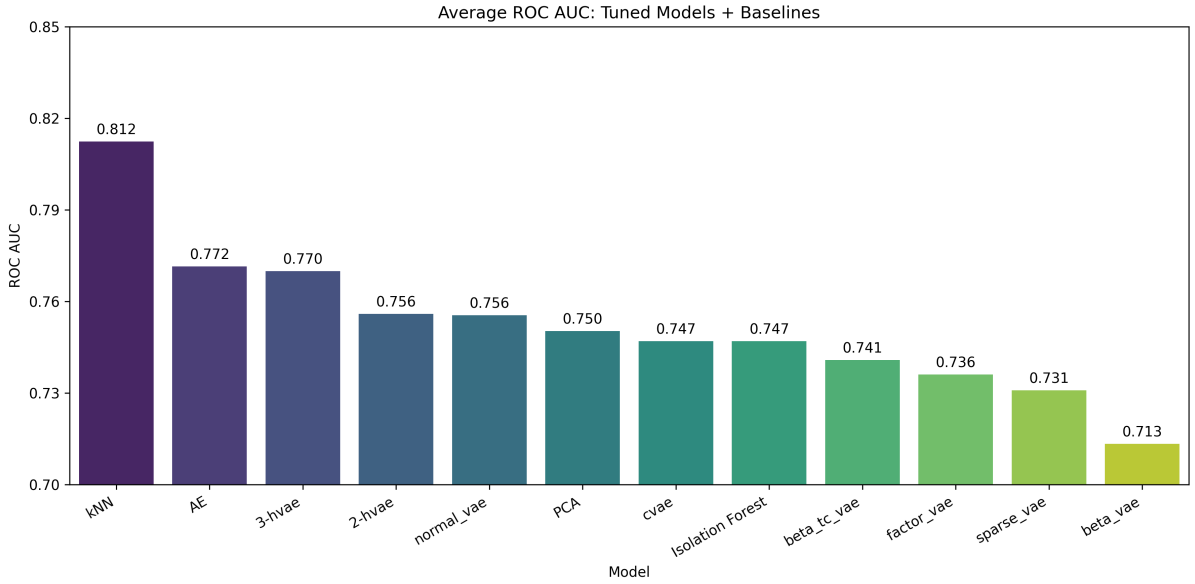


Figure 6.15: Average ROC AUC of VAE variants and classical baselines after hyperparameter optimization

The CD diagram in Fig. 6.16 provides a summary of overall ranking across all models, including the baselines. kNN achieves the best average ranking, followed by 2-HVAE across all modes, including the baselines. 3-HVAE, Normal VAE, and Overcomplete VAE follow closely, and remain statistically comparable. Interestingly, although the Autoencoders reach higher ROC AUC scores than some VAEs in absolute terms, they rank lower overall, indicating that their performance is less consistent across datasets. In contrast, disentanglement-based models occupy the lowest ranks along with PCA and Isolation Forest. These results confirm that while hierarchical and overcomplete VAEs perform competitively, kNN remains the most effective and reliable method overall.

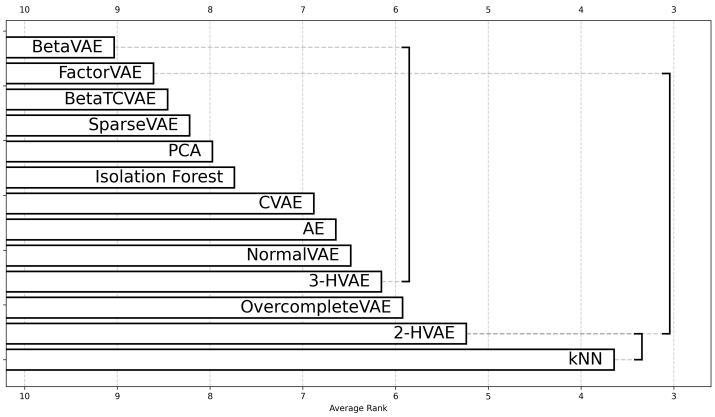


Figure 6.16: Critical Difference (CD) Plot showing the average ranking of VAE variants and classical baselines across datasets

Overall, these results highlight two important observations. First, classical baselines, particularly kNN and the Autoencoder, set a firm benchmark that several VAE variants struggle to exceed. However, the CD analysis shows that the Autoencoder, while achieving high ROC AUC scores, performs less consistently across datasets and is therefore ranked lower overall. Second, while tuning benefits most VAEs, the margin of improvement is not sufficient to close the gap with kNN. This suggests that although VAEs offer flexibility and potential advantages on specific datasets, traditional methods remain powerful competitors in anomaly detection tasks. While these findings may seem somewhat unsatisfactory, we consider it important to report them transparently, as they provide valuable insight into the practical limitations of current VAE models and highlight the need for further research.

