technische universität
dortmund

Enhancing Anomaly Detection through Test-Time Training

Master's Thesis

Minjae Ok
Master Data Science

First reviewer: Prof.Dr.Emmanuel Müller
Second reviewer: Simon Klüttermann
Faculty of Computer Science
Chair of Data Science and Data Engineering
Technische Universität Dortmund
2024

Abstract

This thesis investigates the application of test-time training principles to enhance traditional anomaly detection algorithms, offering a balance between computational efficiency and performance. Inspired by the DOUST framework, this study introduces three novel methods: Strategically Weighted Isolation Forests at Test Time (SWIFT), kNN with Adaptive Test-time Adjustment through Nonuniform Adaptation (KATANA), and Continuous One-class Support Vector Machine Optimization through test-time weight Sampling (COSMOS).

These methods incorporate dynamic weighting mechanisms to adaptively improve anomaly detection during the test phase. SWIFT significantly improves upon the standard Isolation Forest, demonstrating remarkable performance gains that highlight its adaptability and robustness across diverse datasets. KATANA introduces test-time feature weighting to kNN, achieving statistically significant improvements over the standard kNN. COSMOS extends test-time adaptation to kernel-based methods, and while the ROC AUC scores generally are higher, the improvement is not statistically significant, indicating potential for further refinement.

Extensive experiments across multiple datasets demonstrate that the proposed approaches consistently improve anomaly detection performance compared to their baseline models. The results highlight the feasibility of integrating adaptive strategies into traditional methods, achieving competitive results without the complexity or resource demands of deep learning models.

This work bridges the gap between classical and adaptive frameworks in anomaly detection, advancing the theoretical and practical understanding of test-time training. By enhancing traditional models with adaptive mechanisms, the thesis contributes to the development of efficient and versatile tools for detecting anomalies in varied and resource-constrained environments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The increasing complexity of modern systems and the proliferation of digital data have made anomaly detection a cornerstone of data-driven decision-making in domains such as fraud prevention, network security, and healthcare monitoring [4]. Identifying rare and unexpected patterns in data is critical for maintaining system reliability and mitigating risks. Traditional anomaly detection algorithms have long been valued for their simplicity, efficiency, and effectiveness across a variety of applications [9].

Despite their widespread use, traditional anomaly detection algorithms often operate under static assumptions, using fixed parameters learned during training. They operate under the assumption that anomalies differ from normal data but may not account for the evolving nature of anomalies in real-world scenarios. As new types of anomalies emerge with unseen patterns, static models can struggle to detect them effectively [18].

Test-time training addresses this limitation by enabling models to adapt dynamically using unlabeled test data [28]. By adjusting to the specific characteristics of the test data without prior assumptions about anomaly distributions, models become more robust in identifying novel anomalies. While test-time training has been successfully applied in deep learning models [14], its integration into traditional anomaly detection algorithms has remained largely unexplored.

This thesis bridges that gap by proposing a comprehensive framework for integrating test-time training principles into traditional anomaly detection algorithms. The framework introduces three novel methods, each addressing specific aspects of adaptability during testing:

1. Strategically Weighted Isolation Forests at Test time (SWIFT): SWIFT builds on the Isolation Forest algorithm by introducing a mechanism to assign dynamic weights to individual trees during testing. By optimizing these weights to maximize the separation between normal and anomalous data, SWIFT enhances anomaly detection performance.

Experiments demonstrate that SWIFT achieves consistently higher ROC AUC scores than the standard Isolation Forest across a diverse array of datasets.

2. KNN with Adaptive Test-time Adjustment through Nonuniform Adaptation (KATANA): For k-Nearest Neighbors, KATANA introduces feature weighting optimized during the testing phase. By employing objectives such as maximizing the ROC AUC using pseudo-labels or maximize the separation between train and test data, KATANA identifies and emphasizes features most relevant for anomaly detection. The results reveal that KATANA effectively improves upon the standard kNN algorithm.

3. Continuous One-class Support Vector Machine Optimization through test-time weight Sampling (COSMOS): COSMOS extends the OCSVM algorithm by incorporating kernel weighting during testing. This method dynamically adjusts feature contributions in the kernel function, enabling the model to adapt its decision boundaries. While COSMOS exhibits dataset-dependent performance, it demonstrates the potential for test-time adaptations in enhancing OCSVM's robustness.

The proposed methodology leverages test-time training to dynamically optimize weights using objectives that do not rely on true labels, such as maximizing the difference in anomaly scores between training and test data or the Receiver Operating Characteristic Area Under the Curve (ROC AUC) with pseudo-labels. By utilizing the test data itself, this approach enables models to adapt to new distributions without compromising the unsupervised nature of anomaly detection tasks.

The subsequent sections of this thesis delve deeper into the proposed methodologies, their implementation, and their evaluation. The **Related Work** section reviews existing literature on anomaly detection and test-time training, highlighting gaps that this study aims to address. The **Methodology** section provides a detailed description of the three proposed methods—SWIFT, KATANA, and COSMOS—outlining their theoretical foundations and practical implementations. This is followed by the **Results** section, which presents an empirical evaluation of the methods, comparing their performance against standard algorithms across diverse datasets and discussing key findings supported by statistical analyses. Finally, the **Conclusion and Future Work** section summarizes the contributions of this thesis, reflects on its implications for anomaly detection research, and outlines potential avenues for further exploration, particularly in extending test-time training principles to other domains. By integrating test-time training into traditional anomaly detection algorithms, this thesis not only addresses critical limitations in existing methods but also opens new pathways for advancing the field in dynamic and data-rich environments.

Chapter 2

Related Work

Anomaly detection is a critical task in various fields such as cybersecurity, fraud detection, and system monitoring, where identifying rare and unusual patterns is essential for maintaining operational integrity [4]. Traditional anomaly detection algorithms like Isolation Forest (IF) [17], k-Nearest Neighbors (kNN) [11], and One-Class Support Vector Machine (OC-SVM) [3] have been widely used due to their simplicity and effectiveness. However, these methods often rely on fixed parameters determined during training and may not adjust to the specific characteristics of new data at test time. This can limit their effectiveness in detecting anomalies that present differently from those encountered during training.

Isolation Forest is an ensemble-based method that isolates anomalies by randomly partitioning data using binary trees. While efficient, standard implementations assign equal weight to all trees, potentially limiting the algorithm's adaptability to diverse datasets. Similarly, kNN relies on distance metrics to identify anomalies but typically treats all features equally, which may not capture the nuances necessary for effective detection. OCSVM models the training data by finding a hyperplane that separates data from the origin in a high-dimensional space, but its performance can be sensitive to the choice of kernel and parameter settings.

To address these limitations, recent research has explored methods to enhance traditional algorithms. For example, Feature weighting has been applied to kNN to adjust the influence of individual features based on their relevance to anomaly detection [20][29]. In the context of IF, studies have investigated incorporating feature selection and dynamic weighting to improve detection capabilities [19]. Similarly, for OCSVM, an adaptive weighting scheme has been introduced to refine the decision boundary by assigning higher importance to representative samples during training, improving robustness against outliers [31]. However, these enhancements primarily focus on modifications during the training phase rather than adapting to new data during testing.

Test-time training is a paradigm that allows models to adapt to test data without accessing true labels, enhancing their robustness to distribution shifts [25]. Wang et al. [28] introduced Fully Test-time Adaptation (Tent), which adjusts model parameters during testing to improve performance under distributional changes. This concept has been extended to anomaly detection in deep learning models, most notably in DOUST (Deep OUtlier Selection with Test-time training) [14]. DOUST leverages test-time training by fine-tuning deep neural networks using self-supervised objectives, effectively enhancing anomaly detection without supervised labels during testing.

Despite the success of test-time training in deep learning models, there is a gap in the literature regarding its application to traditional anomaly detection algorithms like IF, kNN, and OCSVM. The question remains whether the benefits of test-time training observed in deep learning models can be transferred to traditional algorithms to enhance their adaptability and performance. Our work addresses this gap by investigating the application of test-time training principles to traditional anomaly detection algorithms.

Chapter 3

Methodology

3.1 Isolaton Forest and SWIFT

3.1.1 Isolation Forest

Isolation Forest is an unsupervised anomaly detection algorithm that isolates anomalies instead of modeling normal data distributions. This is achieved through the construction of multiple binary trees, known as isolation trees, where data points are recursively divided based on randomly selected features and their corresponding split values. The algorithm leverages the principle that anomalies are rare and distinct from the majority of the data, allowing them to be isolated with fewer splits. Consequently, anomalies generally have shorter path lengths in the isolation trees compared to normal points.

The anomaly score in Isolation Forest quantifies the degree of isolation for each data point, calculated as the average path length required to isolate it across all trees. Data points with shorter path lengths are considered more anomalous, while those with longer path lengths are classified as normal.
Using these scores, data points can be ranked from the most to the least anomalous. By setting a threshold on the anomaly scores, the algorithm can effectively classify data points as anomalies or normal instances, enabling robust anomaly detection in various applications [17][16].

The calculation of anomaly scores in Isolation Forest involves several steps:

1. Construction of Isolation Trees:

    - Random Subsampling: Each isolation tree is created using a random subset of the data to ensure diversity among the trees.
    - Recursive Partitioning: The tree is constructed by recursively partitioning the data:

   - Random Feature Selection: At each node, a feature q is randomly selected from the available features.
   - Random Split Value: A split value p is uniformly chosen at random within the range of values for feature q in the current subset of data.
   - Data Partitioning: Data points are partitioned into two subsets based on p:
     * Left child node: Contains data points where $x_q < p$.
     * Right child node: Contains data points where $x_q \geqslant p$.
   - Termination Conditions: The recursive partitioning continues until one of the following conditions is met:
     * The node contains only one data point.
     * The maximum tree depth is reached.
     * All data points have identical values for the selected feature, making further splits impossible.

2. Calculation of Path Lengths:

   - Path Length $h(x)$: For a data point x, the path length is the number of edges traversed from the root node to the terminating node.
   - Adjusted Path Length: If a data point terminates before full isolation (due to identical feature values or reaching maximum depth), an adjustment is made to estimate the expected path length. The adjusted path length $h_{adj}(x)$ is calculated as:

   $$h_{adj}(x) = h(x) + c(s)$$

   where s is the number of data points in the leaf node, and $c(s)$ represents the average path length of unsuccessful searches in a Binary Search Tree, given by:

   $$c(s) = 2H(s-1) - (\frac{2(s-1)}{s})$$

   with $H(i)$ is the i-th harmonic number, approximated as $H(i) \approx ln(i) + \gamma$, and $\gamma \approx 0.5772$ being the Euler-Mascheroni constant [17].

3. Computing the Anomaly Score:

   - Anomaly Score $s(x)$: The anomaly score $s(x)$ for a data point x is then calculated as the negative average of adjusted path length:

   $$s(x) = -E[h(x)] = -\frac{1}{n_{\text{trees}}} \sum_{i=1}^{n_{\text{trees}}} h_{\text{adj},i}(x)$$

where n_{trees} is the total number of trees, and $h_{\text{adj},i}(x)$ is the adjusted path length in tree i. The anomaly score is calculated as the negative average of the adjusted path lengths across all trees. The negative sign ensures that higher anomaly scores correspond to data points with shorter path lengths, indicating a higher likelihood of being anomalies [16].

Anomaly scores quantify how isolated a data point is within the dataset. Higher anomaly scores indicate data points that are more likely to be anomalies, as they have shorter average path lengths across the isolation trees. In contrast, lower anomaly scores indicate data points that resemble the normal data distribution, characterized by longer average path lengths.
To classify data points as normal or anomalous based on their anomaly scores, a threshold τ is established. Data points with anomaly scores exceeding the threshold are classified as anomalies, while those below the threshold are considered normal. This approach provides a binary classification based on anomaly scores.
The decision function leverages the anomaly scores to make predictions as follows:

$$\text{decision\_scores} = s(x) - \tau$$

where $s(x)$ represents the anomaly score. Data points are classified as anomalies if their decision scores are greater than or equal to zero ($s(x) \geqslant \tau$), and those with decision scores less than zero ($s(x) < \tau$) are classified as normal. This approach provides a binary classification based on anomaly scores [1].
By interpreting the anomaly scores and applying an appropriate threshold, the Isolation Forest algorithm enables effective identification of anomalous data points within a dataset.

In standard implementations of Isolation Forest, such as those provided by common machine learning libraries, all trees contribute equally to the calculation of anomaly scores. The scores are derived by averaging the outputs from all trees without differentiating their individual contributions [22][32]. However, this uniform weighting may limit the algorithm's ability to adapt to datasets where certain trees are more effective in identifying anomalies. Therefore, this study investigates assigning dynamic weights to individual trees to enhance anomaly detection performance.
Since available libraries do not support dynamic weighting of trees, a custom implementation of Isolation Forest was developed to facilitate this functionality. This custom implementation replicates the standard behavior of Isolation Forest when equal weights are used but allows for the assignment of different weights to each tree in the ensemble. This flexibility is essential for implementing SWIFT, which extends Isolation Forest by optimizing and applying weights to the trees based on their effectiveness in isolating anomalies.

3.1.2 Test-Time Training

Test-time training is a machine learning paradigm where a model continues to adapt its parameters during the testing phase, using the test data itself without accessing its true labels. This approach is particularly valuable in anomaly detection tasks, where the distribution of anomalies in the test data may differ significantly from that in the training data. By adjusting to the unique characteristics of the test data, the model can enhance its ability to identify anomalies that were not well-represented during training [25].

Traditional machine learning models operate under the assumption that their parameters are fixed after training. Consequently, their performance on new data relies entirely on how well the training data represents future scenarios. However, anomalies are inherently unpredictable and may deviate significantly from learned patterns. Test-time training addresses this limitation by enabling the model to refine its parameters during testing, thus improving its ability to distinguish between normal and anomalous instances [28].

The core idea of test-time training in anomaly detection is to utilize the unlabeled test data to adjust the model in a way that enhances its ability to identify anomalies, without violating the unsupervised nature of the task. This is achieved by defining an objective function that captures desirable properties of the anomaly detection problem, such as maximizing the separation between normal and anomalous data points based on certain criteria [25].

For example, the model can be adjusted to maximize the difference between the training data and the test data, under the assumption that the training data consists mostly of normal instances [14]. By optimizing this objective function, the model places greater emphasis on features or components that are effective in highlighting anomalies in the test data. This approach is not limited to a specific anomaly detection algorithm but can be applied to various traditional methods, including Isolation Forest, k-Nearest Neighbors (kNN), and One-Class Support Vector Machine (OCSVM). In each case, the model's parameters or components are adjusted during testing to improve anomaly detection performance.