6.4 Influence of Dataset Characteristics

Understanding how dataset properties affect the relative performance of VAE variants is essential for deriving practical guidelines. We evaluated the roles of dataset complexity, dimensionality, correlation structure, and stability in determining which models perform best. The goal was not only to identify which models achieve the highest scores, but also to uncover the conditions under which they do so.
To describe the datasets, we extracted several characteristics and merged them with VAE performance for further analysis:

- **Dimensionality** (n_features) - number of input variables.
- **Class balance** (test_anomaly_ratio) - share of anomalies in the test set.
- **Outlier ratio** - the fraction of samples in the training set that have at least one feature exceeding three standard deviations ($|z| > 3$), where scaling parameters (μ and σ) are estimated from the training data itself.

$$\text{outlier\_ratio} = \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} \mathbf{1}\left\{ \max_j \left| \frac{x_{ij} - \mu_j}{\sigma_j} \right| > 3 \right\}$$

  Where n_{train} is the number of training samples, μ_j and σ_j are the feature-wise mean and standard deviation computed from the training set.

- **Average feature correlation** (avg_feature_corr) - mean Pearson correlation between

all pairs of non-constant features:

$$avg\_feature\_corr = \frac{1}{N} \sum_{i<j} \rho(x_i, x_j)$$

Where x_i and x_j are feature vectors, ρ is the Pearson correlation, and N is the number of unique feature pairs.

- **PCA explained variance** (pca_ev_k) - the variance explained by the first five principal components:

$$\lambda_k = \frac{s_k^2}{\sum_{j=1}^{d} s_k^2}, k = 1, 2, .., 5$$

Where s_k^2 are the eigenvalues of the covariance matrix of the standardized data, and d is the total number of retained features. A high λ_k indicates that the dataset is simple, whereas lower values spread across multiple components suggest higher complexity.

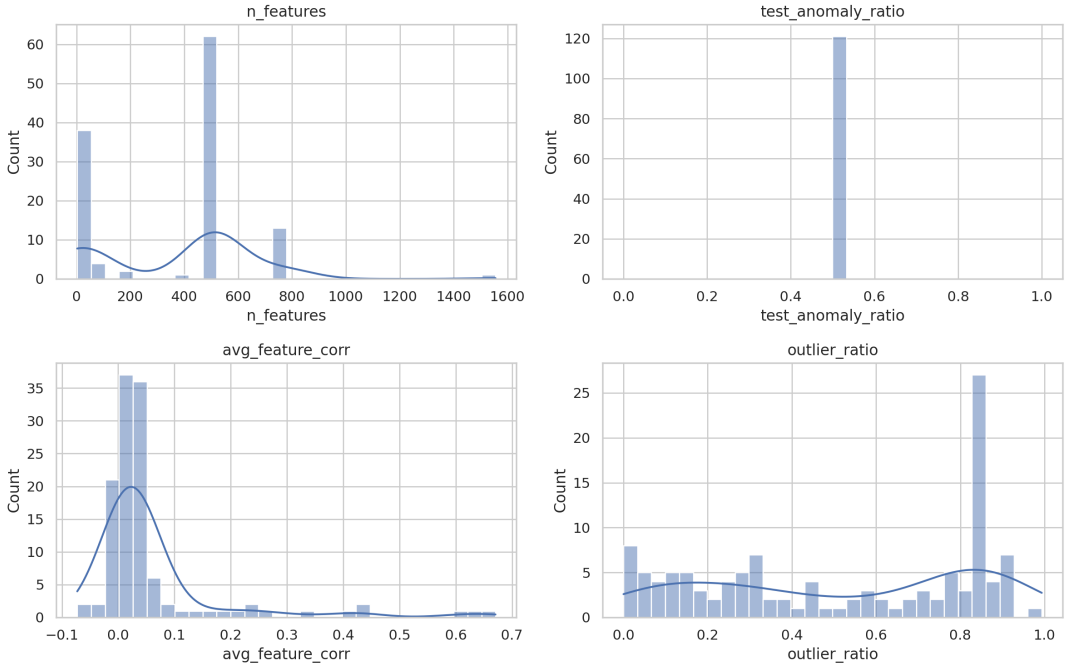


Figure 6.17: Distribution of selected dataset characteristics

The first plot (Fig. 6.17) summarizes the distribution of these characteristics across all datasets. The number of features shows multiple clusters, with some datasets being very low-dimensional, while others concentrate around 500-600 features, and a smaller

group reaching up to 1500 features. The anomaly ratio in the test set is fixed at 0.5 by design, resulting in no meaningful variation across datasets. Average feature correlations are strongly concentrated near zero, suggesting most datasets have little redundancy, although a few exhibit moderate correlation. Outlier ratios display a broad spread, with some datasets containing very few extreme samples, while others have a large majority flagged as outliers. This distribution highlights the diversity of benchmarks, ensuring that both simple and complex dataset structures are represented.

Fig. 6.18 compares the average ROC AUC of VAE variants on low- and high-dimensional datasets. Most models achieve stronger results on low-dimensional data, with Conditional VAE, Factor VAE, and Sparse VAE showing the largest improvements. Beta TCVAE and Beta VAE also drop noticeably when dimensionality increases, reflecting their sensitivity to more complex feature spaces. In contrast, Hierarchical VAEs not only remain stable but even perform slightly better on higher-dimensional datasets, while the Overcomplete VAE and Normal VAE show consistent results across both dimensionality. This suggests that deeper and wider latent structures are particularly effective in managing high-dimensional complexity.

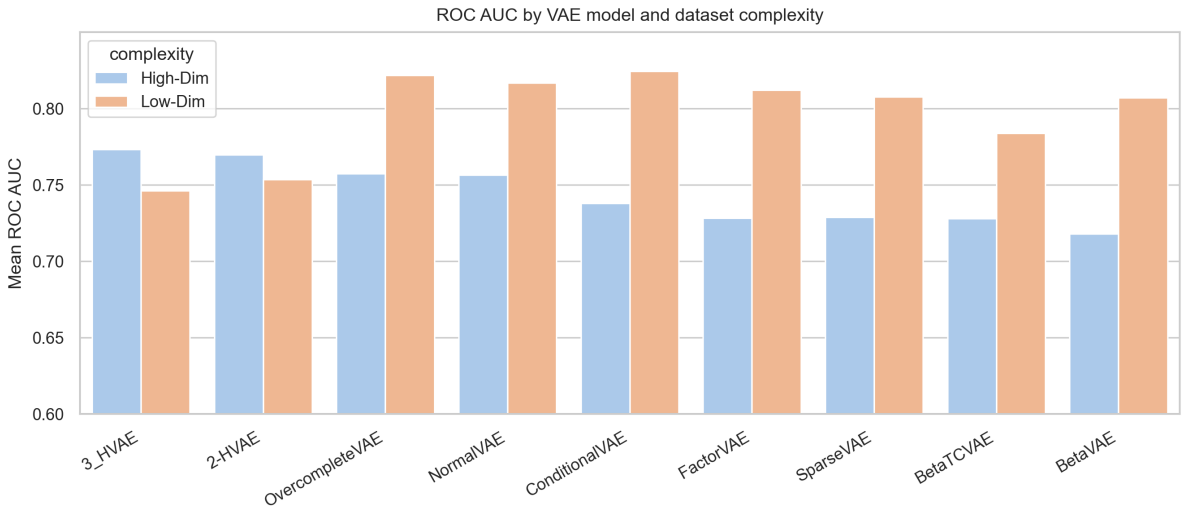


Figure 6.18: Average ROC AUC of VAE variants by dataset complexity

Fig 6.19 visualizes the Spearman rank correlation (ρ) between ROC AUC and dataset characteristics for each VAE variant (red = positive, blue = negative, white ≈ 0). Across nearly all models, the **outlier ratio** shows a consistently negative relationship with performance, meaning that datasets containing a larger share of extreme samples ($|z|>3$) tend to be harder to model effectively. This trend appears systematic rather than model-specific, suggesting that all VAE variants, regardless of architectural depth, struggle when data contain many irregular points. The **number of features** also correlates negatively with

performance, but this effect appears slightly weaker for hierarchical VAEs, suggesting that deeper latent structures may provide a model's advantage in handling high-dimensional data. In contrast, **average feature correlation** shows only weak effects overall: most variants exhibit a mild positive trend, whereas hierarchical VAEs display slightly negative correlations, indicating that they may not benefit from strongly correlated input features to the same extent as other models. **PCA explained variance** in the first components is strongly and consistently positive for all VAEs, implying that datasets where most variance is captured in a few directions tend to yield better anomaly detection performance. While most variants exhibit broadly similar relationships with dataset characteristics, hierarchical VAEs show subtle deviations, appearing slightly less sensitive to high dimensionality and responding differently to correlated features.