By incorporating test-time training, traditional anomaly detection models become more flexible and effective in handling the unpredictable nature of anomalies in real-world data. This paradigm shifts the focus from relying solely on the training data to actively leveraging the test data for model refinement, leading to improved performance in detecting rare and unforeseen anomalies.

In the subsequent sections, this concept will be applied to traditional anomaly detection algorithms, starting with Isolation Forest and extending to methods like kNN and OCSVM. The goal is to demonstrate how test-time training can be effectively integrated into these models to enhance their anomaly detection capabilities across diverse datasets.

3.1.3 Strategically Weighted Isolation Forests at Test time (SWIFT)

The fundamental innovation in SWIFT lies in its ability to dynamically adjust the weights assigned to each tree during the test phase, a feature absent in the standard Isolation Forest. Initially, all trees in SWIFT are assigned equal weights $w_i = 1$ for $i = 1, 2, ..., n_{\text{trees}}$. As test data is introduced, SWIFT adjusts these weights based on each tree's performance in isolating anomalies. Trees that demonstrate better performance are assigned higher weights, allowing the model to adapt to the specific characteristics of the test data. This dynamic weighting mechanism enhances the model's ability to distinguish between normal and anomalous observations, particularly in diverse and complex datasets.

Throughout the testing phase, the weights $\mathbf{w} = [w_1, w_2, ..., w_{n_{trees}}]$ assigned to each tree in the SWIFT model undergo adjustments to enhance the model's ability to discriminate between normal and anomalous observations. This adjustment process is guided by a custom loss function specifically designed to maximize the difference in anomaly scores between the training and test datasets. The anomaly score $s(x)$ for a sample x is calculated as:

$$s(x) = -\sum_{k=1}^{n_{\text{trees}}} w'_k h_k(x)$$

where w'_k are the transformed weights for each tree k and $h_k(x)$ is the path length from tree k for data point x.

Because anomalies are isolated more quickly in isolation trees, they tend to have shorter path lengths. This means that for anomalous data, $h_k(x)$ is smaller, and thus the anomaly $s(x)$ becomes less negative (higher in value). Conversely, normal data points have longer path lengths, resulting in more negative (lower) anomaly scores.

The custom loss function $L(\mathbf{w})$ is defined as:

$$L(\mathbf{w}) = -\left(\frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} s(x_{\text{train},i})^2 - \frac{1}{n_{\text{test}}} \sum_{j=1}^{n_{\text{test}}} s(x_{\text{test},j})^2 \right)$$

where $s(x_{\text{train},i})$ and $s(x_{\text{test},i})$ are the anomaly scores for the i-th training sample and the j-th test sample, respectively. And n_{train} and n_{test} are the number of samples in the training and test sets, respectively.

In SWIFT, the anomaly scores $s(x)$ are all negative values, as they are calculated based on the negative average path lengths in the Isolation Forest. Squaring these negative anomaly scores transforms them into positive values, but the magnitude of these squared values reflects the original anomaly scores. Normal data points have more negative anomaly scores due to their longer path lengths, so when squared, these scores become larger positive numbers. Anomalous data points have less negative anomaly scores because of their shorter path lengths,

resulting in smaller positive numbers when squared. This means that the mean squared anomaly score of the training data will be higher than that of the test data, which may contain anomalies with less negative scores.
By incorporating the squared anomaly scores into the loss function, $L(\mathbf{w})$ effectively captures the contrast between normal and anomalous data. Minimizing $L(\mathbf{w})$, which involves maximizing the difference between the mean squared anomaly scores of the training and test sets, encourages the model to adjust the tree weights $\mathbf{w}$ to enhance this separation. Specifically, the optimization process increases the weights of trees that contribute to assigning more negative anomaly scores to normal data and less negative anomaly scores to anomalies. By adjusting the weights in this manner, SWIFT amplifies the distinction between normal and anomalous observations, improving its effectiveness in detecting anomalies in the test data.

3.1.4 Weight transformation methods

To keep the optimized weights within a desirable range and to ensure numerical stability during optimization, raw weights w_i are transformed using a function with adjustable parameters. The transformed weights w_i' are computed as:

$$w_i' = \beta + (1 - \beta) \times T_i(\alpha \times \mathbf{w})$$

where:

- w_i represents the raw weight for the i-th tree.
- $\alpha > 0$ is a scaling parameter that controls the steepness or scaling of the function.
- $\beta \in [0, 1)$ sets a baseline for the weights to prevent them from becoming too small.
- T_i denotes the transformation function applied to the weights.

Two transformation functions were explored [2][10]:

1. Sigmoid Transformation:

$$T_i(\alpha \times \mathbf{w}) = \sigma(\alpha \times w_i) = \frac{1}{1 + e^{-\alpha w_i}}$$

   Here, the sigmoid function $\sigma(x)$ is applied individually to each scaled weight αw_i.

2. Softmax Transformation:
   The weights are transformed using the softmax function:

$$T_i(\alpha \times \mathbf{w}) = \text{softmax}_i(\alpha \times \mathbf{w}) = \frac{e^{\alpha w_i}}{\sum_{k=1}^{n_{\text{trees}}} e^{\alpha w_k}}$$

In this case, the softmax function is applied to the entire vector of scaled weights $\alpha \times \mathbf{w}$, and softmax_i denotes the i-th element of the resulting vector.

By iteratively updating the weights $\mathbf{w}$ to minimize the loss function $L(\mathbf{w})$, SWIFT dynamically adjusts the importance of each tree based on its effectiveness in differentiating between training and test data. This process enhances the model's ability to distinguish between normal and anomalous observations, especially in diverse and complex datasets.

3.1.5 Optimization Method for Weight Adjustment

The optimization of tree weights in SWIFT is a critical component that enhances the model's ability to distinguish between normal and anomalous data points during the test phase. This optimization process involves formulating a constrained optimization problem, where the objective is to find the set of weights $\mathbf{w} = [w_1, w_2, ..., w_{n_{\text{trees}}}]$ that minimize the custom loss function $L(\mathbf{w})$ defined earlier. The optimization problem is formally stated as:

$$\underset{\mathbf{w}}{\text{minimize}} \quad L(\mathbf{w})$$

To solve this problem efficiently and accurately, the Sequential Least Squares Programming (SLSQP) algorithm, as implemented in the SciPy library [27], is employed. The SLSQP method is particularly suitable for this optimization problem due to its ability to handle nonlinear, constrained optimization efficiently. It combines the merits of sequential quadratic programming with least squares methods, providing robustness and fast convergence for problems of this nature [15].

The SLSQP algorithm uses gradient information to update the weights during optimization. In SWIFT, the gradient of the loss function with respect to the weights provides crucial information about the direction in which the weights should be adjusted. While the specific gradient computations depend on the chosen transformation function, sigmoid or softmax, these computations ensure that the optimization remains stable and accurate.

By applying this optimization method, SWIFT dynamically adjusts the importance of each tree based on its contribution to differentiating between training and test data. The SLSQP algorithm enables efficient convergence to a set of optimized weights that enhance the model's anomaly detection capabilities.

Through this approach, SWIFT leverages the strengths of SLSQP to fine-tune the ensemble of isolation trees during the test phase, emphasizing trees that are more effective in isolating anomalies and improving the overall performance of the anomaly detection system.

3.2 k-Nearest Neighbors and KATANA

3.2.1 k-Nearest Neighbors

The k-Nearest Neighbors (kNN) algorithm is a non-parametric method commonly used for classification and regression tasks. In the context of anomaly detection, kNN can be adapted to identify anomalies based on the distances of data points to their nearest neighbors. The fundamental principle is that normal data points reside in dense regions of the feature space, having shorter distances to their neighbors, whereas anomalies are located in sparse regions, exhibiting larger distances to their nearest neighbors.
In standard kNN for anomaly detection, the algorithm operates as follows [11]:

1. Distance Computation: For each data point x in the dataset, compute the distances to all other data points using a chosen distance metric.

2. Neighbor Selection: Identify the k nearest neighbors of each data point x based on the computed distances.

3. Anomaly Score Calculation: The anomaly score for x is determined by a function of the distances to its k nearest neighbors. The score is set as the distance to the k-th nearest neighbor:
$$s(x) = d_k(x)$$
where $d_k(x)$ is the distance from x to the k-th nearest neighbor.

A higher anomaly score indicates that the data point is farther from its neighbors, suggesting it is an anomaly. Conversely, a lower score implies the data point is close to other points and is likely normal.
In standard kNN for anomaly detection, all features contribute equally to the distance computation. The Euclidean distance is commonly used, calculated as [6]:

$$d(x, y) = \sqrt{\sum_{i=1}^{n_{\text{features}}} (x_i - y_i)^2}$$

where x and y are data points, and n_{features} is the number of features.

However, in datasets where certain features are more informative for distinguishing anomalies from normal data, assigning equal importance to all features may not yield optimal results. To address this, feature weighting can be introduced to adjust the contribution of each feature in the distance computation.

3.2.2 Weighted k-Nearest Neighbors with Random Search Optimization

To enhance the performance of kNN in anomaly detection, a weighted distance metric is employed, where each feature is assigned a weight reflecting its importance [29]. The weighted Euclidean distance between two data points x and y is defined as:

$$d_{\text{weighted}}(x, y) = \sqrt{\sum_{i=1}^{n_{\text{features}}} w_i (x_i - y_i)^2}$$

where $w_i \geqslant 0$ is the weight assigned to the i-th feature.

The challenge lies in determining the optimal set of weights $\mathbf{w} = [w_1, w_2, \ldots, w_{n_{\text{features}}}]$ that improve the anomaly detection performance. Unlike the approach taken with SWIFT for Isolation Forest, where an optimization algorithm was used to adjust tree weights, applying a similar optimization method to kNN did not yield satisfactory results. Therefore, an alternative strategy based on random search was employed.

KATANA (kNN with Adaptive Test-time Adjustment through Nonuniform Adaptation) incorporates feature weighting and uses random search to optimize the weights during the test phase. Two variants of KATANA were developed:

- KATANA-AUC: This variant aims to find feature weights that maximize the ROC AUC score on the combined training and test data, using pseudo-labels that distinguish between training and test instances.

- KATANA-MD: This variant focuses on maximizing the difference in average anomaly scores between the test and training data.

In both variants, the random search optimization proceeds as follows:

1. Initialization: Start with an initial set of weights, setting $w_i = \frac{1}{n_{\text{features}}}$ for all features.

2. Random Perturbation: Iteratively perturb the current best weights by adding small random values drawn from a uniform distribution within a specified range.

3. Weight Normalization and Constraints: Ensure that the weights remain non-negative by clipping at a minimum value (e.g., 1×10^{-5}), and normalize them to sum to 1.

4. Anomaly Score Computation: Compute the weighted distances and anomaly scores for both the training and test data using the current weights.

5. Performance Evaluation and Weight Update:

- In KATANA-AUC, combine the anomaly scores from training and test data, assign 0 for training data and 1 for test data, and compute the ROC AUC score using these labels. If the new ROC AUC score improves over the previous best, update the best weights. This approach works because, as demonstrated in DOUST [14], maximizing the difference between the training and test data effectively maximizes the difference between normal and anomalous instances.
- In KATANA-MD, calculate the mean anomaly scores for the training data ($\bar{s}_{\text{train}}$) and the test data ($\bar{s}_{\text{test}}$). Compute the difference $\Delta = \bar{s}_{\text{test}} - \bar{s}_{\text{train}}$. If Δ is greater than the previous best, update the best weights.

6. Iteration: Repeat steps 2 to 5 for a predefined number of iterations.
7. Final Evaluation: After optimization, the best weights are used to compute anomaly scores on the test data, and the final ROC AUC score is calculated using the true labels of the test data.

KATANA employs test-time training principles by using the test data to inform the adjustment of feature weights, without accessing the true labels of the test data. By optimizing the weights based on performance metrics that reflect the separation between training and test data, the algorithm leverages the unlabeled test data to enhance anomaly detection.
In KATANA-AUC, the use of pseudo-labels (0 for training data, 1 for test data) allows the algorithm to maximize the ROC AUC score, effectively increasing the distinction between the distributions of training and test data in terms of anomaly scores. This approach assumes that anomalies are more prevalent in the test data.
In KATANA-MD, the algorithm seeks to maximize the mean difference in anomaly scores between test and training data, under the assumption that anomalies in the test set will lead to higher average anomaly scores compared to the training set.

3.3 One-Class Support Vector Machine and COSMOS

3.3.1 One-Class Support Vector Machine

The One-Class Support Vector Machine (OCSVM) is a kernel-based unsupervised learning algorithm commonly used for anomaly detection [24]. It models the training data by finding a hyperplane that best separates the data from the origin in a high-dimensional feature space induced by a kernel function. The fundamental idea is to capture regions in the input space where the data is dense, considering new data points outside these regions as anomalies.

In OCSVM, the algorithm operates as follows [3]:

1. Training Phase: Given a set of training data x_i, the OCSVM solves the following optimization problem:

$$\begin{aligned} \underset{\mathbf{w},\rho,\xi_i}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho \\ \text{subject to} \quad & (\mathbf{w}\cdot\phi(x_i)) \geqslant \rho - \xi_i, \quad \xi_i \geqslant 0, \quad i = 1,\ldots,n \end{aligned}$$

   where:

   - $\phi(\cdot)$ is the feature mapping induced by the kernel function.
   - $\nu \in (0,1]$ is a parameter controlling the trade-off between the fraction of outliers and the margin of the hyperplane.
   - ξ_i are slack variables that allow for violations of the margin constraints, accommodating outliers or noise.
   - ρ is the offset.

2. Decision Function: After solving the optimization problem, the decision function for a new data point x is given by:

$$f(x) = \sum_{i=1}^{n}\alpha_i K(x_i, x) - \rho$$

   where:

   - $K(x_i, x)$ is the kernel function.
   - α_i are the dual variables obtained from the dual form of the optimization problem, representing the influence of each training point.

3. Anomaly Score Calculation: The anomaly score for a data point x is defined as the negative of the decision function value:

$$s(x) = -f(x)$$

Data points with higher anomaly scores are considered more likely to be anomalies, as they lie further from the region where the training data is concentrated.

In practice, the Radial Basis Function (RBF) kernel is often used due to its ability to capture non-linear relationships [5]:

$$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right)$$

where $\gamma > 0$ is a parameter defining the kernel width, controlling the influence of individual data points on the decision boundary.