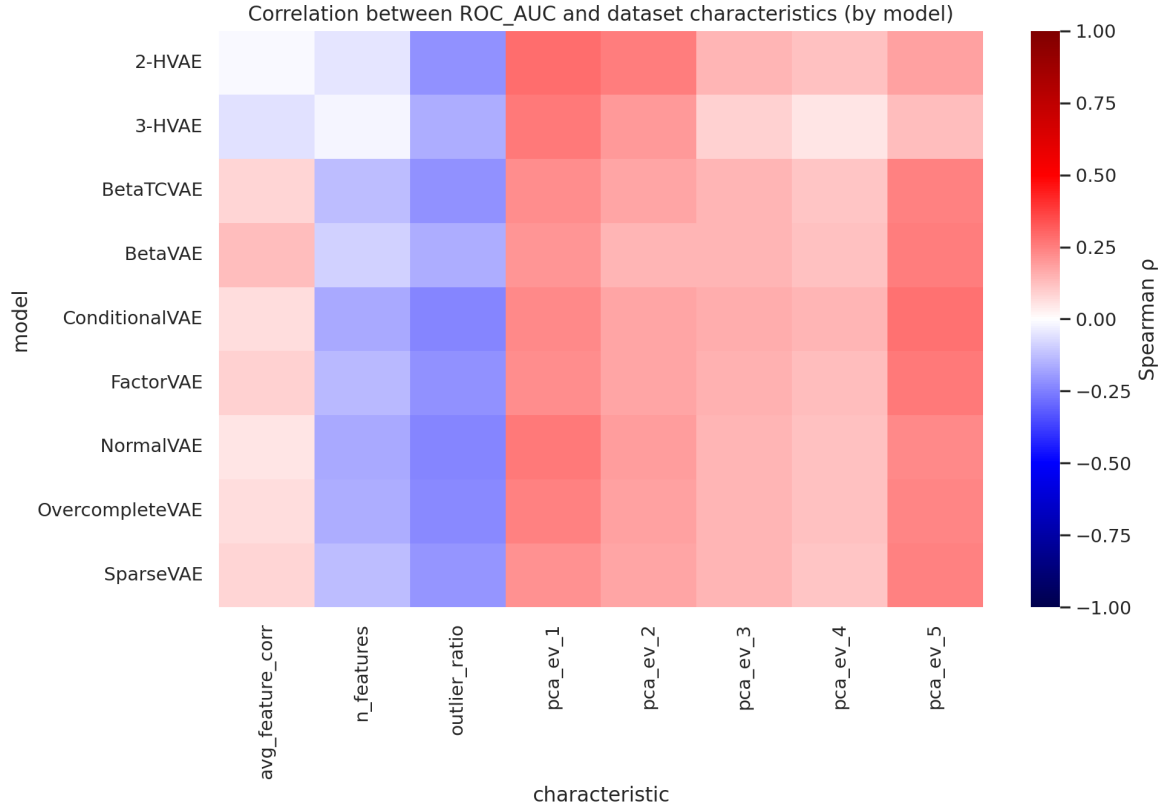


Figure 6.19: Spearman rank correlations between dataset characteristics and VAE performance (ROC AUC), computed per model

The last plot (Fig. 6.20) highlights these findings more directly by showing the win-rates across quartiles of feature dimensionality. In the lowest quartile (Q1), a wide range of models, including NormalVAE, Conditional VAE, and Overcomplete VAE, share competitive performance. However, as dimensionality increases, the advantage shifts clearly

toward Hierarchical VAEs. By the highest quartile (Q3), 2-HVAE and 3-HVAE dominate most datasets, with Overcomplete VAE maintaining a steady presence. This confirms that hierarchical depth and overcomplete representations are particularly effective in high-dimensional environments, while simpler variants remain more suitable for low-dimensional problems. These findings fit with the idea that additional latent capacity is unnecessary in small feature spaces but becomes an advantage when handling complex, high-dimensional inputs.

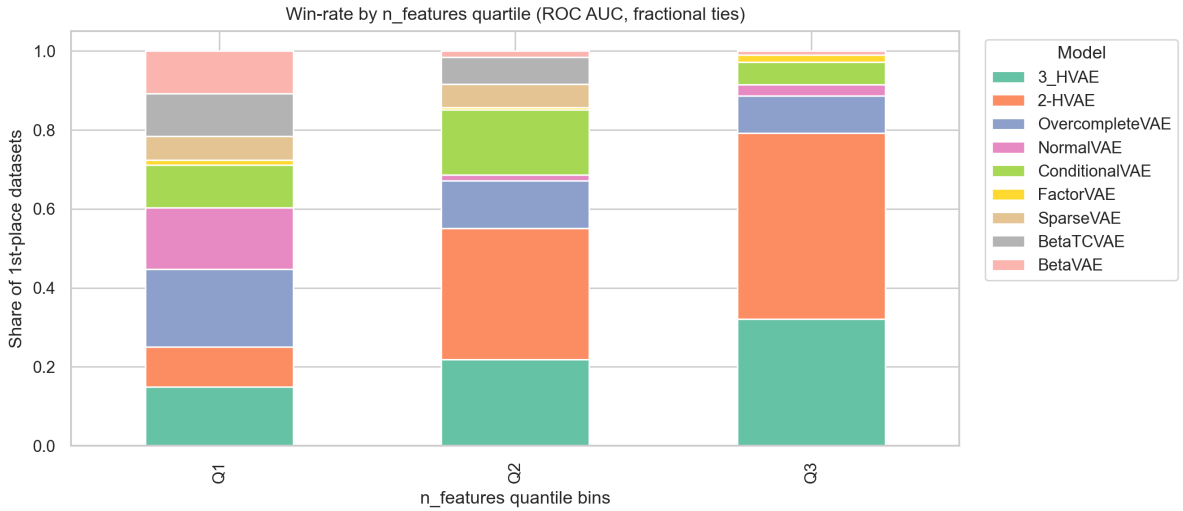


Figure 6.20: Win-rate of VAE variants across feature count quartiles (Q1-Q3)

6.5 Summary of Findings

The experiments reveal several consistent patterns across models and datasets. Hierarchical VAEs (2-HVAE, 3-HVAE) achieved the strongest overall performance in their default configurations, followed by OvercompleteVAE and NormalVAE. Disentanglement-focused variants such as BetaVAE, FactorVAE, and Beta-TCVAE consistently lagged. Training efficiency differed widely: OvercompleteVAE and FactorVAE required the longest training times, while most other models trained faster, and inference remained efficient across all variants.

Hyperparameter optimization did not significantly improve results over the defaults. Hierarchical VAEs remained the best-performing VAEs, which can be explained by their multi-layer latent structure. By introducing several layers of latent variables, these models can capture richer feature dependencies and avoid the bottlenecks present in shallower VAEs. This leads to more accurate reconstructions and stronger anomaly detection signals, particularly in high-dimensional datasets. In contrast, disentanglement-focused

models emphasize latent separation, which is less relevant for anomaly detection and can reduce reconstruction quality. Normal VAE and Overcomplete VAE showed sensitivity to latent dimension and learning rate, and although tuned configurations rarely surpassed defaults, they highlighted parameter ranges that generalize well, reducing the need for repeated HPO across datasets.

Dataset characteristics further shaped the relative performance of VAE variants. Simpler models such as Normal VAE, Conditional VAE, and Factor VAE were competitive on low-dimensional and compressible datasets, while Hierarchical VAEs improved as dimensionality increased and even performed better in high-dimensional settings. A higher share of outliers in the data also boosted the ROC AUC for nearly all models, as anomalies became easier to separate from normal patterns.

When compared with classical baselines, kNN and Autoencoder proved to be strong competitors, particularly on simpler datasets. This shows that while VAEs offer flexibility and robust performance in complex cases, traditional methods remain effective benchmarks for anomaly detection.

7 Conclusion and Future Work

This thesis presented a large-scale comparative study of nine Variational Autoencoder (VAE) variants for unsupervised anomaly detection. Motivated by the lack of standardized evaluation across different VAE architectures, the research established a unified and reproducible benchmarking framework to assess nine representative variants across 121 tabular datasets from the ADBench repository. The study further incorporated a two-phase cross-validation hyperparameter optimization (HPO) process to ensure fair and unbiased evaluation. Through this large-scale analysis, the thesis contributes to a more systematic understanding of how different VAE designs, hyperparameters, and dataset properties influence anomaly detection performance. Building on the detailed findings in Chapter 6, this section interprets the results in relation to the central research questions.

RQ1 - Model effectiveness

Hierarchical VAEs (2-HVAE and 3-HVAE) provided the best overall balance between accuracy, robustness, and scalability. Their layered latent structure enabled them to capture complex feature dependencies more effectively than simpler architectures. Overcomplete and Normal VAEs were competitive but more computationally demanding, whereas disentanglement-oriented variants such as β-VAE, Factor VAE, Sparse VAE, and β-TCVAE offered no clear advantage for reconstruction-based anomaly detection.

RQ2 - Influence of hyperparameters

Hyperparameter optimization did not lead to the significant performance improvements we initially expected. However, it provided valuable insights into how different parameters affect training stability and reconstruction quality across models. The two-phase tuning process revealed consistent sensitivity patterns, particularly for latent dimensionality and learning rate, and helped identify parameter ranges that yield stable, generalizable performance across datasets. Although the overall gains were modest, these findings emphasize the importance of systematic optimization for fair comparison and for understanding the practical limits of each architecture.

RQ3 - Effect of dataset characteristics

Dataset complexity shaped model behavior: hierarchical architectures excelled on high-dimensional or feature-correlated data, while simpler models such as the Normal or Condi-

tional VAE performed adequately for low-dimensional datasets. This suggests that model selection should reflect both data properties and computational constraints, and highlights the strong generalizability of hierarchical VAEs across diverse dataset types.