3.3.2 Weighted One-Class SVM with Random Search Optimization

To enhance the performance of OCSVM in anomaly detection, a weighted kernel function is introduced, assigning a weight to each feature to reflect its importance. The weighted RBF kernel is defined as:

$$K_{\text{weighted}}(x_i, x_j) = \exp\left(-\gamma \sum_{k=1}^{n_{\text{features}}} w_k (x_{i,k} - x_{j,k})^2\right)$$

where $w_k \geqslant 0$ is the weight assigned to the k-th feature. This weighting adjusts the influence of each feature in the kernel computation, allowing the model to emphasize more informative features and reduce the impact of less relevant ones.

Determining the optimal set of weights $\mathbf{w} = [w_1, w_2, \ldots, w_{n_{\text{features}}}]$ that improve anomaly detection performance is challenging. Similar to the approach used in KATANA for kNN, COSMOS (Continuous One-class Support Vector Machine Optimization through test-time weight Sampling) introduces kernel weighting and employs test-time training principles to enhance anomaly detection, with two variants: COSMOS-AUC, which maximizes the ROC AUC score, and COSMOS-MD, which maximizes the mean anomaly score difference between training and test data.

The optimization process is the same as described for KATANA in the kNN algorithm as described in Section 3.2.2. The key difference lies in the application of feature weighting within the kernel function for OCSVM, as opposed to the distance metric in kNN.

3.4 Performance Evaluation and Statistical Analysis

3.4.1 Receiver Operating Characteristic Area Under the Curve (ROC AUC)

The Receiver Operating Characteristic Area Under the Curve (ROC AUC) is a widely used performance metric for evaluating binary classification models, particularly in anomaly detection tasks where class imbalance is common. ROC AUC measures the model's ability to distinguish between positive and negative classes across all possible threshold settings, providing a comprehensive assessment of its discriminative power.

The ROC curve is a plot of the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold levels [23]. The TPR, also known as sensitivity or recall, is defined as:

$$\mathrm{TPR} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

The FPR is calculated as:

$$\mathrm{FPR} = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}}$$

The area under the ROC curve (AUC) quantifies the overall performance of the model, with a value ranging from 0.5 (no discriminative ability) to 1.0 (perfect discrimination). A higher ROC AUC indicates that the model is better at ranking positive instances (anomalies) higher than negative instances (normal data) [7].

In this study, ROC AUC was chosen as the primary evaluation metric for assessing the performance of SWIFT, KATANA, and the custom One-Class SVM algorithm. This choice is supported by findings from our recent research, which demonstrated that ROC AUC remains consistent across various configurations, including different levels of contamination in the training data and varying outlier fractions in the test set [21]. This consistency makes ROC AUC a reliable metric for comparing anomaly detection models under diverse conditions.

Applying ROC AUC to our models involves computing anomaly scores for the test data and evaluating how well these scores differentiate between normal instances and anomalies. For SWIFT, anomaly scores are based on weighted path lengths in the Isolation Forest; for KATANA, scores are derived from weighted distances to the k-th nearest neighbor; and for the custom One-Class SVM, scores come from the negative decision function values using the weighted kernel.

By calculating the ROC AUC for each model, we obtain a threshold-independent measure of performance that reflects the models' abilities to rank anomalies higher than normal instances. This is particularly important in anomaly detection, where selecting an appropriate threshold can be challenging due to class imbalance and the rarity of anomalies. ROC AUC allows us to compare models without relying on a specific threshold, ensuring a fair and comprehensive evaluation.

In summary, ROC AUC serves as an effective and consistent metric for evaluating and comparing the performance of anomaly detection models like SWIFT, KATANA, and the custom One-Class SVM. Its use in this study is justified by its robustness across different data configurations and its ability to provide a holistic view of model performance in distinguishing between normal and anomalous data.

3.4.2 Wilcoxon Signed-Rank Test with Holm's Correction

In order to rigorously evaluate the statistical significance of the performance differences between the proposed models and standard anomaly detection algorithms, we used the Wilcoxon Signed-Rank Test, a non-parametric test suitable for paired data comparisons. This test assesses whether the median of the differences between paired observations is zero, making it appropriate for comparing model performances across datasets. Additionally, Holm's correction was applied to adjust for multiple comparisons and control the family-wise error rate (FWER).

The Wilcoxon Signed-Rank Test operates as follows [30]:

1. For paired observations (x_i, y_i), calculate the differences:

$$d_i = x_i - y_i$$

2. Exclude Zero Differences:
   Remove any pairs where $d_i = 0$, reducing the sample size to n'.

3. Rank Absolute Differences: Assign ranks R_i to the absolute differences $|d_i|$, with ties receiving averaged ranks.

4. Assign Signs to Ranks: Multiply each rank by the sign of d_i:

$$S_i = \text{sign}(d_i) \times R_i.$$

5. Calculate Test Statistic: Calculate the sum of positive ranks (W^+) and negative ranks (W^-). The test statistic T is the smaller of the two:

$$T = \min(W^+, W^-).$$

6. Determine Statistical Significance:

- For small sample sizes, use Wilcoxon Signed-Rank tables to find the critical value corresponding to the significance level α and sample size n'.
- For larger samples ($n' > 20$), approximate the distribution of T using a normal distribution:

$$Z = \frac{T - \frac{n'(n'+1)}{4}}{\sqrt{\frac{n'(n'+1)(2n'+1)}{24}}}$$

  where Z is compared to a standard normal distribution to calculate the p-value.

When conducting multiple statistical tests, the chance of obtaining a Type I error (false positive) increases. To account for multiple comparisons and control the FWER, Holm's correction was applied to the p-values obtained from the Wilcoxon tests. Holm's correction adjusts significance thresholds in a stepwise manner [13]:

1. Order the p-values:
   Arrange the m individual p-values from the Wilcoxon tests in ascending order:

$$p_{(1)} \leqslant p_{(2)} \leqslant \cdots \leqslant p_{(m)}$$

2. Adjust Significance Levels:
   For each ordered p-value $p_{(i)}$, calculate the adjusted significance level α_i:

$$\alpha_i = \frac{\alpha}{m - i + 1}$$

3. Sequential Testing: Begin with the smallest p-value $p_{(1)}$. Reject the null hypothesis for $p_{(i)}$ if $p_{(i)} \leqslant \alpha_i$. Stop testing as soon as a p-value exceeds its corresponding threshold.

Holm's method controls the FWER at the desired level α, offering more statistical power than the traditional Bonferroni correction by making adjustments based on the ordered p-values.

In this study, the Wilcoxon Signed-Rank Test with Holm's correction is applied as follows:

- Paired Observations:
  ROC AUC scores for each dataset were paired to evaluate performance differences. Comparisons were made between all proposed models (SWIFT, KATANA, and COSMOS), standard models, and DOUST.

- Statistical Testing:
  The Wilcoxon Signed-Rank Test was applied to assess performance differences for each pair of models across datasets, and p-values were calculated.

- Adjustment for Multiple Comparisons:
  Since multiple tests were performed comparing different models and across multiple datasets, Holm's correction was applied to the p-values to account for the multiple pairwise comparisons, ensuring that the overall FWER remained below the chosen significance level α.

By using these statistical methods, we ensured that our conclusions about the superiority of the proposed models were statistically valid and robust against false positives. The combination of the Wilcoxon Signed-Rank Test and Holm's correction provided a rigorous framework for comparing model performance while addressing the challenges of multiple hypothesis testing.

Chapter 4

Results

4.1 Strategically Weighted Isolation Forests at Test time (SWIFT)

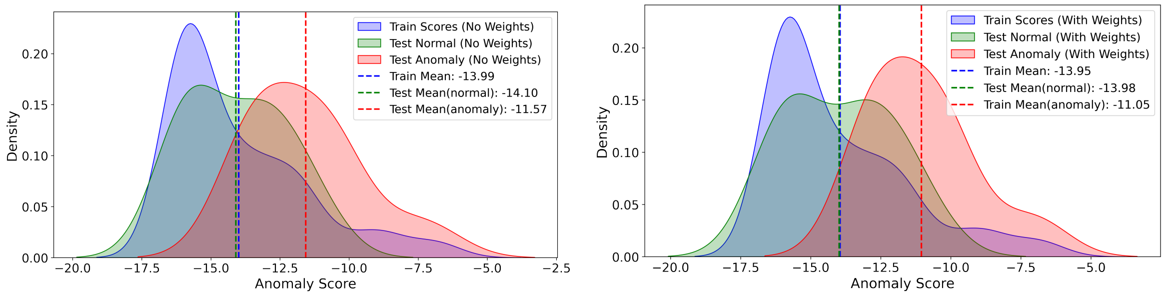
In this section, a comparative analysis is presented between Strategically Weighted Isolation Forests at Test time (SWIFT) and standard implementation of Isolation Forest (IF) from scikit-learn, focusing on the models' performance across various datasets. The results aim to highlight SWIFT's ability to outperform the traditional IF through its strategic weighting mechanism and specific configuration adjustments, such as experimenting with initial weights, applying different weight transformation functions, and tuning parameters to optimize performance.

Both SWIFT and the standard Isolation Forest were configured with 100 trees, a number chosen to optimize the trade-off between performance and computational efficiency. This is consistent with sklearn's default setting for the Isolation Forest [22]. The `max_samples` parameter was set to `auto`, allowing each tree to use up to 256 samples or the entire dataset if smaller. This setting ensures adequate sampling diversity without overwhelming computational resources.

4.1.1 Impact on Anomaly Score Distributions

To illustrate the impact of SWIFT's test-time weighting mechanism on anomaly detection, the `glass` dataset was analyzed as a representative example [12][8]. This dataset effectively demonstrates how SWIFT adapts model behavior and improves scoring metrics through strategic weight modifications.

Figure 4.1 illustrates the density distributions of anomaly scores derived from the **glass** dataset, providing a visual representation of how SWIFT's test-time weighting mechanism affects scoring behavior across both training and testing phases. The figure highlights the



(a) Without Weights: Training mean $= -13.99$, Normal test mean $= -14.10$, Anomaly test mean $= -11.57$

(b) With Weights: Training mean $= -13.95$, Normal test mean $= -13.98$, Anomaly test mean $= -11.05$

Figure 4.1: Comparison of Anomaly Score Distributions for the *glass* Dataset Using SWIFT: The plots illustrate how the test-time weighting mechanism affects the anomaly score distribution, highlighting improvements in distinguishing between normal and anomalous data in the training and testing phases.

differences in the anomaly score distributions between different model configurations, emphasizing the shifts observed in the training, normal test, and anomalous test scores. These shifts offer insights into how the model adapts to different phases and refines its scoring behavior when weights are applied.

After applying the modified weights in SWIFT, the mean values of the distributions exhibit stable behavior in both the training and normal test phases. Specifically, the training distribution's mean adjusts slightly from -13.99 to -13.95, while the normal test scores shift from -14.10 to -13.98. This minimal difference indicates that both distributions remain largely stable, with the normal test scores becoming slightly more similar to the training scores, suggesting an increased similarity between the two phases.

Notably, the mean of the anomalous scores in the test set shows a substantial shift from -11.57 to -11.05, indicating a significant divergence from both training and normal test means. This change highlights the model's enhanced capability to discriminate anomalies from normal observations, which is one of the critical features of robust anomaly detection systems. The increased separation of the anomaly mean, following the application of weights, is crucial for ensuring the model's effectiveness in accurately detecting and isolating anomalous behavior.

By starting with the **glass** dataset, the analysis demonstrates how SWIFT's adaptive mechanisms significantly improve anomaly score distributions. These findings are then extended across diverse datasets to confirm that the observed improvements are consistent and repeatable.

4.1.2 Evaluation Across Multiple Datasets

A diverse array of 47 datasets from the Anomaly Detection Benchmark (ADBench) was employed to evaluate the models [12]. These datasets span a diverse range of domains, including network security, healthcare, and financial fraud detection, ensuring the evaluation encompasses a broad spectrum of real-world anomaly detection scenarios.

Consistent preprocessing was applied across all datasets to maintain comparability. Numeric features were standardized using scikit-learn's **StandardScaler** to prevent any single feature from dominating due to differences in scale. A 50% outlier fraction was maintained in the test set, meaning that for each dataset, the number of normal data points equaled the number of outliers in the test set. For example, if there were 5 anomalies in a dataset, the test set comprised 5 anomalies and 5 normal points, with the remaining data allocated to the training set.

Following the configuration described in the methodology, SWIFT was rigorously evaluated across 47 different datasets, comparing its performance to that of the standard Isolation Forest. The evaluation focused on the Receiver Operating Characteristic Area Under the Curve (ROC AUC) metric, which measures the model's ability to distinguish between normal and anomalous instances across various threshold settings. The detailed results are presented in Table 4.1.

As observed in Table 4.1, SWIFT consistently outperforms the standard Isolation Forest across the majority of datasets. Out of 47 datasets, SWIFT achieved higher ROC AUC scores in 43 cases, with ties in 3 datasets (*WBC*, *Lymphography* and *WDBC*), where both models reached perfect scores in 2 datasets, and a slight underperformance in 1 dataset (*WPBC*). The average ROC AUC score across all datasets increased from 0.7916 for the Isolation Forest to 0.8161 for SWIFT, highlighting the overall improvement in anomaly detection capability.

To assess the statistical significance of the performance enhancements achieved by SWIFT, a Wilcoxon signed-rank test was conducted. This non-parametric test is appropriate for comparing two related samples—in this case, the ROC AUC scores from the same datasets processed by the two different models. The test resulted in a p-value of $p = 1.15 \times 10^{-8}$, which is significantly lower than the conventional significance level of 0.05. This exceptionally low p-value not only confirms the statistical significance of SWIFT's superior performance but also underscores the substantial impact of the adaptive weighting mechanism.

Table 4.1: Comparison of ROC AUC Scores Between Isolation Forest and SWIFT Across 47 Datasets: **Bold** values indicate superior performance for each dataset. The results demonstrate SWIFT's effectiveness across diverse datasets, reflecting its enhanced anomaly detection capabilities through test-time training. The average ROC AUC score increases from **0.7916** (Isolation Forest) to **0.8161** (SWIFT).