When compared with classical anomaly detection baselines such as kNN, Isolation Forest, Autoencoder (AE), and Principal Component Analysis (PCA), the VAE-based methods showed comparable but not superior performance. Traditional algorithms remained competitive in accuracy and were considerably more efficient, showing that deep generative models are not always the most practical choice for tabular anomaly detection. It was initially expected that the advanced VAE variants would deliver stronger improvements, but the results revealed only modest gains. This honest outcome emphasizes the importance of empirical evidence over theoretical intuition and highlights that architectural complexity alone does not guarantee better performance.

Future work could extend this benchmark to a wider range of algorithms and diverse dataset types, including computer-generated or artificial datasets with controlled properties. More advanced HPO techniques and broader search spaces may reveal whether some variants can perform better when tuned more extensively. Since hierarchical architectures performed best, hybrid designs that combine hierarchical and disentanglement mechanisms or deeper hierarchical models, such as 4-HVAE or 5-HVAE, could be explored to enhance representational depth. Finally, theoretical analysis of why hierarchical and overcomplete VAEs generalize well, for example, through an information-theoretic or geometric perspective, could offer deeper insight into their latent representations.

In summary, this thesis provides a transparent and reproducible foundation for evaluating VAE variants in anomaly detection. It demonstrates that hierarchical architectures are currently the most effective overall, while classical methods remain valuable baselines. These findings offer realistic guidance for future research and practical deployment of deep generative models for anomaly detection.

List of Figures

2.1 Comparison between AE and VAE architectures. Image adapted from [29] . 7

3.1 Illustration of two-layer Hierarchical Variational Autoencoder (HVAE). The encoder (left, brown arrows) infers latent variables $\mathbf{z_1}$ and $\mathbf{z_2}$ in a bottom-up manner, while the decoder (right, green arrows) reconstructs the data through a top-down generative process. Image Adapted from [39] 11

6.1 Average ROC AUC score per variant with default parameters across all datasets 26
6.2 Average PR AUC score per variant with default parameters across all datasets 26
6.3 First Place counts for each variant across all datasets 27
6.4 Training Time Plot showing the average time taken by each variant to train 27
6.5 Inference Time Plot for each VAE variant 28
6.6 Critical Difference (CD) Plot showing the average ranking of VAE variants with default parameters across datasets 28
6.7 Heatmap showing the process of Hyperparameter Optimization 30
6.8 Combined PR AUC Score from Phase 1 and Phase 2 after HPO 30
6.9 Combined ROC AUC Score from Phase 1 and Phase 2 after HPO..... 31
6.10 First Place counts for each variant after tuning the hyperparameters 31
6.11 Critical Difference (CD) Plot showing the average ranking of VAE variants after HPO 32
6.12 Optimization histories for selected VAE Variants showing ROC AUC per trial (blue) and the best-so-far performance (orange) 33
6.13 Sensitivity of ROC AUC to Latent dimension and Learning rate for selected VAE variants 35
6.14 Average ROC AUC of VAE variants and classical baselines with default hyperparameters 37
6.15 Average ROC AUC of VAE variants and classical baselines after hyperparameter optimization 38

6.16 Critical Difference (CD) Plot showing the average ranking of VAE variants and classical baselines across datasets 38
6.17 Distribution of selected dataset characteristics 40
6.18 Average ROC AUC of VAE variants by dataset complexity 41
6.19 Spearman rank correlations between dataset characteristics and VAE performance (ROC AUC), computed per model 42
6.20 Win-rate of VAE variants across feature count quartiles (Q1-Q3) 43

List of Tables

5.1 Hyperparameter search space per VAE variant. 23

6.1 Default hyperparameters used for VAE variants 25
6.2 Hyperparameter optimization setup used for all VAE variants. 30
6.3 Best hyperparameter configurations for each VAE variant from Phase 1, reported with their corresponding ROC AUC and PR AUC. 36
6.4 Best hyperparameter configurations for each VAE variant from Phase 2, reported with their corresponding ROC AUC and PR AUC. 36
6.5 Best hyperparameter configurations for each VAE variant across both tuning Phases, reported with their corresponding ROC AUC and PR AUC. . . . 36

References

[1] Suman Kalyan Adari and Sridhar Alla. "Introduction to Anomaly Detection". In: *Beginning Anomaly Detection Using Python-Based Deep Learning: Implement Anomaly Detection Applications with Keras and PyTorch*. Berkeley, CA: Apress, 2024, pp. 1–22.

[2] Charu C. Aggarwal. "An Introduction to Outlier Analysis". In: *Outlier Analysis*. Cham: Springer International Publishing, 2017, pp. 1–34.