Filename	Ifor	SWIFT
annthyroid	0.9168	**0.9379**
breastw	0.9975	**0.9977**
cardio	0.9362	**0.9491**
Cardiotocography	0.8258	**0.8585**
glass	0.8148	**0.8642**
Hepatitis	0.7929	**0.8166**
InternetAds	0.4357	**0.4759**
Ionosphere	0.8644	**0.8774**
letter	0.6103	**0.6440**
Lymphography	**1.0000**	**1.0000**
mammography	0.8931	**0.9003**
PageBlocks	0.9273	**0.9312**
Pima	0.7368	**0.7420**
satellite	0.8052	**0.8197**
satimage-2	0.9956	**0.9964**
SpamBase	0.8235	**0.8482**
Stamps	0.9147	**0.9230**
thyroid	0.9894	**0.9920**
vertebral	0.3900	**0.4456**
vowels	0.7816	**0.8200**
Waveform	0.7718	**0.8104**
WBC	**0.9900**	**0.9900**
WDBC	**1.0000**	**1.0000**
Wilt	0.4854	**0.5523**

Filename	Ifor	SWIFT
wine	0.8700	**0.8900**
WPBC	**0.5781**	0.5745
yeast	0.3953	**0.4138**
optdigits	0.6372	**0.7490**
fault	0.6519	**0.6735**
mnist	0.7962	**0.8571**
musk	0.9716	**0.9985**
shuttle	0.9968	**0.9974**
landsat	0.5996	**0.6396**
pendigits	0.9579	**0.9670**
smtp	0.9122	**0.9200**
skin	0.8861	**0.8984**
campaign	0.6855	**0.7026**
magic.gamma	0.8062	**0.8211**
celeba	0.6975	**0.7430**
donors	0.9484	**0.9725**
ALOI	0.5568	**0.5638**
cover	0.8138	**0.8897**
http	0.9888	**0.9937**
speech	0.4859	**0.4985**
backdoor	0.8707	**0.8944**
fraud	0.9551	**0.9565**
census	0.4432	**0.5480**
average	0.7916	**0.8161**

4.1.3 Impact of Random Initial weights

Building upon the initial success of SWIFT with uniform initial weights, the impact of using random initial weights in the weighting mechanism was further explored. In the standard SWIFT implementation, the initial weights $\mathbf{w}$ are set uniformly to 1. However, initializing weights randomly can potentially lead to different optimization trajectories, possibly enhancing the model's performance by allowing the optimizer to explore a broader solution space. Due to time constraints, the effect of random initial weights was examined on a subset of 8 datasets.

Table 4.2: Comparison of ROC AUC Scores for Isolation Forest, SWIFT with Uniform Initial Weights, and SWIFT with Random Initial Weights (sampled from $\text{Uniform}(0.1, 1)$): **Bold** values indicate highest performance for each dataset. The results show that SWIFT with random initial weights provide slight improvements on several datasets, suggesting potential benefits in optimization.

Filename	Isolation forest	SWIFT	SWIFT (Random Init)
annthyroid	0.9168	0.9379	**0.9380**
breastw	0.9975	**0.9977**	**0.9977**
cardio	0.9362	**0.9491**	**0.9491**
Cardiotocography	0.8258	**0.8585**	**0.8585**
glass	0.8148	**0.8642**	**0.8642**
Hepatitis	0.7929	**0.8166**	**0.8166**
InternetAds	0.4357	**0.4759**	**0.4759**
Ionosphere	0.8644	0.8774	**0.8775**

To investigate this, the initial weights were modified by sampling from a uniform distribution within a specified range, specifically between 0.1 and 1:

$$w_i^{(0)} \sim \text{Uniform}(0.1, 1)$$

The same optimization process as before was conducted, adjusting the weights to minimize the loss function $L(\mathbf{w})$. The results of this experiment are summarized in Table 4.2, which includes the ROC AUC scores for SWIFT with random initial weights, denoted as *SWIFT (Random Init)*.

As shown in Table 4.2, SWIFT with random initial weights achieved slightly higher ROC AUC scores on several datasets compared to SWIFT with uniform initial weights. Although the improvements are minor, this suggests that random initialization of weights can provide a small performance boost, likely due to the optimizer escaping local minima that may be associated with uniform starting points.

Given these observations, and considering that random initial weights do not negatively impact the model's performance, it was decided to proceed with random weights in subsequent experiments. This choice allows for the possibility of enhanced optimization outcomes without incurring significant additional computational costs. Continuing with random initial weights aims to capitalize on any potential performance gains across different datasets, acknowledging that even minor improvements can be valuable in critical anomaly detection applications.

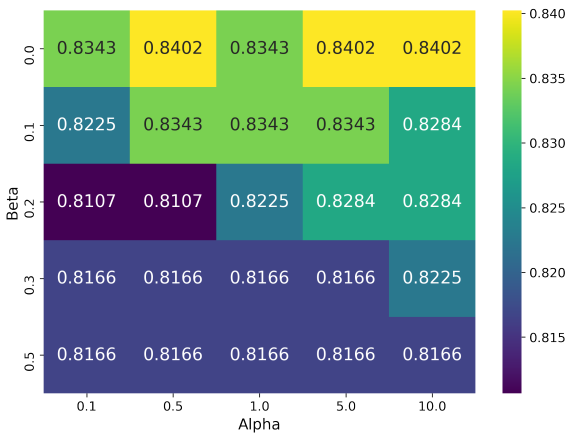
4.1.4 Effect of Weight Transformation Functions on Model Performance

The exploration of different weight transformation functions aimed to enhance SWIFT's performance by investigating how modifying the weight transformations impacts the model's ability to distinguish between normal and anomalous data points. Specifically, the sigmoid and softmax transformations were compared to determine their effectiveness in improving anomaly detection performance. Both transformations have merit: the sigmoid transformation scales each weight individually, potentially allowing for fine-grained adjustments, while the softmax transformation considers the entire set of weights collectively, accentuating relative differences among them.

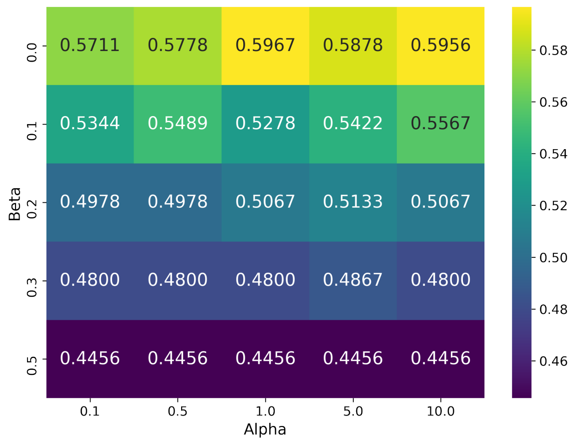
Experiments were conducted on two datasets, `Hepatitis` and `vertebral`, to evaluate SWIFT under different weight transformation configurations. The parameters α and β were varied within specified ranges ($\alpha \in \{0.1, 0.5, 1, 5, 10\}$ and $\beta \in \{0, 0.1, 0.2, 0.3, 0.5\}$), and the weights were used in their raw form and after transformed and normalization. The anomaly scores for each sample were calculated using these weights, as described in the methodology section.

The heatmaps in Figure 4.2 illustrate the ROC AUC scores achieved under different configurations of α and β for the softmax transformation with raw weights and the sigmoid transformation with transformed and normalized weights on the `Hepatitis` and `vertebral` datasets. Other configurations, such as softmax with transformed and normalized weights and sigmoid with raw weights, are not presented here because they yielded relatively lower ROC AUC scores compared to the configurations shown. The corresponding plots are provided in the Appendix for reference (see Figure 1).

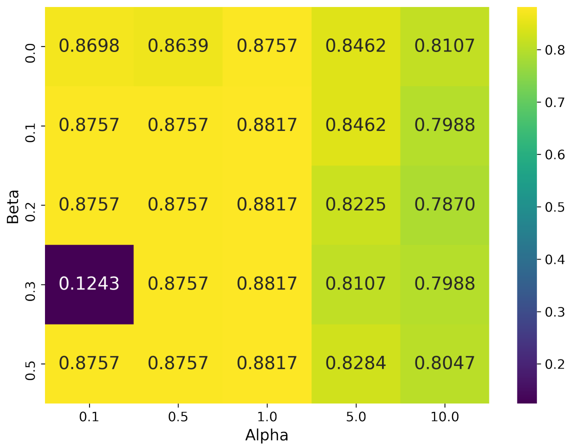
These visualizations reveal that the softmax transformation with raw weights consistently outperforms the sigmoid transformation with normalized weights across various parameter settings. In the softmax configuration, the ROC AUC scores remain high for $\alpha \leqslant 1$ and $\beta \leqslant 0.2$. For instance, on the `vertebral` dataset, the ROC AUC scores achieved with the softmax transformation are noticeably higher than those obtained with the sigmoid transformation. In contrast, the sigmoid transformation exhibits lower and more variable ROC AUC scores, indicating less consistent performance.



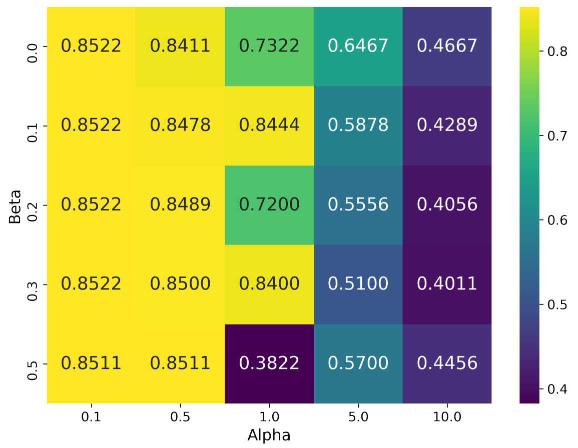
(a) `Hepatitis` Sigmoid Transformation with transformed and normalized weights



(b) `vertebral` Sigmoid Transformation with transformed and normalized weights



(c) `Hepatitis` Softmax Transformation with raw weights



(d) `vertebral` Softmax Transformation with raw weights

Figure 4.2: ROC AUC Scores for Different Weight Transformations on the `Hepatitis` and `vertebral` Datasets

An intriguing observation was that using raw transformed weights led to higher ROC AUC scores than using normalized weights, especially in the softmax configuration. This finding is somewhat surprising because the ROC AUC metric is theoretically scale-invariant, meaning that scaling the anomaly scores should not affect the performance. However, in this context, scaling and normalization are applied to the weights before computing the anomaly scores, not directly to the scores themselves.

The anomaly score for a sample x is calculated as:

$$s(x) = -\sum_{k=1}^{n_{\text{trees}}} w'_k h_k(x)$$

, where w'_i are the transformed weights and $h_k(x)$ are the path lengths from the isolation trees. The interaction between the weights and the path lengths means that scaling or normalizing the weights affects the anomaly scores in a non-monotonic manner. This alteration can change the distribution and ordering of the anomaly scores, thereby impacting the ROC AUC. While ROC AUC is scale-invariant when applying monotonic transformations directly to the anomaly scores $s(x)$, scaling the weights w'_i affects $s(x)$ in a way that is not equivalent to applying a monotonic function to $s(x)$. Therefore, scaling or normalizing the weights can impact the ROC AUC.

One possible explanation for the observed differences is that using different weight transformation functions, such as sigmoid or softmax, leads to different optimal weights because these transformations alter the optimization landscape. The sigmoid function scales weights individually and can cause gradient saturation, where extreme weight values result in very small gradients, potentially hindering the optimizer's progress toward the global minimum. This may cause the optimizer to converge to suboptimal weights that do not sufficiently emphasize the most significant trees. In contrast, the softmax function considers all weights collectively, accentuating relative differences and providing more pronounced gradients that facilitate more effective optimization. These differences influence how the optimizer explores the solution space and can lead to different sets of optimal weights. Consequently, using different transformations can result in variations in the model's performance, as the optimizer converges to different weights under each transformation.

The comparison between the sigmoid and softmax transformations was crucial in understanding how different weight transformations affect SWIFT's performance. The sigmoid function scales each weight individually based on its value, which may not sufficiently highlight the relative importance of each tree in the ensemble. In contrast, the softmax function considers the entire set of weights, accentuating differences and creating a probability distribution that inherently emphasizes trees with higher weights. This collective scaling appears to enhance the model's ability to detect anomalies by leveraging the most significant trees more effectively.

Based on these observations, the softmax transformation with raw weights with $\alpha = 1$ and $\beta = 0.1$ was selected for further experiments. This parameter setting falls within the ranges where consistent and high ROC AUC scores were observed ($\alpha \leqslant 1$ and $\beta \leqslant 0.2$). Although other parameter settings occasionally produced comparable scores on specific datasets, this configuration provided a good balance between performance and stability.

With this configuration, SWIFT will be implemented on all the datasets used in previous experiments. This approach allows for a comprehensive assessment of SWIFT's capabilities using a weight transformation that effectively emphasizes the most informative trees without disproportionately diminishing the contributions of others.

4.1.5 SWIFT Performance with Softmax Transformation

To assess the impact of the softmax weight transformation on SWIFT's anomaly detection performance, this configuration was evaluated across all datasets used in the previous study. The objective was to determine whether the enhancements observed in initial experiments could be generalized and to explore how the choice of weight transformation and the tuning of parameters α and β contribute to improving SWIFT's effectiveness.

Three models were compared:

- Isolation Forest: The standard algorithm without any weighting scheme.
- SWIFT: The original implementation using the sigmoid transformation with $\alpha = 1$ and $\beta = 0.5$.
- SWIFT with softmax transformation: Employting the softmax transformation with $\alpha = 1$ and $\beta = 0.1$.

Table 4.3 provides the ROC AUC scores for each model across all datasets. The results indicate that SWIFT with the softmax transformation achieved the highest ROC AUC scores in the majority of datasets. The average ROC AUC notably increased to 0.8947, compared to 0.8161 for the original SWIFT and 0.7916 for Isolation Forest. This suggests that the softmax transformation can enhance the model's ability to detect anomalies by effectively emphasizing the most informative trees in the ensemble.
However, for certain datasets such as `breastw`, `Lymphography`, and `WBC`, the performance of SWIFT with the softmax transformation was lower than that of the other models. In these cases, the original SWIFT or Isolation Forest achieved higher ROC AUC scores. This suggests that while the softmax transformation generally improves performance, it may not be universally optimal for all datasets.
The observed improvements can be attributed to the exploration of weight transformations and the tuning of parameters α and β. By adjusting these parameters, the model can better emphasize the most informative trees in the ensemble, leading to enhanced detection of anomalies. This indicates the potential for other configurations and transformations that might further enhance SWIFT's performance, and that the chosen softmax transformation

with $\alpha = 1$ and $\beta = 0.1$ is one effective choice among potentially many.