[3] Haleh Akrami, Sergul Aydore, Richard M. Leahy, and Anand A. Joshi. *Robust Variational Autoencoder for Tabular Data with Beta Divergence*. 2020. arXiv: 2006.08204 [cs.LG].

[4] Jinwon An and Sungzoon Cho. "Variational Autoencoder based Anomaly Detection using Reconstruction Probability". In: 2015.

[5] Ruina Bai, Ruizhang Huang, Yongbin Qin, Yanping Chen, and Chuan Lin. "HVAE: A deep generative model via hierarchical variational auto-encoder for multi-view document modeling". In: *Information Sciences* 623 (2023), pp. 40–55.

[6] Roel Bouman, Zaharah Bukhsh, and Tom Heskes. *Unsupervised anomaly detection algorithms on real-world data: how many do we need?* 2023. arXiv: 2305.00735 [cs.LG].

[7] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM Comput. Surv.* 41.3 (July 2009).

[8] Ricky T. Q. Chen, Xuechen Li, Roger Grosse, and David Duvenaud. *Isolating Sources of Disentanglement in Variational Autoencoders*. 2019. arXiv: 1802.04942 [cs.LG].

[9] Jesse Davis and Mark Goadrich. "The relationship between Precision-Recall and ROC curves". In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, pp. 233–240.

[10] Janez Demšar. “Statistical comparisons of classifiers over multiple data sets”. In: *Journal of Machine learning research* 7.Jan (2006), pp. 1–30.

[11] Xueying Ding, Lingxiao Zhao, and Leman Akoglu. “Hyperparameter Sensitivity in Deep Outlier Detection”. In: *arXiv preprint arXiv:2206.07647* (2022).

[12] Tom Fawcett. “An introduction to ROC analysis”. In: *Pattern Recognition Letters* 27.8 (2006). ROC Analysis in Pattern Recognition, pp. 861–874.

[13] Victor Geadah, Gabriel Barello, Daniel Greenidge, Adam S. Charles, and Jonathan W. Pillow. “Sparse-Coding Variational Autoencoders”. In: *Neural Computation* 36.12 (Nov. 2024), pp. 2571–2601. eprint: `https://direct.mit.edu/neco/article-pdf/36/12/2571/2479569/neco\_a\_01715.pdf`.

[14] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016.

[15] Songqiao Han, Xiyang Hu, Hailiang Huang, Mingqi Jiang, and Yue Zhao. *ADBench: Anomaly Detection Benchmark*. 2022. arXiv: `2206.09426 [cs.LG]`.

[16] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. “beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework”. In: *International Conference on Learning Representations*. 2017.

[17] Haoqi Huang, Ping Wang, Jianhua Pei, Jiacheng Wang, Shahen Alexanian, and Dusit Niyato. “Deep Learning Advancements in Anomaly Detection: A Comprehensive Survey”. In: *IEEE Internet of Things Journal* 12.21 (2025), pp. 44318–44342.

[18] Hyunjik Kim and Andriy Mnih. *Disentangling by Factorising*. 2019. arXiv: `1802.05983 [stat.ML]`.

[19] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: `1312.6114 [stat.ML]`.

[20] Simon Klüttermann. *harithaCD: Haritha Style Critical Difference Diagramm*. `https://github.com/psorus/harithaCD`. Accessed: 2025-11-05. 2025.

[21] Simon Klüttermann, Shubham Gupta, and Emmanuel Müller. “Evaluating Anomaly Detection Algorithms: The Role of Hyperparameters and Standardized Benchmarks”. In: *Proceedings of the 2025 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. to appear. IEEE, 2025.

[22] Mark A. Kramer. “Nonlinear principal component analysis using autoassociative neural networks”. In: *AIChE Journal* 37.2 (1991), pp. 233–243. eprint: `https://aiche.onlinelibrary.wiley.com/doi/pdf/10.1002/aic.690370209`.

[23] Umberto Michelucci. "An Introduction to Autoencoders". In: *CoRR* abs/2201.03898 (2022). arXiv: 2201.03898.

[24] Asif Ahmed Neloy and Maxime Turgeon. "A comprehensive study of auto-encoders for anomaly detection: Efficiency and trade-offs". In: *Machine Learning with Applications* 17 (2024), p. 100572.

[25] Huy Hoang Nguyen, Cuong Nhat Nguyen, Xuan Tung Dao, Quoc Trung Duong, Dzung Pham Thi Kim, and Minh-Tan Pham. "Variational autoencoder for anomaly detection: A comparative study". In: *arXiv preprint arXiv:2408.13561* (2024).

[26] Minjae Ok, Simon Klüttermann, and Emmanuel Müller. *Exploring the Impact of Outlier Variability on Anomaly Detection Evaluation Metrics*. 2024. arXiv: 2409.15986 [cs.LG].

[27] Madalina Olteanu, Fabrice Rossi, and Florian Yger. "Meta-survey on outlier and anomaly detection". In: *Neurocomputing* 555 (2023), p. 126634.

[28] Adrian Alan Pol, Victor Berger, Cecile Germain, Gianluca Cerminara, and Maurizio Pierini. "Anomaly detection with conditional variational autoencoders". In: *2019 18th IEEE international conference on machine learning and applications (ICMLA)*. IEEE. 2019, pp. 1651–1657.

[29] Kurtis Pykes. *Variational Autoencoders: How They Work and Why They Matter*. 2024. URL: https://www.datacamp.com/tutorial/variational-autoencoders (visited on 08/13/2025).

[30] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. "Efficient algorithms for mining outliers from large data sets". In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. SIGMOD '00. Dallas, Texas, USA: Association for Computing Machinery, 2000, pp. 427–438.

[31] Takaya Saito and Marc Rehmsmeier. "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets". In: *PLOS ONE* 10 (Mar. 2015), pp. 1–21.

[32] Rushikesh Shende. *Autoencoders, Variational Autoencoders (VAE) and -VAE*. 2023. URL: https://medium.com/@rushikesh.shende/autoencoders-variational-autoencoders-vae-and-%CE%B2-vae-ceba9998773d (visited on 08/03/2025).

[33] Assaf Shmuel, Oren Glickman, and Teddy Lazebnik. "A comprehensive benchmark of machine and deep learning across diverse tabular datasets". In: *arXiv preprint arXiv:2408.14817* (2024).

[34] Ravid Shwartz-Ziv and Amitai Armon. “Tabular data: Deep learning is not all you need”. In: *Information Fusion* 81 (2022), pp. 84–90.

[35] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. “A novel anomaly detection scheme based on principal component classifier”. In: (2003).

[36] Jonas Soenen, Elia Van Wolputte, Lorenzo Perini, Vincent Vercruyssen, Wannes Meert, Jesse Davis, and Hendrik Blockeel. “The effect of hyperparameter tuning on the comparative evaluation of unsupervised anomaly detection methods”. In: *Proceedings of the KDD’21 Workshop on Outlier Detection and Description*. Outlier Detection and Description Organising Committee. 2021, pp. 1–9.

[37] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. “Learning Structured Output Representation using Deep Conditional Generative Models”. In: *NIPS*. 2015, pp. 3483–3491.

[38] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. *Ladder Variational Autoencoders*. 2016. arXiv: `1602.02282 [stat.ML]`.

[39] Jiaming Song and Shengjia Zhao. *Learning Hierarchical Features from Generative Models*. 2018.

[40] A.K Subramanian. *PyTorch-VAE*. `https://github.com/AntixK/PyTorch-VAE`. 2020.

[41] Akanksha Toshniwal, Kavi Mahesh, and R. Jayashree. “Overview of Anomaly Detection techniques in Machine Learning”. In: *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. 2020, pp. 808–815.

[42] Arash Vahdat and Jan Kautz. *NVAE: A Deep Hierarchical Variational Autoencoder*. 2021. arXiv: `2007.03898 [stat.ML]`.

[43] Chi Wang, Qingyun Wu, Markus Weimer, and Erkang Zhu. “Flaml: A fast and lightweight automl library”. In: *Proceedings of machine learning and systems* 3 (2021), pp. 434–447.

[44] Bang Xiang Yong and Alexandra Brintrup. “Do autoencoders need a bottleneck for anomaly detection?” In: *IEEE Access* 10 (2022), pp. 78455–78471.

[45] Yue Zhao, Zain Nasrullah, and Zheng Li. “PyOD: A Python Toolbox for Scalable Outlier Detection”. In: *Journal of Machine Learning Research* 20.96 (2019), pp. 1–7.

Eidesstattliche Versicherung

(Affidavit)

Poudel, Apeksha | 231594

Name, Vorname (surname, first name) | Matrikelnummer (student ID number)

☒ Bachelorarbeit (Bachelor's thesis) ☐ Masterarbeit (Master's thesis)

Titel (Title)

Performance Evaluation and Comparative Study of Variational Autoencoder Variants for Anomaly Detection

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem oben genannten Titel selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I declare in lieu of oath that I have completed the present thesis with the above-mentioned title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution before.

Essen, 12.11.2025

Ort, Datum (place, date) | Unterschrift (signature)

Belehrung:
Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin") zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:
Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to EUR 50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the Chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, Section 63 (5) North Rhine-Westphalia Higher Education Act (*Hochschulgesetz, HG*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund University will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:*

Essen, 12.11.2025

Ort, Datum (place, date) | Unterschrift (signature)

***Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.**