Table 4.3: ROC AUC Scores for Isolation Forest, SWIFT with Sigmoid Transformation, and SWIFT with Softmax Transformation ($\alpha = 1$, $\beta = 0.1$) Across 47 Datasets: **Bold** values indicate the highest performance for each dataset. The results demonstrate that SWIFT with softmax transformation achieves an impressive improvement in anomaly detection performance, with an average ROC AUC score of **0.8947**, compared to **0.8161** for SWIFT with sigmoid transformation and **0.7916** for Isolation Forest.

Filename	Isolation forest	SWIFT	SWIFT with Softmax (alpha=1, beta=0.1)
annthyroid	0.9168	0.9379	**0.9944**
breastw	0.9975	**0.9977**	0.9822
cardio	0.9362	0.9491	**0.9883**
Cardiotocography	0.8258	0.8585	**0.9443**
glass	0.8148	0.8642	**1.0000**
Hepatitis	0.7929	0.8166	**0.8817**
InternetAds	0.4357	0.4759	**0.6792**
Ionosphere	0.8644	0.8774	**0.9465**
letter	0.6103	0.6440	**0.7594**
Lymphography	**1.0000**	**1.0000**	0.9444
mammography	0.8931	0.9003	**0.9211**
PageBlocks	0.9273	0.9312	**0.9484**
Pima	0.7368	0.7420	**0.7785**
satellite	0.8052	0.8197	**0.8837**
satimage-2	0.9956	**0.9964**	0.9923
SpamBase	0.8235	0.8482	**0.9334**
Stamps	0.9147	0.9230	**0.9844**
thyroid	0.9894	**0.9920**	0.9917
vertebral	0.3900	0.4456	**0.8444**
vowels	0.7816	0.8200	**0.9100**
Waveform	0.7718	0.8104	**0.9107**
WBC	**0.9900**	**0.9900**	0.8300
WDBC	**1.0000**	**1.0000**	**1.0000**
Wilt	0.4854	0.5523	**0.9067**
wine	0.8700	0.8900	**1.0000**

Continued on next page

Filename	Isolation Forest	SWIFT	SWIFT with Softmax (alpha=1, beta=0.1)
WPBC	0.5781	0.5745	**0.5957**
yeast	0.3953	0.4138	**0.6835**
optdigits	0.6372	0.7490	**0.9915**
fault	0.6519	0.6735	**0.7716**
mnist	0.7962	0.8571	**0.9421**
musk	0.9716	0.9985	**1.0000**
shuttle	0.9968	0.9974	**0.9967**
landsat	0.5996	0.6396	**0.8141**
pendigits	0.9579	0.9670	**0.9894**
smtp	0.9122	0.9200	**0.9722**
skin	0.8861	0.8984	**0.9826**
campaign	0.6855	0.7026	**0.7641**
magic.gamma	0.8062	**0.8211**	0.8078
celeba	0.6975	0.7430	**0.9078**
donors	0.9484	0.9725	**0.9997**
ALOI	0.5568	0.5638	**0.5985**
cover	0.8138	0.8897	**0.9986**
http	0.9888	0.9937	**0.9998**
speech	0.4859	0.4985	**0.6006**
backdoor	0.8707	0.8944	**0.9201**
fraud	0.9551	**0.9565**	0.9401
census	0.4432	0.5480	**0.8215**
average	0.7916	0.8161	**0.8947**

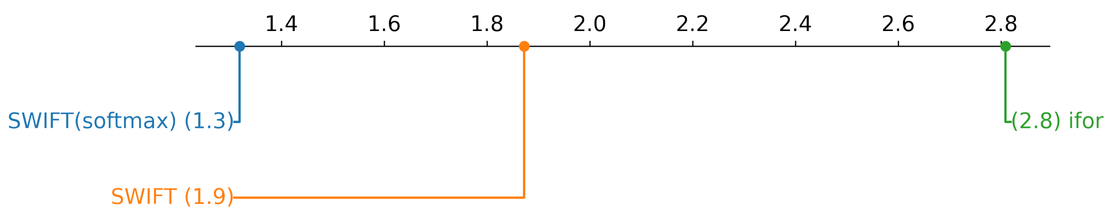


Figure 4.3: Comparison of Methods by Average Rank with Statistical Significance: The figure illustrates the average ranks of Isolation Forest, SWIFT with sigmoid transformation, and SWIFT with softmax transformation, based on ROC AUC performance. Lower ranks indicate better performance. The Wilcoxon signed-rank test with Holm's correction confirms that all pairwise comparisons are statistically significant, with no connecting lines between methods.

To statistically validate these performance differences among the models, the Wilcoxon signed-rank test with Holm's correction was applied. The comparison of average ranks reveals that SWIFT with the softmax transformation achieves the highest performance, with an average rank of 1.319. This is followed by the standard SWIFT model, which holds an average rank of 1.872. The Isolation Forest algorithm, by contrast, has the lowest ranking among the three methods, with an average rank of 2.809. Figure 4.3 visualizes this comparison of methods by average rank, along with the statistical significance of their differences [26]. In this plot, no lines are shown between methods because all pairwise comparisons are statistically significant.

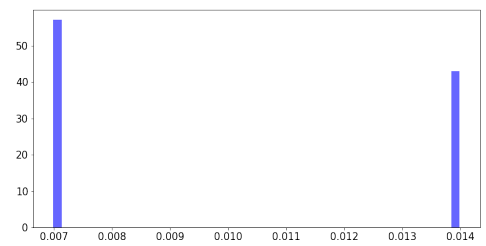
The Wilcoxon test results, corrected for multiple comparisons using Holm's method, confirm the significant differences between models. Raw p-values for Isolation Forest versus SWIFT, Isolation Forest versus SWIFT with softmax, and SWIFT versus SWIFT with softmax are 1.15×10^{-8}, 3.08×10^{-7}, and 6.47×10^{-7}, respectively. After Holm's correction, the p-values remain highly significant at 3.46×10^{-8}, 6.16×10^{-7}, and 6.47×10^{-7}. These values support the effectiveness of the softmax-transformed SWIFT, as indicated by the red lines in Figure 4.3, reinforcing the findings from the average rank analysis.

In summary, SWIFT with the softmax transformation substantially improves anomaly detection performance across a wide range of datasets. The enhancement is achieved by exploring the weight transformation functions and tuning the parameters α and β. While the chosen configuration of the softmax transformation $\alpha = 1$ and $\beta = 0.1$ yielded significant improvements, it represents one effective option among others. Further exploration of different transformations and parameter settings may uncover additional ways to enhance SWIFT's performance.

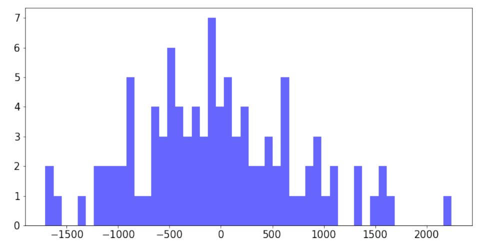
4.1.6 Analysing the Weights in Different Configurations

SWIFT was evaluated with various configurations, including the use of softmax and sigmoid transformation functions, as well as applying raw weights versus transformed and normalized weights. An interesting phenomenon was observed in some datasets, particularly the `vertebral` dataset, where significant variations in performance emerged depending on the weight configuration employed.
On the `vertebral` dataset, the baseline performance using the standard Isolation Forest without any weighting yielded a ROC AUC score of 0.39. When applying the sigmoid transformation with raw weights, the model's performance decreased, resulting in a ROC AUC of 0.1533. However, when the weights were transformed and normalized using the sigmoid function, there was a slight improvement over the baseline, with a ROC AUC of 0.4456.

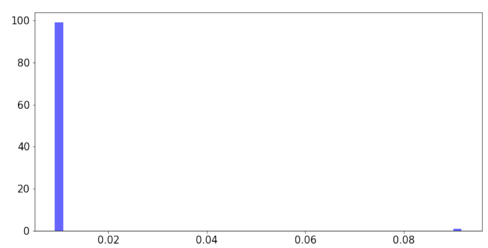
A particularly noteworthy outcome was observed when using the softmax transformation with raw weights. In this configuration, the model achieved a significantly higher ROC AUC score of 0.8444, indicating a substantial enhancement in anomaly detection capability. In contrast, when the softmax-transformed and normalized weights were used, the ROC AUC decreased to 0.4256, which is close to the baseline performance.
To understand these outcomes, the distribution of the optimized weights obtained from each configuration was analyzed. Figure 4.4 presents histograms of the weights for the different configurations. In the softmax transformation with raw weights, the optimized weights exhibited a diverse distribution, including both positive and negative values. The presence of significant negative weights played a crucial role in enhancing the model's performance. Since the anomaly score in SWIFT is calculated as the negative weighted sum of the path lengths, negative weights effectively inverted the influence of certain trees. This inversion adjusted the anomaly scores in a way that improved the model's ability to distinguish between normal and anomalous instances, leading to a higher ROC AUC.



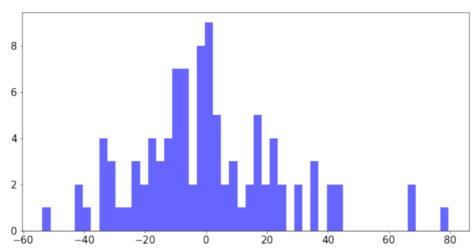
(a) Sigmoid Transformation with transformed and normalized weights



(b) Sigmoid Transformation with raw weights



(c) Softmax Transformation with transformed and normalized weights



(d) Softmax Transformation with raw weights

Figure 4.4: Histograms of Weights of `vertebral` Dataset in Different Configurations

Interestingly, the ROC AUC of 0.1533 obtained with the sigmoid transformation and raw weights is approximately equal to 1 minus the ROC AUC of 0.8444 from the softmax transformation with raw weights (i.e.,$0.1533 \approx 1 - 0.8444$). This suggests that in the sigmoid with raw weights configuration, the model's predictions were effectively inverted compared to the softmax with raw weights configuration. In other words, the model was assigning higher anomaly scores to normal instances and lower scores to anomalies, which drastically reduced performance.
Similar trends were observed in the `wine` dataset. The standard Isolation Forest achieved a baseline ROC AUC of 0.87. Applying the sigmoid transformation with raw weights yielded 0.0, indicating inverted predictions, while using transformed and normalized weights improved the score to 0.89. The best result came from the softmax transformation with raw weights, reaching a perfect 1.0, whereas using softmax-transformed and normalized weights slightly reduced the score to 0.88.

Further analysis revealed how the transformed and normalized weights affected the model's performance. In the case of the sigmoid transformation with transformed and normalized weights, the optimized weights predominantly took on two distinct values, effectively creating a binary weighting scheme. This allowed the model to emphasize certain trees over others, resulting in a slight improvement over the baseline with a ROC AUC of 0.4456. The presence of two distinct weight values enabled the model to leverage the contributions of specific trees more effectively, enhancing its ability to detect anomalies compared to using uniform weights. In contrast, the softmax transformation with transformed and normalized weights produced a weight distribution where only one tree received a significantly higher weight, while the rest had similar, much lower weights. This configuration led to a ROC AUC of 0.4256, close to the baseline performance. Although the high weight assigned to a single tree slightly influenced the anomaly scores, it was insufficient to markedly improve the model's performance. Relying heavily on one tree diminished the ensemble effect of the Isolation Forest, limiting the benefits of aggregating the decisions of multiple diverse trees.
These findings indicate that the way weights are transformed and normalized significantly impacts the weight distribution across the ensemble and, consequently, the model's performance. With the sigmoid transformation and normalized weights, the model emphasized a subset of trees, allowing it to focus on more informative trees and improve anomaly detection. In contrast, the softmax transformation with normalized weights resulted in an imbalance where one tree dominated, which did not significantly enhance the model's ability to distinguish anomalies.

These observations underscore the importance of applying the intended weight transformations in SWIFT. Properly transforming and normalizing the weights ensures that they contribute meaningfully to the anomaly score calculation, enabling the model to leverage the ensemble of trees effectively. Understanding how different weight transformations affect the model allows for better tailoring of SWIFT to enhance anomaly detection performance across various datasets.

4.1.7 Outlier Fraction

The impact of anomaly fractions in the test set on anomaly detection models is a key consideration, particularly in real-world settings where anomalies are typically rare. This subsection explores how the proportion of anomalies in the test set influences SWIFT's performance, using its final configuration with softmax-transformed raw weights. The test sets across datasets were initially composed of equal numbers of normal and anomalous instances, resulting in an original anomaly fraction of 50%. To investigate the effect of lower anomaly fractions, we systematically reduced the number of anomalies while keeping all normal instances in the test set. Desired anomaly fractions of $0.01, 0.05, 0.10, 0.20, 0.30$ and 0.50 were considered.
To achieve the desired anomaly fractions, the number of anomalies to include (k_ν) was calculated as:

$$k_\nu = \left\lfloor \frac{\nu}{1-\nu} \times N_{\text{normal}} \right\rfloor$$

where N_{normal} is the number of normal instances in the test set, and ν is the desired anomaly fraction. While this approach calculates k_ν to achieve the desired fraction, the actual anomaly fraction typically differs slightly due to rounding and constraints on the number of available anomalies. The actual anomaly fraction was recalculated to reflect these adjustments:

$$\text{Actual Anomaly Fraction} = \frac{k_\nu}{N_{\text{normal}} + k_\nu}$$

The adjusted test sets were evaluated using SWIFT, and the performance was assessed using the ROC AUC metric. The results are visualized in Figure 4.5 and the full numerical results are in Appendix Table 1.

The results reveal a sigmoidal trend in SWIFT's performance as the anomaly fraction in the test set increases. As the anomaly fraction rises, there is a sharp improvement in performance, followed by a plateau at higher fractions. This trend suggests that once the test set contains enough anomalies to inform test-time training, SWIFT can adapt effectively, regardless of further increases in the anomaly fraction.

The sigmoidal trend observed in SWIFT aligns with insights derived from the DOUST framework [14]. The DOUST paper highlights that effective adaptation in test-time train-

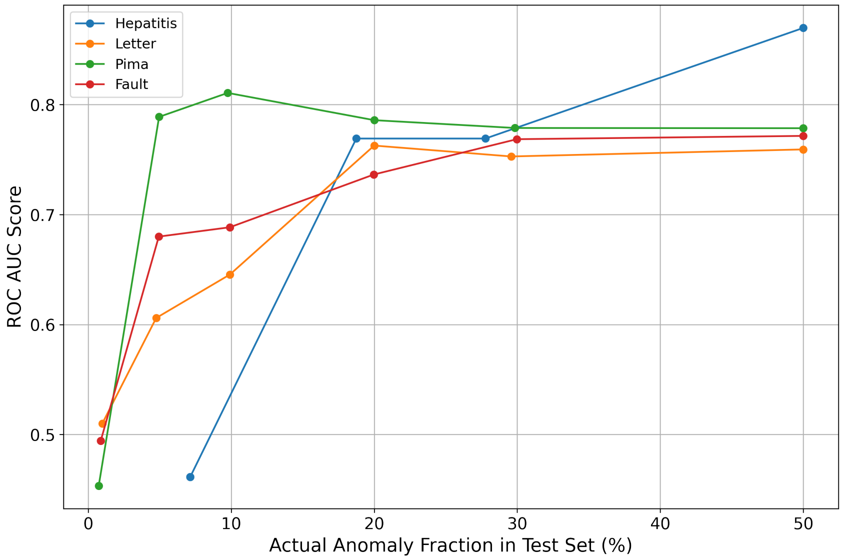


Figure 4.5: SWIFT's Performance Across Varying Actual Anomaly Fractions in the Test Set: The figure illustrates ROC AUC scores achieved by SWIFT as the actual anomaly fraction in the test set varies. A sigmoidal trend emerges, showing improved performance as the anomaly fraction increases, followed by a plateau at higher fractions.

ing approaches relies on two key factors: ensuring the training set is free of anomalies and maintaining a sufficiently large test set size relative to the anomaly fraction. Specifically, the size of the test set (N) needs to be much greater than $1/\nu^2$, where ν represents the outlier fraction.

In our experiments, datasets with larger test sizes exhibited earlier and more pronounced improvements in performance, consistent with the DOUST findings. For instance, the Pima dataset, with 768 total samples and 268 outliers in the test set, generally works with fewer anomalies compared to the smaller Hepatitis dataset, which contains only 80 total samples and 13 outliers in the test set. The sigmoidal pattern in Pima emerges at lower outlier fractions, whereas the Hepatitis dataset requires much higher outlier fractions to achieve similar trends.This difference illustrates how a higher number of test set size can reduce dependency on the anomaly fraction and enable better adaptation during test-time training.

However, addressing the relationship between test set size, anomaly fraction, and model performance is not the primary objective of this thesis. This work serves as an initial exploration, with ongoing research building on the DOUST framework to refine optimization techniques and improve robustness across diverse scenarios.

4.2 kNN with Adaptive Test-time Adjustment through Nonuniform Adaptation (KATANA)

In this section, we explore the application of test-time training to the k-Nearest Neighbors (kNN) algorithm, aiming to enhance anomaly detection performance by adapting the approach used in SWIFT to a non-ensemble method. KATANA, our proposed enhancement of kNN, utilizes test-time training principles to adjust feature weights based on the unlabeled test data, similar to the methodology applied in SWIFT. By integrating test-time training into kNN, we investigate whether this strategy can improve the detection of anomalies in various datasets.

The standard kNN algorithm was implemented using the default settings provided by scikit-learn [22], serving as a baseline for comparison. KATANA introduces feature weighting into kNN, optimizing these weights during the test phase through random search. Two variants of KATANA were developed: KATANA-AUC, which optimizes feature weights to maximize the Receiver Operating Characteristic Area Under the Curve (ROC AUC) between training and test data using pseudo-labels, and KATANA-MD, which aims to maximize the mean difference in anomaly scores between the test and training data.

The algorithms were evaluated on a subset of the datasets used in the previous section, with some datasets omitted due to memory and computational constraints. The value of k was set to 5 for all experiments, balancing sensitivity to local data structures and computational efficiency. Feature scaling was applied using standardization to ensure that all features contributed equally before weighting adjustments.

Table 4.4: ROC AUC Scores for kNN, KATANA-AUC, and KATANA-MD Across 35 Datasets: **Bold** values indicate the highest performance for each dataset. The results demonstrate that both KATANA variants outperform the standard kNN in most datasets, reflecting the benefits of integrating test-time training into kNN through adaptive feature weighting.

Filename	kNN	KATANA-AUC	KATANA-MD
annthyroid	0.9405	**0.9589**	0.956
breastw	**0.9927**	0.9911	0.9901
cardio	0.92	**0.9644**	0.9473
Cardiotocography	0.7125	**0.8687**	0.779
glass	**1.0**	**1.0**	0.9753
Hepatitis	0.8225	0.8402	**0.8521**

Continued on next page

Filename	kNN	KATANA-AUC	KATANA-MD
InternetAds	**0.8409**	**0.8409**	0.8336
Ionosphere	0.9536	0.9547	**0.9644**
letter	0.8682	**0.9162**	0.9084
Lymphography	**1.0**	**1.0**	0.9444
mammography	0.8687	**0.8929**	0.876
PageBlocks	0.9619	0.9682	**0.9687**
Pima	0.7482	0.7584	**0.7588**
satellite	0.8739	**0.8782**	0.8771
satimage-2	0.9984	**0.9986**	0.998
SpamBase	0.8242	0.8732	**0.8901**
Stamps	0.9553	**0.9563**	**0.9563**
thyroid	**0.9895**	0.9891	0.9889
vertebral	0.4044	0.4078	**0.4156**
vowels	0.97	0.9608	**0.972**
WBC	0.99	**1.0**	0.99
WDBC	**1.0**	**1.0**	**1.0**
Wilt	0.6946	0.7664	**0.7886**
wine	0.91	0.92	**0.95**
WPBC	0.5315	**0.5351**	0.5115
yeast	0.4558	0.4585	**0.4592**
optdigits	0.941	0.972	**0.9814**
fault	0.8067	**0.8128**	0.808
mnist	0.9462	**0.9564**	0.9393
musk	**1.0**	**1.0**	**1.0**
shuttle	0.9991	0.999	**0.9993**
landsat	0.767	0.7779	**0.7803**
pendigits	0.9972	0.9991	**0.9995**
smtp	0.9233	0.9233	**0.9367**
speech	0.505	**0.5297**	0.5179
average	0.8604	**0.8763**	0.8718

Table 4.4 presents the ROC AUC scores for each method across the datasets. The standard kNN algorithm achieved varying levels of performance, reflecting its sensitivity to the intrinsic properties of each dataset. Both KATANA-AUC and KATANA-MD consistently outperformed the standard kNN, demonstrating the effectiveness of incorporating test-time

training into kNN through feature weighting.

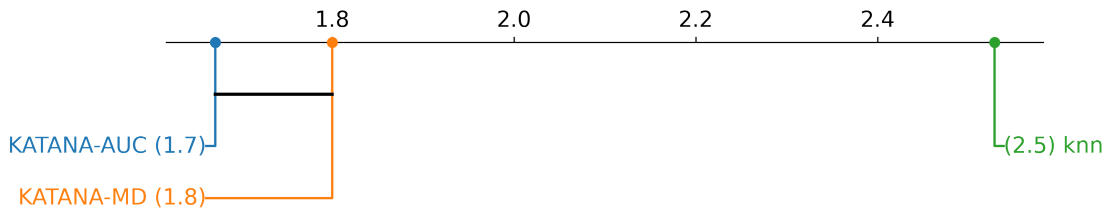


Figure 4.6: Critical Difference Diagram for Variants of the kNN Algorithm: The diagram presents the average ranks of kNN, KATANA-AUC, and KATANA-MD based on ROC AUC scores. Lower ranks indicate better performance. The Wilcoxon signed-rank test with Holm's correction confirms that both KATANA variants significantly outperform the standard kNN algorithm. No statistically significant difference is observed between KATANA-AUC and KATANA-MD, as indicated by the connecting line between them.

To assess the statistical significance of the performance differences, the Wilcoxon signed-rank test with Holm's correction was conducted for multiple comparisons. The average ranks of the methods were calculated, with kNN having an average rank of 2.5286, KATANA-AUC at 1.6714, and KATANA-MD at 1.8. The critical difference diagram in Figure 4.6 illustrates these ranks and indicates that both KATANA variants significantly outperform the standard kNN algorithm. In the diagram, methods connected by a horizontal line are not significantly different at the 0.05 significance level, highlighting that while both KATANA-AUC and KATANA-MD outperform kNN, there is no significant difference between the two KATANA variants.

The adjusted p-values from the Wilcoxon signed-rank test shows that the improvements of KATANA-AUC and KATANA-MD over kNN are statistically significant, with p-values of 0.000086 and 0.0095, respectively. The difference between KATANA-AUC and KATANA-MD is not statistically significant with p-value of 0.4106, suggesting that both optimization objectives are effective in enhancing anomaly detection performance.

The incorporation of test-time training into kNN via KATANA significantly enhances anomaly detection capabilities. By adjusting feature weights based on the unlabeled test data, KATANA effectively adapts to the specific characteristics of the test set, emphasizing features that contribute more to distinguishing between normal and anomalous instances. This approach aligns with the study's objective of enhancing anomaly detection through test-time training and demonstrates that leveraging test data during model adjustment can improve performance in non-parametric methods like kNN.

4.3 Continuous One-class Support Vector Machine Optimization through test-time weight Sampling (COSMOS)

In this section, we extend the application of test-time training to the One-Class Support Vector Machine (OCSVM) algorithm, introducing COSMOS (Continuous One-class Support Vector Machine Optimization through test-time weight Sampling). Similar to SWIFT and KATANA, COSMOS aims to enhance anomaly detection performance by adjusting feature weights based on the unlabeled test data. By integrating test-time training into OCSVM, we explore whether this strategy can also improve the detection of anomalies across various datasets.

The standard OCSVM algorithm was implemented using the default settings provided by scikit-learn [22], with the Radial Basis Function (RBF) kernel. COSMOS introduces feature weighting into the RBF kernel, optimizing these weights during the test phase through random search. Two variants of COSMOS were developed:

- COSMOS-AUC: Optimizes the kernel weights to maximize the ROC AUC between the training and test data using pseudo-labels (0 for training data, 1 for test data).
- COSMOS-MD: Aims to maximize the mean difference in anomaly scores between the test and training data.

Both variants employ random search optimization during the test phase, adjusting the kernel weights based on the unlabeled test data without accessing the true labels.

The algorithms were evaluated on a subset of the datasets used in previous sections, with some datasets omitted due to memory and computational constraints. Feature scaling was applied using standardization. Examining the ROC AUC scores presented in Table 4.5, it can be seen that the best-performing method varies across different datasets. In many cases, COSMOS-AUC and COSMOS-MD achieve the highest ROC AUC, indicating improved performance over the standard OCSVM without weights. However, there are also several datasets where the standard OCSVM achieved the highest ROC AUC.
These observations suggest that the performance improvements offered by COSMOS are not consistent across all datasets. The best-performing method varies depending on the dataset, and in some cases, the standard OCSVM without weights achieves the best performance. This variability indicates that while COSMOS has the potential to enhance anomaly detection performance through test-time training and kernel weighting, it does not universally outperform the standard OCSVM across all datasets.

Table 4.5: ROC AUC Scores for OCSVM, COSMOS-AUC, and COSMOS-MD Across 39 Datasets: **Bold** values indicate the highest performance for each dataset. The results show that COSMOS frequently achieves the best performance, reflecting its effectiveness in optimizing feature weights through test-time training.

Filename	OCSVM	COSMOS-AUC	COSMOS-MD
annthyroid	0.8724	**0.9267**	0.9128
breastw	0.992	**0.9925**	**0.9925**
cardio	0.953	**0.9733**	0.9655
Cardiotocography	0.8067	**0.888**	0.8649
glass	**0.6667**	0.5432	0.5062
Hepatitis	**0.8462**	0.8402	0.7988
InternetAds	0.7992	**0.8158**	0.7912
Ionosphere	0.9429	0.9307	**0.9563**
letter	0.609	**0.6302**	0.625
Lymphography	**1.0**	**1.0**	**1.0**
mammography	0.8939	0.9065	**0.911**
PageBlocks	0.9403	0.9448	**0.9461**
Pima	0.6917	**0.7042**	0.6899
satellite	**0.7569**	0.6964	0.6927
satimage-2	**0.9964**	0.9891	0.9839
SpamBase	0.8031	0.8496	**0.8565**
Stamps	0.9324	0.9438	**0.9688**
thyroid	0.9847	0.985	**0.9865**
vertebral	**0.5189**	0.4756	0.41
vowels	**0.8148**	0.7712	0.7736
WBC	**0.99**	0.98	0.98
WDBC	**1.0**	**1.0**	**1.0**
Wilt	0.309	0.3276	**0.33**
wine	0.92	**0.95**	**0.95**
WPBC	0.5138	0.5066	**0.5206**
yeast	0.4486	**0.4951**	0.4476
optdigits	0.5756	**0.898**	0.7934
fault	0.6116	**0.6145**	0.6092

Continued on next page

Filename	OCSVM	COSMOS-AUC	COSMOS-MD
mnist	0.9117	**0.9286**	0.9175
musk	**1.0**	**1.0**	**1.0**
shuttle	**0.9962**	0.9954	0.9943
landsat	**0.4594**	0.4107	0.4113
pendigits	**0.9714**	0.9569	0.9387
smtp	**0.85**	0.8322	0.8456
campaign	0.7788	**0.8676**	0.7462
magic.gamma	0.7422	**0.7513**	0.7341
ALOI	0.5597	**0.5715**	0.5674
speech	0.4805	**0.4897**	0.4757
backdoor	0.6148	**0.7923**	0.5836
average	0.7834	**0.7993**	0.7815

The statistical analysis reflects this variability in performance across datasets. Despite COSMOS-AUC achieving the highest average rank (1.7436) and COSMOS-MD having the lowest average rank (2.1538), the Wilcoxon signed-rank test with Holm's correction indicates that the differences between COSMOS-AUC and OCSVM are not statistically significant (adjusted p-value = 0.2181). Similarly, the difference between COSMOS-MD and OCSVM is not statistically significant (adjusted p-value = 0.7653). However, the adjusted p-value between COSMOS-AUC and COSMOS-MD is 0.0202, indicating a statistically significant difference between these two methods at the 0.05 significance level.

The critical difference (CD) diagram in Figure 4.7 visualizes these relationships. Methods not connected by horizontal lines are significantly different. In this case, COSMOS-AUC is not connected to COSMOS-MD, indicating that COSMOS-AUC significantly outperforms COSMOS-MD across the datasets evaluated. Both COSMOS-AUC and COSMOS-MD are connected to OCSVM, indicating that their performance differences with OCSVM are not statistically significant.

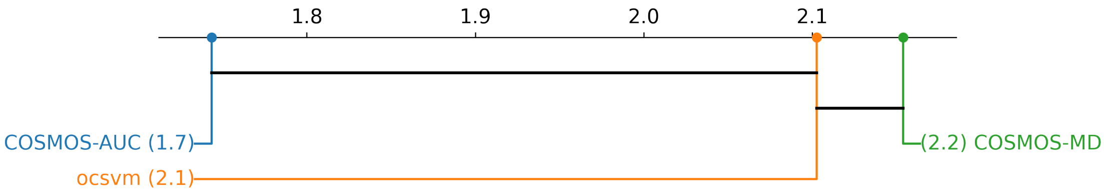


Figure 4.7: Critical Difference Diagram for Variants of the OCSVM Algorithm: The diagram displays the average ranks of OCSVM, COSMOS-AUC, and COSMOS-MD based on ROC AUC scores. Lower ranks indicate better performance. The Wilcoxon signed-rank test with Holm's correction indicates a statistically significant difference between COSMOS-AUC and COSMOS-MD, with no significant differences observed between the standard OCSVM and either COSMOS variant.

These results suggest that while COSMOS-MD alone shows the lowest average rank (2.1538) and ROC AUC score (0.7815), the combined performance of COSMOS-AUC and COSMOS-MD demonstrates the clear potential of COSMOS as a test-time training approach. In many datasets, COSMOS-AUC and COSMOS-MD together outperform the standard OCSVM, indicating that test-time training through kernel weighting can effectively enhance anomaly detection. This combined success highlights that COSMOS, as a framework, has significant room for development and improvement.

The possibility for further enhancement is promising, especially since, unlike SWIFT, COSMOS did not yet explore other configurations such as alternative weight transformations, optimization parameters, or initial weight settings.

4.4 Final Comparison Between Different Models

This study is inspired by the principles behind DOUST, a deep learning-based method that innovatively enhances anomaly detection through test-time training by adjusting model parameters to better identify anomalies during testing [14]. Extending this concept to traditional algorithms, test-time training was applied to Isolation Forest, kNN, and OCSVM, resulting in the development of SWIFT, KATANA-AUC, and COSMOS-AUC. The results demonstrate that these enhanced methods generally outperform their respective baseline models.

The final comparison in this section focuses on 35 datasets, chosen to ensure fairness and consistency as KATANA was applied to the smallest subset due to computational constraints. DOUST serves as the benchmark model, though for clarity and emphasis on traditional algorithms, the highest ROC AUC values in Table 4.6 are highlighted only among the remaining methods, excluding DOUST. This approach allows the evaluation to isolate and showcase the contributions of test-time training enhancements.

Table 4.6: ROC AUC Scores for Isolation Forest, kNN, OCSVM, KATANA-AUC, COSMOS-AUC, SWIFT with Softmax Transformation, and DOUST Across 35 Datasets: The table compares ROC AUC scores across seven models, highlighting the highest score per dataset among traditional models and their variants in **bold** (excluding DOUST). The results demonstrate that SWIFT and KATANA-AUC consistently achieve competitive scores, reflecting the effectiveness of test-time training enhancements in anomaly detection.

Filename	ifor	ocsvm	knn	KATANA	COSMOS	SWIFT	DOUST
annthyroid	0.9168	0.8724	0.9405	0.9589	0.9267	**0.9944**	0.9956
breastw	**0.9975**	0.992	0.9927	0.9911	0.9925	0.9822	0.9884
cardio	0.9362	0.953	0.92	0.9644	0.9733	**0.9883**	0.9929
Cardiotocography	0.8258	0.8067	0.7125	0.8687	0.888	**0.9443**	0.9578
glass	0.8148	0.6667	**1.0**	**1.0**	0.5432	**1.0**	0.9877
Hepatitis	0.7929	0.8462	0.8225	0.8402	0.8402	**0.8817**	0.8107
InternetAds	0.4357	0.7992	**0.8409**	**0.8409**	0.8158	0.6792	0.9085
Ionosphere	0.8644	0.9429	0.9536	**0.9547**	0.9307	0.9465	0.98
letter	0.6103	0.609	0.8682	**0.9162**	0.6302	0.7594	0.9499
Lymphography	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	0.9444	1.0
mammography	0.8931	0.8939	0.8687	0.8929	0.9065	**0.9211**	0.9495

Continued on next page

Filename	ifor	ocsvm	knn	KATANA	COSMOS	SWIFT	DOUST
PageBlocks	0.9273	0.9403	0.9619	**0.9682**	0.9448	0.9484	0.9832
Pima	0.7368	0.6917	0.7482	0.7584	0.7042	**0.7785**	0.7361
satellite	0.8052	0.7569	0.8739	0.8782	0.6964	**0.8837**	0.9545
satimage-2	0.9956	0.9964	0.9984	**0.9986**	0.9891	0.9923	0.9984
SpamBase	0.8235	0.8031	0.8242	0.8732	0.8496	**0.9334**	0.9257
Stamps	0.9147	0.9324	0.9553	0.9563	0.9438	**0.9844**	0.9854
thyroid	0.9894	0.9847	0.9895	0.9891	0.985	**0.9917**	0.9985
vertebral	0.39	0.5189	0.4044	0.4078	0.4756	**0.8444**	0.8911
vowels	0.7816	0.8148	**0.97**	0.9608	0.7712	0.91	0.9924
WBC	0.99	0.99	0.99	**1.0**	0.98	0.83	0.97
WDBC	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	1.0
Wilt	0.4854	0.309	0.6946	0.7664	0.3276	**0.9067**	0.9941
wine	0.87	0.92	0.91	0.92	0.95	**1.0**	1.0
WPBC	0.5781	0.5138	0.5315	0.5351	0.5066	**0.5957**	0.656
yeast	0.3953	0.4486	0.4558	0.4585	0.4951	**0.6835**	0.7262
optdigits	0.6372	0.5756	0.941	0.972	0.898	**0.9915**	0.9997
fault	0.6519	0.6116	0.8067	**0.8128**	0.6145	0.7716	0.8243
mnist	0.7962	0.9117	0.9462	**0.9564**	0.9286	0.9421	0.9872
musk	0.9716	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	1.0
shuttle	0.9968	0.9962	**0.9991**	0.999	0.9954	0.9967	0.9953
landsat	0.5996	0.4594	0.767	0.7779	0.4107	**0.8141**	0.9542
pendigits	0.9579	0.9714	0.9972	**0.9991**	0.9569	0.9894	1.0
smtp	0.9122	0.85	0.9233	0.9233	0.8322	**0.9722**	0.8378
speech	0.4859	0.4805	0.505	0.5297	0.4897	**0.6006**	0.7396
Average	0.7937	0.796	0.8604	0.8763	0.8055	**0.8972**	0.933449

The results in Table 4.6 reveal that SWIFT and KATANA-AUC frequently achieve the highest ROC AUC scores. SWIFT, employing a strategic tree-weighting mechanism, and KATANA-AUC, which adapts feature weights within kNN, both consistently exhibit competitive performance. The critical difference diagram in Figure 4.8 confirms their success, showing that SWIFT and KATANA-AUC share the same average rank (3.1). This alignment is further supported by the p-values heatmap in Figure 4.9, which indicates no statistically significant difference between SWIFT and KATANA-AUC. However, the heatmap reveals significant differences between KATANA-AUC and the baseline kNN, underscoring the substantial improvements introduced by test-time training.

COSMOS-AUC, designed to integrate test-time training into OCSVM through kernel weighting, displays mixed results. While it occasionally outperforms its baseline model, its performance varies significantly across datasets. This highlights the need for further refinement of its kernel optimization and weighting strategies to achieve greater consistency.
Although DOUST consistently outperforms all other methods due to its deep learning architecture, SWIFT and KATANA-AUC emerge as practical alternatives. These models combine competitive performance, as reflected in the highest average ROC AUC values for SWIFT in Table 4.6, with advantages such as simplicity, computational efficiency, and reliability.

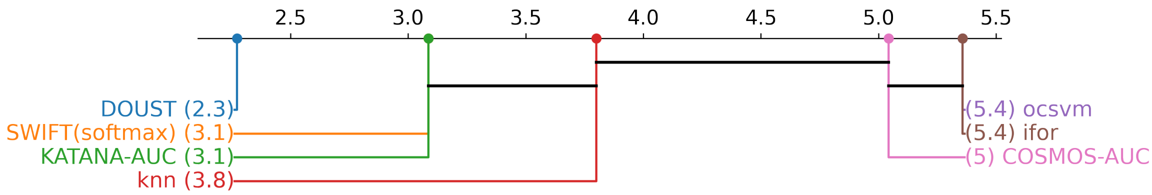


Figure 4.8: Critical Difference Diagram for Model Comparisons: The figure presents average ranks of Isolation Forest, kNN, OCSVM, KATANA-AUC, COSMOS-AUC, SWIFT with Softmax Transformation, and DOUST based on ROC AUC performance. Lower ranks indicate better performance. Models not connected by horizontal lines are significantly different according to the Wilcoxon signed-rank test with Holm's correction, highlighting the competitiveness of SWIFT and KATANA-AUC.

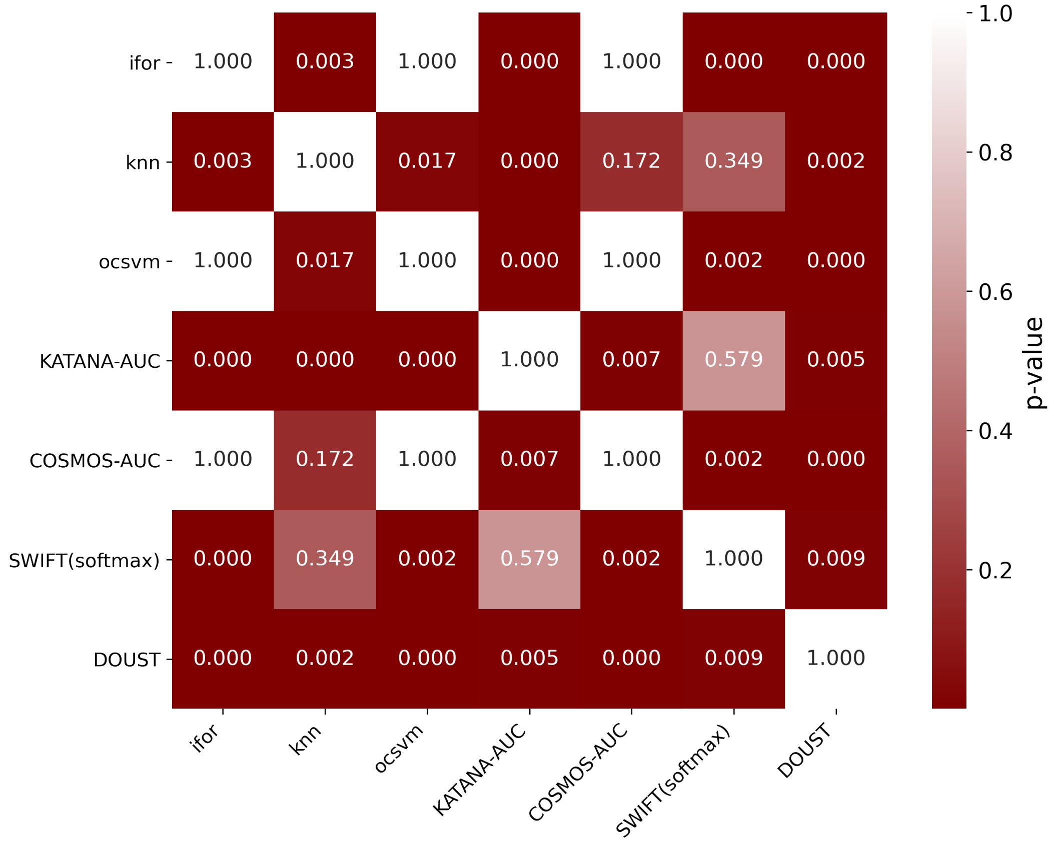


Figure 4.9: p-value Heatmap for Pairwise Model Comparisons: The heatmap illustrates the p-values from pairwise comparisons among the evaluated models using the Wilcoxon signed-rank test with Holm's correction. Lower p-values (darker regions) indicate statistically significant differences.

Chapter 5

Conclusions

This thesis explored the adaptation of test-time training principles to traditional anomaly detection algorithms, with the goal of enhancing their performance while maintaining computational efficiency and interpretability. Inspired by the DOUST framework, the study introduced novel extensions to Isolation Forest, kNN, and OCSVM. The proposed methods—SWIFT, KATANA, and COSMOS—demonstrated the feasibility of incorporating adaptive mechanisms into classical models to improve their anomaly detection capabilities.

A major contribution of this work was the development of SWIFT, which significantly outperformed its baseline model, the standard Isolation Forest. SWIFT achieved an impressive improvement in average ROC AUC scores, increasing from 0.7916 for the standard model to 0.8947 in its final version. This result underscores SWIFT's effectiveness in adapting tree-specific weights during the test phase, consistently delivering robust performance across a wide range of datasets.

KATANA introduced test-time feature weighting to kNN, resulting in statistically significant improvements compared to the baseline. While the average ROC AUC score of KATANA-AUC increased from 0.8604 for the standard kNN to 0.8763, this improvement was not as dramatic as the gains achieved by SWIFT. However, the statistical significance of KATANA's enhancements underscores its effectiveness in leveraging adaptive feature weighting to improve anomaly detection performance, addressing specific dataset characteristics in a meaningful way. This demonstrates KATANA's capacity to enhance kNN's applicability and robustness.

COSMOS extended test-time training to kernel-based methods like OCSVM, offering valuable insights into the challenges of generalizing such techniques. While COSMOS did not achieve statistically significant improvements over the standard OCSVM, its ROC AUC scores on many datasets indicated notable promise. These results suggest that further refinement of kernel weighting strategies and optimization processes could unlock COSMOS's full potential and broaden its applicability.

In addition to developing these models, this thesis conducted a detailed investigation into factors influencing test-time training effectiveness. For instance, SWIFT demonstrated the impact of weight transformation functions and initialization strategies on model performance. These findings emphasize the importance of systematically optimizing model components to maximize the benefits of test-time training.

Limitations & Future Work

While this work demonstrates promising advancements, it also identifies several limitations that provide opportunities for further research. The dataset constraints, particularly for KATANA, limited the number of datasets used in comparisons due to computational requirements, which may affect the generalizability of findings. The variability in COSMOS-AUC's performance across datasets underscores the need for more robust kernel weighting and optimization strategies to improve consistency. Additionally, further investigation into automating optimization processes for hyperparameters, including weight transformation and initialization strategies, could enhance the scalability and applicability of these models.

Future research could focus on refining COSMOS through improved kernel weighting mechanisms and alternative loss functions to address its variability. Extending the methods to handle other data modalities, such as time-series, image, or graph data, could significantly expand their applicability. Adapting these models for real-time or streaming data would also be a valuable direction, enabling dynamic anomaly detection in evolving environments.

In conclusion, this thesis successfully demonstrated the potential of test-time training to enhance traditional anomaly detection algorithms. By bridging the gap between classical methods and adaptive frameworks, this work lays the foundation for future innovations in resource-efficient, high-performance anomaly detection. These contributions advance both the theoretical understanding and practical implementation of test-time training, offering significant opportunities for addressing real-world challenges in anomaly detection.

Bibliography

[1] Charu C. Aggarwal. 'An Introduction to Outlier Analysis'. In: *Outlier Analysis*. Cham: Springer International Publishing, 2017, pp. 1–34. ISBN: 978-3-319-47578-3. DOI: 10.1007/978-3-319-47578-3_1. URL: https://doi.org/10.1007/978-3-319-47578-3_1 (cit. on p. 7).

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738 (cit. on p. 10).

[3] Abdenour Bounsiar and Michael Madden. 'One-Class Support Vector Machines Revisited'. In: May 2014. DOI: 10.1109/ICISA.2014.6847442 (cit. on pp. 3, 15).

[4] Varun Chandola, Arindam Banerjee and Vipin Kumar. 'Anomaly Detection: A Survey'. In: *ACM Comput. Surv.* 41 (July 2009). DOI: 10.1145/1541880.1541882 (cit. on pp. 1, 3).

[5] Yin-Wen Chang, Cho-Jui Hsieh, Kai-Wei Chang, Michael Ringgaard and Chih-Jen Lin. 'Training and Testing Low-degree Polynomial Data Mappings via Linear SVM'. In: *Journal of Machine Learning Research* 11.48 (2010), pp. 1471–1490. URL: http://jmlr.org/papers/v11/chang10a.html (cit. on p. 16).

[6] Kittipong Chomboon, P. Chujai, Pongsakorn Teerarassammee, Kittisak Kerdprasop and Nittaya Kerdprasop. 'An Empirical Study of Distance Metrics for k-Nearest Neighbor Algorithm'. In: (2015), pp. 280–285. DOI: 10.12792/ICIAE2015.051 (cit. on p. 12).

[7] Tom Fawcett. 'Introduction to ROC analysis'. In: *Pattern Recognition Letters* 27 (June 2006), pp. 861–874. DOI: 10.1016/j.patrec.2005.10.010 (cit. on p. 17).

[8] B. German. *Glass Identification*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5V 1987 (cit. on p. 21).

[9] Prasanta Gogoi, D.K. Bhattacharyya, B. Borah and Jugal K. Kalita. 'A Survey of Outlier Detection Methods in Network Anomaly Identification'. In: *The Computer Journal* 54.4 (2011), pp. 570–588. DOI: 10.1093/comjnl/bxr026 (cit. on p. 1).

[10] Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016 (cit. on p. 10).

[11] Xiaoyi Gu, Leman Akoglu and Alessandro Rinaldo. *Statistical Analysis of Nearest Neighbor Methods for Anomaly Detection*. 2019. arXiv: `1907.03813 [stat.ML]`. URL: `https://arxiv.org/abs/1907.03813` (cit. on pp. 3, 12).

[12] Songqiao Han, Xiyang Hu, Hailiang Huang, Mingqi Jiang and Yue Zhao. *ADBench: Anomaly Detection Benchmark*. 2022. arXiv: `2206.09426 [cs.LG]`. URL: `https://arxiv.org/abs/2206.09426` (cit. on pp. 21, 23).

[13] Sture Holm. 'A Simple Sequentially Rejective Multiple Test Procedure'. In: *Scandinavian Journal of Statistics* 6.2 (1979), pp. 65–70. ISSN: 03036898, 14679469. URL: `http://www.jstor.org/stable/4615733` (visited on 25/11/2024) (cit. on p. 19).

[14] Simon Klüttermann and Emmanuel Müller. *About Test-time training for outlier detection*. 2024. arXiv: `2404.03495 [cs.LG]`. URL: `https://arxiv.org/abs/2404.03495` (cit. on pp. 1, 4, 8, 14, 35, 44).

[15] D. Kraft. *A Software Package for Sequential Quadratic Programming*. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR, 1988. URL: `https://books.google.de/books?id=4rKaGwAACAAJ` (cit. on p. 11).

[16] Fei Tony Liu, Kai Ting and Zhi-Hua Zhou. 'Isolation-Based Anomaly Detection'. In: *ACM Transactions on Knowledge Discovery From Data - TKDD* 6 (Mar. 2012), pp. 1–39. DOI: `10.1145/2133360.2133363` (cit. on pp. 5, 7).

[17] Fei Tony Liu, Kai Ming Ting and Zhi-Hua Zhou. 'Isolation Forest'. In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, pp. 413–422. DOI: `10.1109/ICDM.2008.17` (cit. on pp. 3, 5, 6).

[18] Tianyuan Lu, Lei Wang and Xiaoyong Zhao. 'Review of Anomaly Detection Algorithms for Data Streams'. In: *Applied Sciences* 13.10 (2023). ISSN: 2076-3417. DOI: `10.3390/app13106353`. URL: `https://www.mdpi.com/2076-3417/13/10/6353` (cit. on p. 1).

[19] Zhi Lu, Jing Hao, Min Zhang and Yuqing Yang. 'A Weighted Extended Isolation Forest Method Based on Hierarchical Feature Selection'. In: *2024 7th International Conference on Computer Information Science and Application Technology (CISAT)*. 2024, pp. 1140–1145. DOI: `10.1109/CISAT62382.2024.10695247` (cit. on p. 3).

[20] Younes Manzali, Khalidou Barry, Rachid Flouchi, Youssef Balouki and Mohamed Elfar. 'A feature weighted K-nearest neighbor algorithm based on association rules'. In: *Journal of Ambient Intelligence and Humanized Computing* 15 (Apr. 2024), pp. 1–14. DOI: 10.1007/s12652-024-04793-z (cit. on p. 3).

[21] Minjae Ok, Simon Klüttermann and Emmanuel Müller. *Exploring the Impact of Outlier Variability on Anomaly Detection Evaluation Metrics*. 2024. arXiv: 2409.15986 [cs.LG]. URL: https://arxiv.org/abs/2409.15986 (cit. on p. 17).

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay. 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on pp. 7, 21, 37, 40).

[23] David M. W. Powers. *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*. 2020. arXiv: 2010.16061 [cs.LG]. URL: https://arxiv.org/abs/2010.16061 (cit. on p. 17).

[24] Bernhard Schölkopf, John Platt, John Shawe-Taylor, Alexander Smola and Robert Williamson. 'Estimating Support of a High-Dimensional Distribution'. In: *Neural Computation* 13 (July 2001), pp. 1443–1471. DOI: 10.1162/089976601750264965 (cit. on p. 15).

[25] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros and Moritz Hardt. *Test-Time Training with Self-Supervision for Generalization under Distribution Shifts*. 2020. arXiv: 1909.13231 [cs.LG]. URL: https://arxiv.org/abs/1909.13231 (cit. on pp. 4, 8).

[26] Maksim A. Terpilowski. 'scikit-posthocs: Pairwise multiple comparison tests in Python'. In: *Journal of Open Source Software* 4.36 (2019), p. 1169. DOI: 10.21105/joss.01169. URL: https://doi.org/10.21105/joss.01169 (cit. on p. 32).

[27] Pauli Virtanen et al. 'SciPy 1.0: fundamental algorithms for scientific computing in Python'. In: *Nature Methods* 17.3 (Feb. 2020), pp. 261–272. ISSN: 1548-7105. DOI: 10.1038/s41592-019-0686-2. URL: http://dx.doi.org/10.1038/s41592-019-0686-2 (cit. on p. 11).

[28] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen and Trevor Darrell. *Tent: Fully Test-time Adaptation by Entropy Minimization*. 2021. arXiv: 2006.10726 [cs.LG]. URL: https://arxiv.org/abs/2006.10726 (cit. on pp. 1, 4, 8).

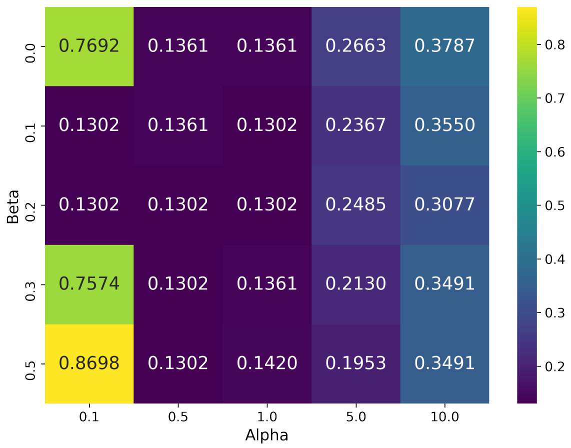
[29] Dietrich Wettschereck, David Aha and Takao Mohri. 'A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms'. In: *Artificial Intelligence Review* 11 (June 2000). DOI: `10.1023/A:1006593614256` (cit. on pp. 3, 13).

[30] Frank Wilcoxon. 'Individual Comparisons by Ranking Methods'. In: *Biometrics Bulletin* 1.6 (1945), pp. 80–83. ISSN: 00994987. URL: `http://www.jstor.org/stable/3001968` (visited on 25/11/2024) (cit. on p. 18).

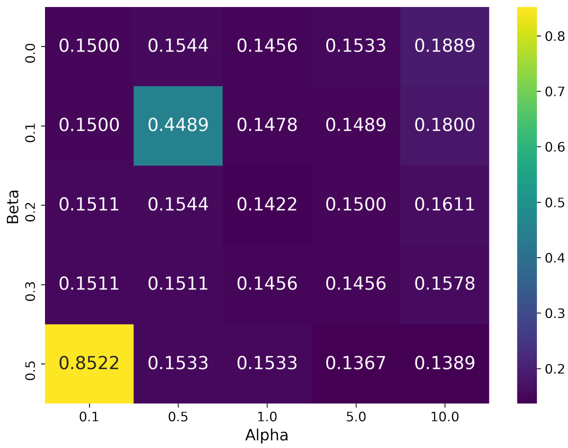
[31] Jinhong Yang, Tingquan Deng and Ran Sui. 'An Adaptive Weighted One-Class SVM for Robust Outlier Detection'. In: *Proceedings of the 2015 Chinese Intelligent Systems Conference*. Ed. by Yingmin Jia, Junping Du, Hongbo Li and Weicun Zhang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 475–484. ISBN: 978-3-662-48386-2 (cit. on p. 3).

[32] Yue Zhao, Zain Nasrullah and Zheng Li. 'PyOD: A Python Toolbox for Scalable Outlier Detection'. In: *CoRR* abs/1901.01588 (2019). arXiv: `1901.01588`. URL: `http://arxiv.org/abs/1901.01588` (cit. on p. 7).

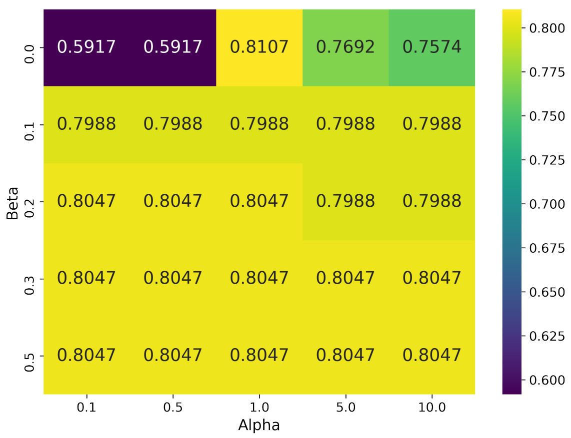
Appendix



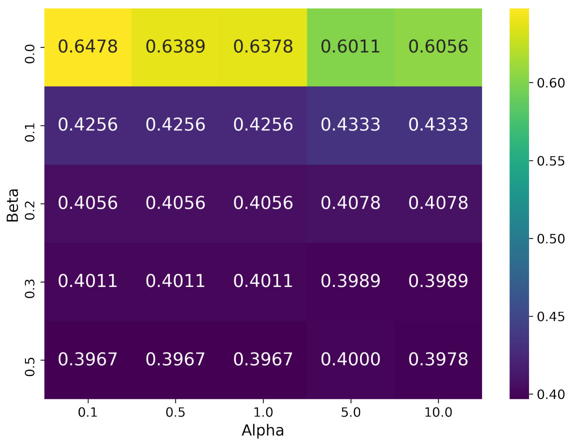
(a) **`Hepatitis`** Sigmoid Transformation with raw weights



(b) **`vertebral`** Sigmoid Transformation with raw weights



(c) **`Hepatitis`** Softmax Transformation with transformed and normalized weights



(d) **`vertebral`** Softmax Transformation with transformed and normalized weights

Figure 1: ROC AUC Scores for Different Weight Transformations on the `Hepatitis` and `vertebral` Datasets

Table 1: ROC AUC Scores Across Different Outlier Fractions for Selected Datasets: The table presents ROC AUC scores for the `Hepatitis`, `Letter`, `Pima`, and `Fault` datasets across varying actual outlier fractions.

Dataset	**Actual Outlier Fractions (%)**	**ROC AUC Scores**
Hepatitis	7.14, 7.14, 7.14, 18.75, 27.78, 50.00	0.4615, 0.4615, 0.4615, 0.7692, 0.7692, 0.8817
Letter	0.99, 4.76, 9.91, 20.00, 29.58, 50.00	0.5100, 0.6060, 0.6455, 0.7628, 0.7529, 0.7594
Pima	0.74, 4.96, 9.76, 20.00, 29.84, 50.00	0.4534, 0.7889, 0.8107, 0.7859, 0.7788, 0.7785
Fault	0.88, 4.94, 9.91, 19.98, 29.97, 50.00	0.4943, 0.6800, 0.6885, 0.7365, 0.7686, 0.7716

Eidesstattliche Versicherung

(Affidavit)

Ok, Minjae
Name, Vorname
(surname, first name)

229916
Matrikelnummer
(student ID number)

☐ Bachelorarbeit
(Bachelor's thesis)

■ Masterarbeit
(Master's thesis)

Titel
(Title)

Enhancing Anomaly Detection through Test-Time Training

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.

Darmstadt, den 08.12.2024
Ort, Datum
(place, date)

옥민재
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:*

Darmstadt, den 08.12.2024
Ort, Datum
(place, date)

옥민재
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.**